A Deep Neural Network for Audio Classification with a Classifier Attention Mechanism

Haoye Lu[a,*], Haolong Zhang[a], Amit Nayak[a]

[a] *University of Ottawa, Ottawa, Ontario, Canada, K1N 6N5*

Abstract

Audio classification is considered as a challenging problem in pattern recognition. Recently, many algorithms have been proposed using deep neural networks. In this paper, we introduce a new attention-based neural network architecture called Classifier-Attention-Based Convolutional Neural Network (CAB-CNN). The algorithm uses a newly designed architecture consisting of a list of simple classifiers and an attention mechanism as a classifier selector. This design significantly reduces the number of parameters required by the classifiers and thus their complexities. In this way, it becomes easier to train the classifiers and achieve a high and steady performance. Our claims are corroborated by the experimental results. Compared to the state-of-the-art algorithms, our algorithm achieves more than 10% improvements on all selected test scores.

Keywords: audio classification, attention-based, deep neural network

1. Introduction

Sounds contain rich information and help people sense the environments around them. People are able to recognize complex sounds and filter out the meaningful information. In this way, useless noise is dropped and the raw information is distilled. Today, sensors can easily collect tons of raw audio data; however, processing them to get meaningful information remains arduous. Many researchers hope to design a human-like machine to alleviate this kind of problems, and one important and fundamental branch of it is called audio classification.

Currently, audio classification is used to distinguish audio samples by key words, intonation and accent. Audio classification can lead to real time transcription and translation of audio. The majority of audio classification research focuses on a specific classification task to obtain high accuracy. However, due to

*Corresponding author
Email address: hlu044@uottawa.ca (Haoye Lu)
URL: hzhan006@uottawa.ca (Haolong Zhang), anaya085@uottawa.ca (Amit Nayak)

the complexity of audio data, various techniques must be employed to analyse the data.

In order to effectively classify the audio data, the features must be extracted from the audio sample. Three widely used techniques for audio classification research are Mel-Frequency Cepstral Coefficient (MFCC), Zero-Crossing Rate (ZCR) and Linear Predictive Coding (LPC) [1, 2]. MFCCs have been used for feature extraction to improve speaker recognition. After this, Support Vector Machine (SVM) and Gaussian Mixture Model (GMM) are applied to do the classifications [3, 4].

Recently, Deep neural networks (DNNs) and more specifically Convolutional neural networks (CNNs) have been used to automatically learn feature representations from complex data [5]. This universal technique has been applied in many areas to replace ad hoc function designs and has shorten a decade-long development period to a few months. The related applications have been seen in the audio classification area. For example, DNNs in conjunction with transformed MFCCs have been used to improve the accuracy of speaker age classification [1]. Other researchers have used DNNs for cepstral feature extraction of audio samples [6]. CNNs are able to deal with complex nonlinear mappings and can share weights across the input, which allows for translation invariance of the input.

Most of the DNN-based algorithms need to convert the original audios into spectrograms before processing them. Spectrograms provide a visual representation of the frequencies with respect to time. Methods that use a time distributed approach [7, 8] split the spectrogram into frames to create a time-distributed spectrogram. The time-distributed spectrogram is used as the input into the CNN to train the model to distinguish local features at different time steps. A different approach [9] to audio classification splits the spectrogram along frequency to create a frequency-distributed spectrogram. Using this approach allows for the model to learn features based on various frequencies.

Although the models based on spectrograms have achieved great successes, there are some intrinsic problems that are hard to eliminate. In particular, the function to generate spectrograms is independent from the later classification process. Practitioners must generate spectrograms from the audios before training the models. As a result, the spectrogram-generating function cannot be jointly optimized with the classification networks, which would considerably harm the performances of the algorithms. Besides, the spectrogram-generating process spans the originally one-dimensional audio data into three dimensions (one for time, one for frequency and one for three color channels: red, green and red), which makes the representation sparse (thus hard to learn) and adds extra noises that could interfere the later classification process.

In this paper, we propose a new audio classification algorithm with an attention mechanism for the selection of the audio classifiers. We name the algorithm Classifier-Attention-Based CNN (CAB-CNN). Compared to the other DNN-based algorithms,

1. unlike the attention-based algorithm proposed by Wu et al. [10], our at-

tention unit dynamically assigns importance weights to a list of classifiers rather than attend to different frequencies and time intervals. This design let a single classifier only need to focus on a small portion of features. So, a classifier only needs to possess a small model capacity and does not need to have a large number of capacity. Therefore, the classifiers are much easier to be trained.

2. since every single classifier only needs to learn a simple feature for distinguishing accents in principle. They can be trained easily and fast. Therefore, the CAB-CNN model is more robust and have more stable performance in the independent training and testing processes.

We test the CAB-CNN model using UT-Podcast corpus [11] by implementing an accent classification task. The test results corroborate what we have just claimed. Compared to the state-of-the-art algorithms [12, 10], the CAB-CNN model has over 10% improvements on all test scores and has reached 95.99% test accuracy.

The rest of the paper is organized as follows. In Section 2, we introduce more techniques that have been used in the audio classification problems. In Section 3, we propose our new algorithm formally. In Section 4, we test our algorithm on UT-Podcast corpus and compare its performance with some popular and the state-of-the-art algorithms. We conclude the paper in Section 5 and finally, we discuss some potential future work in Section 6.

2. Related Work

The audio classification algorithms can be generally divided into two parts: the feature extraction part and the classification part [13]. The feature extraction parts are mostly implemented by CNNs as they can efficiently extract characteristics from raw data [14, 15].

The implementations of CNNs can be grouped into two classes based on how they preprocess the input audios: waveforms [14, 16] or spectrograms [10, 15, 17]. The waveform-based method process the input data as an 1D data array directly, while the spectrogram-based implementations have to convert the raw audio files into spectrograms by Fourier transform first. Compared to the waveform-based method, the spectrogram-based methods manually extract frequency informations and plot them as a heat map. In other words, the strengths of the frequencies at each moment are indicated by the color or brightness. This preprocessing may facilitate CNNs to find frequency related features. In comparison, the waveform-based algorithm processes the raw audio files directly without involving plotting any graphs which are likely to introduce extra noises and/or make the data structure sparse. Besides, the entire algorithm (data extraction and classification parts) can then be trained together and tuned jointly while the spectrogram-based methods have no control upon the spectrogram plotting part.

The Multi-task Learning (MTL) method [18] has been used for multiple audio classification tasks. MTL is a focus of machine learning in which mul-

tiple learning tasks are solved simultaneously, improving the accuracy of multiple classifications by narrowing the gap between the training and testing errors [19]. By employing a shared hidden layer, neural networks can use the MTL method [18]. Studies have shown that MTL-SVM based models have better performance than task-specific SVM models [20]. The reason why the MTL works is that those factors that explain the variation of data could be shared among various tasks.

Despite many audio classification techniques being effective for a specific classification class, researchers have used convolutional deep belief networks (CDBNs) to classify audio data with high performance over multiple audio classification tasks [21]. The use of DNNs for cepstral feature extraction has also been used for multiple audio classification tasks [6].

Researchers are using deep residual networks (ResNets) along with a gate mechanism in order to extract feature representation in audio data. This was shown to be more effective with multiple audio classification tasks and has achieved higher accuracy compared to task specific models that were trained separately [22].

3. Approach

In this section, we introduce our classification algorithm with an attention mechanism for the selection of the audio classifiers. We name the algorithm Classifier-Attention-Based CNN (CAB-CNN), and its complete architecture is plotted in Fig 1.

The key part of the CAB-CNN algorithm is the attention-based classifier block (ACB) containing n classifiers and an attention unit (see Fig 2). In order to improve both training and statistical efficiency, we do not feed original audio to the block directly. Instead, we use a "distilled" representation a, which is generated by feeding the original data into a stack of 1D-CNN followed by MaxPooling layers (see Fig 3). We do so in order to

1. enhance local features,
2. decrease the input data size of the classifier ACB, and
3. preserve a one-to-one correspondence relationship (over time axis) between the original data and the generated representations.

Roughly speaking, as CNN preserves the spatial information of the audio file, it partitions the original audio file into p intervals (the value of p depends on the architectures of CNNs, MaxPooling layers and the length of the input file) and applying the same transformation to extract features. Notice that these intervals would have some overlaps, which depends on the architectures of CNNs and MaxPooling layers. In Fig 3, we can observe that the first node of the MaxPooling layer covers the first three inputs of the raw data array while the second node covers the second to the fifth. Although the overlaps may cause some ambiguity of what the extracted features represent, they have no significant effect on the performance of the algorithm.

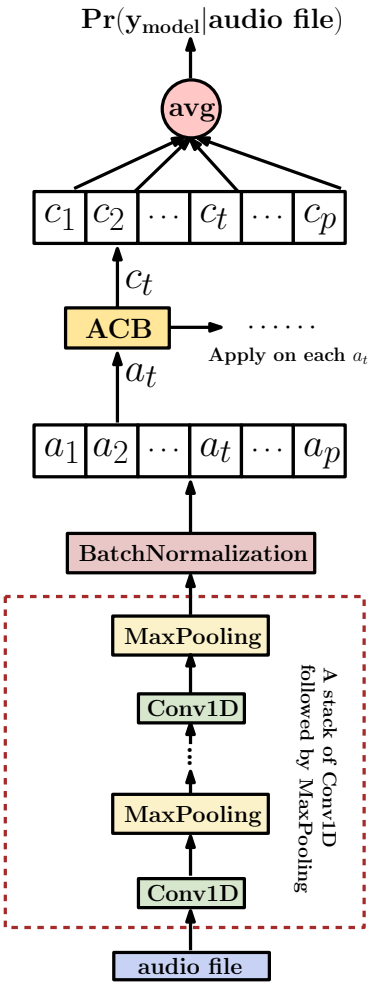

Figure 1: The complete architecture of CAB-CNN. The original audio file is first fed into a stack of CNNs and MaxPooling layers to get a "distilled" representation a. The batch normalization is applied at the end for making the model easier to train. For each time interval t, the ACB (the detailed design is presented in Fig 2) processes the representation of each a_t and outputs the probability of the classes c_t. Finally, the output class probability is the unweighted average of c_t.

As our feature extraction algorithm preserves the spatial information, we can list its output feature vectors in the order of time:

$$a = [a_1, a_2, \cdots, a_t, \cdots, a_p] \tag{1}$$

where a_t is the representation of the t-th time interval in the original data.

For each a_t in the list, the ACB outputs a weighted average of the probability vector generated by the classifiers.

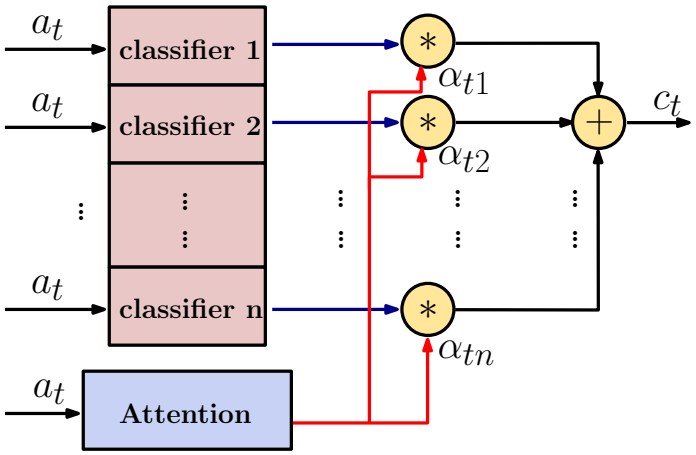

Figure 2: The attention-based classifier block (ACB) at time t. The block consists of an array of n classification unit and an attention mechanism. The block receives an representation (a_t) of the audio file at time t and then let each classifier implement an classification and the attention unit (attn) produce the importance weights α_{ti} for each classifier. The context vector c_t then equals to the sum of the weighted outputs of the classifiers.

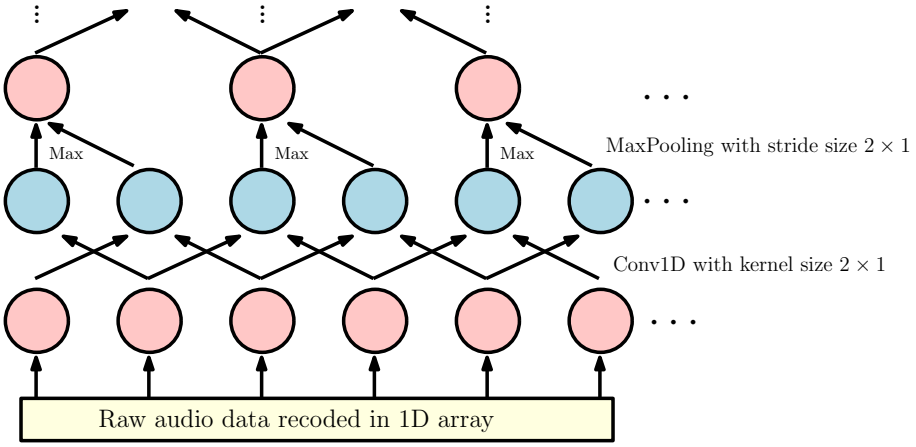

Figure 3: 1D-CNN followed by a MaxPooling layer. The kernel of 1D-CNN can extract features from the raw data with very high parameter efficiency. The MaxPooling layer can then distill the data fed by the lower layers and thus reduces the output volume. In our algorithm, we have applied a stack of such structure to recursively distill information. In this way, the features that are useful for the later classification task are preserved, while those irrelevant informations are removed.

In more details, suppose the ACB has n classifiers, and there are m classes to distinguish. Then, when receiving a_t, classifier i produces a classification probability vector $\mathbf{c}_{ti} \in \mathbb{R}^m$ for $i = 1, 2, \cdots n$ and the attention unit generates the importance weights

$$\alpha_t = [\alpha_{t1}, \alpha_{t2}, \cdots, \alpha_{tn}].$$

After this, the block outputs

$$c_t = \sum_{i=1}^{n} \alpha_{ti} \mathbf{c}_{ti}.$$

We can explain the design of the importance weights in two ways. First, they show how important a classifier is when classifying the audio at time t. Also, they represent the confidence that each classifier gives a correct output.

By using this attention mechanism, we relieve a single classifier from having to distinguish a large set of features related to an audio classification task. Instead, one classifier only needs to focus on a certain type of features.

In more details, assume that we need to classify some audios by the accents of the speakers. For a person, he/she may use the following strategy: 1) identify whether some certain features are present 2) if a feature is present and exclusively belongs to an accent, then we can say the audio is of this accent. Our algorithm works in a similar way. In particular, the attention unit identifies which features are present and lets the corresponding classifiers do the classification. This is done by assigning those classifiers with high importance weights. In this way, each classifier only needs to focus on a small subset of all available features, which thus can be trained easily and has a high predication accuracy.

By feeding a_t into the classifier ACB, we get a list of c_t representing the predicted probabilities of the classes at time t. At last, we simply take the unweighted average over all c_t to make the final predication. That is,

$$Pr(y_{\text{model}}) = \sum_{t \geq 1} c_t.$$

Notice that this unweighted average implies a prior: the features implying the label of the classes have the identical probability of being active in each time interval.

4. Experiments

In this section, we introduce our experimental mythology and the dataset for testing our new algorithm. We make some quantitative comparisons among the state-of-the art, a few popular neural network implementations and our new algorithm, CAB-CNN. The experimental results show that our new algorithm has a considerably better performance than the state-of-the-art. Moreover, we also provide more experimental results to show the behaviour of our algorithm.

We test our algorithm by performing accent classification task based on UT-Podcast corpus [11]. This corpus contains audios of three English accents: American (US), Australian (AU) and Great British (GB). In the original dataset, the distributions of three accents in the training and test data are significantly different. To fix this problem, we mix them and take 60.0%, 10.0% and 30.0% of samples for training, validation and test. We detail the allocation of samples in Table 1.

Table 1: The allocation of samples of UT-Podcast corpus for training, validation and test.

	US	**UK**	**AU**	**Total**
Training	387	347	458	1192
Validation	65	58	76	199
Test	194	174	230	598
Total	646	579	764	1989

Preprocessing of the audio file. We need to preprocess the audio files to reduce their sizes and filter out noises to facilitate the learning of the model. In particular, we first normalize the audio by subtracting the mean followed by dividing by the standard deviation. Then for every second of the audio, we partition the audio into $4,000$ sub-intervals and pick the maximum element of each sub-interval to produce the input array.

Model configurations. The detailed configurations of our model for testing are listed in Table 2. For the configurations of the ACB, we simply use multilayer perceptrons to implement the classifiers and the attention block. In particular, all the classifiers consist of two fully connected layers of eight and four neurons with the ReLU activation functions, followed by a softmax layer to classify three accents. For the attention block, it has two fully connected layers of size 160 and 80 and a softmax layer for producing importance weights for 40 classifiers. We also use the ReLU functions for introducing non-linearity.

Model training. We implement our model using the Keras library and train it from scratch by Adam optimizer [23] on NVIDIA GeForce GTX 1080 GPU. The loss is defined by the regular cross entropy function. We tested our model using at most the first T seconds of the audio files, where $T = 5$s, 10s, 30s and 60s (that is, if the audio length is less than T, we use the whole audio; otherwise, we only use the first T seconds). In the following discussion, we add time length after CAB-CNN to specify which implementation we are referring to. We set the batch size to 128. Since the Keras library requires that the input size of the neural network in a single batch must be the same, we truncate the sizes of the batch inputs just equal to the shortest one of that batch. We also apply the early stopping technique with patience 15 to fight against the overfitting problem.

Table 2: The layer configurations of CAB-CNN

Layer Name	**Parameters**	**Activation**
Conv1D	# filters: 16, size: 4×1, padding: same	ReLU
MaxPooling	size: 4×1, stride: 2×1, padding: same	Dropout(0.15)
Conv1D	# filters: 32, size: 4×1, padding: same	ReLU
MaxPooling	size: 4×1, stride: 2×1, padding: same	Dropout(0.15)
Conv1D	# filters: 32, size: 10×1, padding: same	ReLU
MaxPooling	size: 10×1, stride: 5×1, padding: same	Dropout(0.1)
Conv1D	# filters: 128, size: 10×1, padding: same	ReLU
MaxPooling	size: 10×1, stride: 5×1, padding: same	
BatchNormalization	N/A	N/A
ACB	40 Classifiers and 1 attention block	N/A
Average	N/A	N/A

Configurations of other algorithms for comparison. We demonstrate the performance of CAB-CNN by comparing it with some typical CNN architectures and the state-of-the-art algorithm. In particular, they are GatedResNet [22], Alexnet [24], VGG11$_A$ [12], ResNet18 [25] and AttentionCNN [10]. All these implementations need to preprocess the audios by converting them into spectrograms. In our experiments, we simply use the sizes suggested by the authors. For GatedResNet and Alexnet, the audios are converted to the graph of size 256×256, and for VGG11$_A$ and ResNet18, the spectrograms have size 224×224. All these graphs have three channels: red, green and blue. Regarding AttentionCNN, the sprectograms are in grayscale and of size 256×256. For all these models, we apply the original configurations presented in the papers except modifying the softmax layer for fitting our tasks. All these models are trained from scratch with the stochastic gradient descent (SGD) optimizer with learning rates 0.001, weight decay 10^{-8} and momentum 0.9. The batch size is set to 48. We also apply the early stopping technique with the patience equal to 15.

Table 3 lists the accuracy, unweighted recall, F1-Score of the selected algorithms by training and testing them on the remixed UT-Podcast corpus. The highest value of each score is bold. For each algorithm, we repeat the experiment for eight times and list the result having the highest accuracy. Since the CAB-CNN has the best performance when processing at most the first 30 seconds of the audio file, we only list the scores of CAB-CNN-30s here. We summarize the performances of CAB-CNN with other configurations in Table 4.

Suppose the test dataset contains N samples. Let $y_{data}^{(i)}$ denote the actual accent of the i-th sample in the test data and $y_{model}^{(i)}$ the predicted result generated

Table 3: The best result of each audio classification algorithm among eight parallel tests over the remixed UT-Podcast corpus.

Algorithm	**Accuracy**	**Recall$_{\text{unweighted}}$**	**F1-Score**
GatedResNet [22]	0.7939	0.7487	0.7557
Alexnet [24]	0.6793	0.6115	0.6152
VGG11$_A$ [12]	0.8645	0.8244	0.8331
ResNet18 [25]	0.8015	0.7424	0.7512
AttentionCNN [10]	0.8626	0.8257	0.8330
CAB-CNN-30s	**0.9599**	**0.9424**	**0.9523**

Table 4: Performances of the CAB-CNN using at most the first T seconds of the audios ($T = 5, 10, 30$ & 60).

Algorithm	**Accuracy**	**Recall$_{\text{unweighted}}$**	**F1-Score**
CAB-CNN-5s	0.9198	0.8953	0.9092
CAB-CNN-10s	0.9542	0.9383	0.9475
CAB-CNN-30s	**0.9599**	**0.9424**	**0.9523**
CAB-CNN-60s	0.9466	0.9228	0.9356

by the algorithms. Then the scores are calculated by

$$Accuracy = \frac{\sum_{i=1}^{N} \mathbf{1}_{\left(y_{data}^{(i)} = y_{model}^{(i)}\right)}}{N}.$$

Let L denote the set of the accents for classifying and T the test sample set. For accent $l \in L$, T_l denotes the samples having accent l in T. Let $|S|$ be the size of set S. Then we define

$$Recall_{unweighted} = \frac{1}{|L|} \sum_{l \in L} Recall(l),$$

where

$$Recall(l) = \frac{\sum_{i \in T_l} \mathbf{1}_{\left(y_{data}^{(i)} = y_{model}^{(i)}\right)}}{|T_l|}.$$

Let

$$Precision(l) = \frac{\sum_{i \in T_l} \mathbf{1}_{\left(y_{data}^{(i)} = y_{model}^{(i)}\right)}}{\sum_{i \in T} \mathbf{1}_{\left(y_{model}^{(i)} = l\right)}}.$$

Then

$$F1\text{-}Score = \frac{1}{|L|} \sum_{l \in L} F1\text{-}Score(l),$$

Table 5: The confusion matrix of the best results of VGG11$_A$ and AttentionCNN tested on the UT-Podcast dataset.

	VGG11$_A$			AttentionCNN		
Act. \ Pred.	**AU**	**US**	**UK**	**AU**	**US**	**UK**
AU	213	7	10	208	11	11
US	9	176	9	7	179	8
UK	24	12	64	22	13	65

Table 6: The confusion matrix of the CAB-CNN using at most the first T seconds of the audios ($T = 5, 10, 30$ & 60).

	CAB-CNN-5s			CAB-CNN-10s			CAB-CNN-30s			CAB-CNN-60s		
Act. \ Pred.	**AU**	**US**	**UK**	**AU**	**US**	**UK**	**AU**	**US**	**UK**	**AU**	**US**	**UK**
AU	219	9	2	222	8	0	226	4	0	224	5	1
US	9	185	0	1	192	1	2	191	1	3	191	0
UK	10	12	78	7	7	86	8	6	86	10	9	81

where

$$\textit{F1-Score}(l) = 2 \cdot \frac{\textit{Precision}(l) \cdot \textit{Recall}(l)}{\textit{Precision}(l) + \textit{Recall}(l)}.$$

From Table 3, we can observe that GAB-CNN-30s has a significant better result than other models. In particular, compared with the state-of-the-art algorithms VGG11$_A$ [12] and AttentionCNN [10], our algorithm has more than 10 percent improvement in accuracy, recall and F1-Score.

By comparing the test results of the GAB-CNN with various maximum length of input audio (see Table 4), we can observe that GAB-CNN has the best performance when the maximum length of the audio is set to be 30s. It is quite reasonable that the algorithm could suffer from a low performance if the input audio length is too short. In more details, a short audio may not contain enough features for accent classification. Consider the extreme case that the input audio only contains one word. Then if the pronunciation of this word is the same among all three accents, there is no way to classify it and the classification output will be randomly picked.

A decline of the performance can be also observed if the input audio is too long. This decline could be partially caused by the avg layer that summarizes the classification results in each time interval and outputs the predication in probability distribution (see Fig 1). As what we have mentioned in the previous paragraph, it is possible that some words do not contain information for the accent classification. When a long audio is given, this kind words may become prevalent. Then the random results corresponding to these words would weaken or even conceal the true results that are generated from those classifiable words.

Table 7: The average algorithm performances by implementing eight parallel tests over the remixed UT-Podcast corpus.

Metrics / Model	**Accuracy**	**Recall$_{\text{unweighted}}$**	**F1-Score**
GatedResNet	0.6594	0.5854	0.5686
Alexnet	0.5795	0.5590	0.4793
VGG11$_A$	0.6527	0.6227	0.6330
ResNet18	0.7950	0.7373	0.6908
AttentionCNN	0.7410	0.6012	0.4794
CAB-CNN-30s	**0.9370**	**0.9163**	**0.9282**

We list the confusions matrices of models VGG11$_A$, AttentionCNN and GAB-CNN with difference configurations in Table 5 and Table 6. We can obverse that the score improvements are contributed by all kinds of classification problems. For instance, from Table 5, we can see that both VGG11$_A$ and AttentionCNN models are likely to misclassify an UK accent as AU one, while GAB-CNN-30s can reduce this kind of error by two thirds.

Another highlight of our model is that the GAB-CNN is more robust. Specifically, while the other models require repeating the training processes for many times before finding an acceptable parameter solution, our model can always converge to a solution having good test scores. We justify this statement by averaging the scores of each algorithm over the eight parallel tests. We summarize the results in Table 7. From the table, we can observe that there is not a large gap between the best and the average performances of the GAB-CNN. In comparison, large drops can be observed for other algorithms. This observation implies that, the attention mechanism integrated in our model makes it easy for the model to be trained. As we have discussed in Section 3, the attention block relieves a classifier from having to identify all features that are different among various accents. Instead, it is enough for a classifier to only focus on a list of similar features. In this way, a classifier does not have to possess a large model capacity which requires a huge number of parameters. Therefore, both training and statistical efficiencies are improved, and the model is much easier to be trained.

5. Conclusion

In this paper, we have proposed an attention-based audio classification deep neural network named the Classifier-Attention-Based CNN (CAB-CNN). Unlike many of other DNN implementations in this area, our algorithm does not need to convert the audio files into spectrograms as a preprocessing step and thus avoids the unnecessary introduction of noises and makes jointly training on the whole model possible. Unlike Wu et al.'s attention-based model that attends to different frequencies and time intervals [10], our model instead uses the attention

block to select the proper classifiers to distinguish the input audios. This design makes our algorithm more robust and it has a significant better performance than all published neural network models in this area. Our work shows that an accent classification algorithm can gain a remarkable performance improvement by deploying a list of simple and specialized classifiers with an attention mechanism determining which classifier's result is more trustworthy and thus has a larger portion in the final predication.

We tested our model by performing accent classification tasks on the UT-Podcast corpus. Compared to the state-of-the-art algorithms [10, 12], our model has more than 10% improvements on all test scores and has reached 95.99% test accuracy.

6. Future work

Although, in this paper, we have presented an algorithm with a new architecture that has a descent improvement compared to the state of the art, its performance still has large room to be improved.

In our implementation, we simply use fully connected layers to implement the classifiers and the attention mechanism. We believe that some dedicated implementations can further improve its performance. Besides, as we have mentioned in Section 4, our algorithm suffers from a performance decline if the input audio length is too long. To fix this problem, we could design another learning algorithm to find the best length of input audio. Also, we could train an algorithm to locate the words that are accent classifiable and only input those words into our classification network.

References

[1] Z. Qawaqneh, A. A. Mallouh, B. D. Barkana, Deep neural network framework and transformed MFCCs for speaker's age and gender classification, Knowledge-Based Systems 115 (2017) 5–14.

[2] S. Poria, E. Cambria, A. Hussain, G.-B. Huang, Towards an intelligent framework for multimodal affective data analysis, Neural Networks 63 (2015) 104–116.

[3] C. Pedersen, J. Diederich, Accent classification using support vector machines, in: IEEE/ACIS International Conference on Computer and Information Science (ICIS), IEEE, 2007, pp. 444–449.

[4] P. Shegokar, P. Sircar, Continuous wavelet transform based speech emotion recognition, in: International Conference on Signal Processing and Communication Systems (ICSPCS), IEEE, 2016, pp. 1–8.

[5] Z. Yi, Foundations of implementing the competitive layer model by lotka–volterra recurrent neural networks, IEEE Transactions on Neural Networks 21 (3) (2010) 494–507.

[6] Z. Fu, G. Lu, K. M. Ting, D. Zhang, Optimizing cepstral features for audio classification, in: International Joint Conference on Artificial Intelligence, 2013.

[7] M. Espi, M. Fujimoto, K. Kinoshita, T. Nakatani, Exploiting spectro-temporal locality in deep learning based acoustic event detection, Journal on Audio, Speech, and Music Processing 2015 (1) (2015) 26.

[8] W. Lim, D. Jang, T. Lee, Speech emotion recognition using convolutional and recurrent neural networks, in: 2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA), IEEE, 2016, pp. 1–4.

[9] Y. Leng, C. Sun, X. Xu, Q. Yuan, S. Xing, H. Wan, J. Wang, D. Li, Employing unlabeled data to improve the classification performance of SVM, and its application in audio event classification, Knowledge-Based Systems 98 (2016) 117–129.

[10] Y. Wu, H. Mao, Z. Yi, Audio classification using attention-augmented convolutional neural network, Knowledge-Based Systems 161 (2018) 90–100.

[11] J. H. Hansen, G. Liu, Unsupervised accent classification for deep data fusion of accent and language information, Speech Communication 78 (2016) 19 – 33.

[12] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556.

[13] J. Pons, X. Serra, Randomly weighted cnns for (music) audio classification, in: ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019.

[14] S. Dieleman, B. Schrauwen, End-to-end learning for music audio, in: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2014, pp. 6964–6968.

[15] J. Pons, X. Serra, Designing efficient architectures for modeling temporal features with convolutional neural networks, in: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017, pp. 2472–2476.

[16] J. Lee, J. Park, K. L. Kim, J. Nam, Samplecnn: End-to-end deep convolutional neural networks using very small filters for music classification, Applied Sciences (Switzerland) 8 (1) (2018) 1–14.

[17] K. Choi, G. Fazekas, M. Sandler, Automatic tagging using deep convolutional neural networks, Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016 (2016) 805–811.

[18] R. Caruana, Multitask learning, Machine Learning 28 (1) (1997) 41–75.

[19] J. Baxter, Learning internal representations, Proceedings of the 8th Annual Conference on Computational Learning Theory, COLT 1995 (1995) 311–320.

[20] B. Zhang, G. Essl, E. M. Provost, Recognizing emotion from singing and speaking using shared models, in: International Conference on Affective Computing and Intelligent Interaction (ACII), IEEE, 2015, pp. 139–145.

[21] H. Lee, P. Pham, Y. Largman, A. Y. Ng, Unsupervised feature learning for audio classification using convolutional deep belief networks, in: Advances in Neural Information Processing Systems, 2009, pp. 1096–1104.

[22] Y. Zeng, H. Mao, D. Peng, Z. Yi, Spectrogram based multi-task audio classification, Multimedia Tools and Applications 78 (3) (2019) 3705–3722.

[23] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: The International Conference on Learning Representations (ICLR), San Diego, USA, 2015.

[24] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.

[25] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.