INTERSPEECH 2020
October 25–29, 2020, Shanghai, China

Deep Embedding Learning for Text-Dependent Speaker Verification

Peng Zhang[1], *Peng Hu*[2], *Xueliang Zhang*[1]

[1]College of Computer Science, Inner Mongolia University, Hohhot, China
[2]Elevoc Technology Co., Ltd, Shenzhen, China

zhangpeng@mail.imu.edu.cn, peng.hu@elevoc.com, cszxl@imu.edu.cn

Abstract

In this paper we present an effective deep embedding learning architecture for speaker verification task. Compared with the widely used residual neural network (ResNet) and time-delay neural network (TDNN) based architectures, two main improvements are proposed: 1) We use densely connected convolutional network (DenseNet) to encode the short term context information of the speaker. 2) A bidirectional attentive pooling strategy is proposed to further model the long term temporal context and aggregate the important frames which reflect the speaker identity. We evaluate the proposed architecture on the task of text-dependent speaker verification in the Interspeech 2020 Far Field Speaker Verification Challenge (FFSVC2020). Result shows that the proposed algorithm outperforms the official baseline of FFSVC2020 with 8.06%, 19.70% minDCFs and 9.26%, 16.16% EERs relative reductions on the evaluation set of Task 1 and Task 3 respectively.

Index Terms: speaker verification, dense connections, bidirectional attentive pooling

1. Introduction

The goal of speaker verification (SV) is to verify the speaker identity associated with the enrolled target speaker from the digital audio signal level [1]. For different applications, two categories can be defined for speaker verification, namely the text-dependent speaker verification (TD-SV) and text-independent speaker verification (TI-SV). In a TD-SV system, transcripts of enrollment are constrained to a specific phrase, which is not the case in TI-SV systems. Because of the constraint of the phonetic variability, TD-SV systems usually achieve robust verification results with very short enrollment utterances. With intelligent speech assistants such as Alexa, Google Home, Siri and Cortana being used in smart speakers and smart phones, human–machine interactions through voice command are becoming widespread. A common application requirement is to wake up the smart device via a specific keyword by a certain speaker for personalized service and unauthorized usage prevention. So, the TD-SV is essential in this scenario.

In speaker recognition field, i-vectors [2] with Probabilistic Linear Discriminative Analysis (PLDA) as backend [3] is the most popular method. The standard approach consists of a universal background model (UBM), and a large projection matrix **T** that are learned in an unsupervised way to maximize the data likelihood. The projection maps high-dimensional statistics from the UBM into a low-dimensional representation.

Recently, with the development of deep neural networks (DNN) and inspired by the great success of incorporating deep neural networks into speech recognition, researchers in the speaker recognition community also investigated the application of DNN for speaker modeling. More attention has been drawn to the use of DNNs to generate discriminative speaker embedding representations [4, 5, 6, 7, 8, 9, 10], i.e. time-delay neural network (TDNN) [6, 9], convolutional neural network (CNN) [4, 11], or long short-term memory network (LSTM) [5]. Deep learning based methods have been dominating to obtain utterance-level representation aka deep speaker embedding or embedding for short, which contains essential information to discriminate among speakers. Pooling layer is an essential component to capture speaker characteristic for speaker recognition systems, average pooling, max pooling [8], statistic pooling [7], and attentive pooling [10] are popular choices.

Comparing with traditional i-vector systems, deep embedding learning benefits from the discriminative ability of DNNs, and the span of acoustic features for exploiting time-frequency context information. It has been demonstrated that the effectiveness of embedding learning can be improved by exploiting information from multiple layers [11], or by introducing an attention mechanism [10, 12]. Okabe et al. [13] proposed a parametric based attentive pooling to give different weights for each frame feature. Zhu et al. [10] further extended it to multiple head to generate utterance representation from different views. Tang et al. [14] proposed a pooling strategy which collects speaker information from both TDNN and LSTM layers. However, these methods mainly focus on local feature aggregation not temporal context information.

In this paper, we propose an effective deep embedding learning architecture for speaker verification. Motivated by the recent success of densely connected convolutional network (DenseNet) in image classification [15], music source separation [16], speaker separation [17] and speaker recognition [18], we employ the DenseNet as frame-level feature extractor which concatenates feature-maps learned by different layers to increase variation in the input of subsequent layers and also training efficiency. Conceptually, each dense block acts as a small CNN system. The difference is that the outputs of layers are densely connected in a feed-forward manner, implemented by concatenation of outputs in the channel dimension. Furthermore, we design a bidirectional attentive pooling layer for further modeling the temporal context to improve the representational power of utterance-level features. We combine the bidirectional gated recurrent unit (BGRU) layer and attentive pooling into one unified architecture. The BGRU layer model long range, context information, then using the attentive pooling to compute attentive weight for better temporal aggregation over the bidirectional encoded sequences. We evaluate the deep embedding learning architecture on FFSVC2020 dataset. Our proposed model significantly outperforms the official baseline with 8.06%, 19.70% minDCFs and 9.26%, 16.16% relative reductions on the evaluation set of Task 1 and Task 3 respectively.

The remainder of this paper is organized as follows: Section 2 introduces the proposed method. The experimental setup is presented in Section 3. The results are presented and discussed in Section 4. Finally, we conclude the whole work in Section 5.

http://dx.doi.org/10.21437/Interspeech.2020-1354

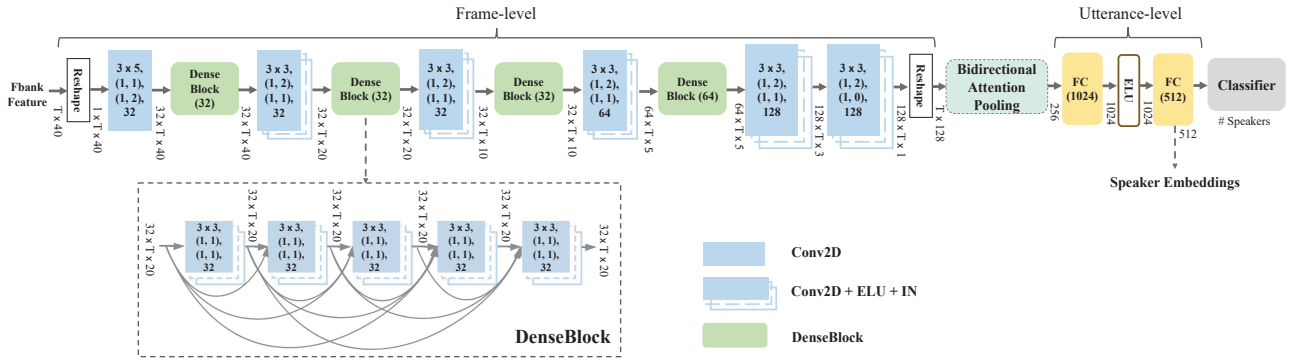

Figure 1: *A schematic of the proposed deep architecture. For the frame-level stage, each DenseBlock is made up of 5 convolution layers (Conv2D), exponential linear units (ELU) and instance normalizations (IN). The tensor shape after each DenseBlock is in the format: featureMaps × timeSteps × frequencyChannels. Each Conv2D and Conv2D+IN+ELU is specified in the format: kernelSizeTime × kernelSizeFreq, (stridesTime, stridesFreq), (paddingsTime, paddingsFreq), featureMaps. Each DenseBlock(g) contains five Conv2D+IN+ELU blocks with growth rate g. For the utterance-level stage, numbers denote the channel of output feature maps or embedding dimensions in our implementaion.*

2. Methods

Our proposed deep embedding learning architecture as shown in Figure 1, which consists of three parts. Firstly, we introduce DenseNet as the frame-level feature extractor, which includes four dense blocks (DenseBlock) that each has five CNN layers. After frame-level feature extraction, the bidirectional attentive pooling layer is used to convert frame-level features into fixed-dimensional vectors, and followed by two fully-connected hidden layers to form utterance-level features. The output is a softmax classifier layer, and each node corresponds to a speaker ID.

2.1. DenseBlock

The main idea of the DenseBlock architecture first presented in [15] is to introduce direct connection from each layer to all subsequent layers in a network building block of a CNN, which designed to capture long time-frequency context information in an efficient way. For a DenseBlock as shown in the dashed box in Figure 1, $H_l(\cdot)$ denotes a general transformation of the l-th layer. $\mathbf{x}_1$, $\mathbf{x}_2$, ... , $\mathbf{x}_{l-1}$ are the outputs of the preceding layers, and $\mathbf{x}_0$ is the input of this DenseBlock. All convolutional layers in the same DenseBlock can be designed to output feature maps with matching time and frequency dimensions. That is, $\mathbf{x}_l \in R^{C\times T\times F}$, where C, T and F are channel, time and frequency dimensions of the output feature maps. The general convolution operation for the layer l is $\mathbf{x}_l = H_l(\mathbf{x}_{l-1})$, but here it acts on the concatenation of $\mathbf{x}_1$, $\mathbf{x}_2$, ... , $\mathbf{x}_{l-1}$ as

$$\mathbf{x}_l = H_l([\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{l-1}]), \quad (1)$$

where $[\cdot]$ is concatenation in the channel dimension. The l-th convolutional layer takes feature maps of size $C \times (l-1) + C_0$ as input, where C_0 is the channel dimension of feature maps corresponding to the DenseBlock's input $\mathbf{x}_0$. As can be observed, each layer is connected to all following layers directly via feature map concatenation. Such a connectivity pattern is designed to yield a better gradient flow between layers during training and to capture temporal context information by giving each layer access to all feature representations of earlier layers.

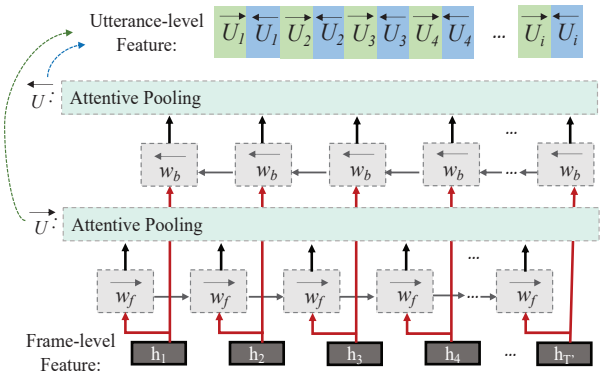

Figure 2: *A schematic of the bidirectional attentive pooling (BAP). $\boldsymbol{h}_i$ stands for i-th vector of the BAP input, $\boldsymbol{w}_f$ and $\boldsymbol{w}_b$ stands for the forward and backward hidden state of BGRU, respectively. $\overrightarrow{U}$ and $\overleftarrow{U}$ are the bidirectional outputs from the BGRU layers.*

2.2. Bidirectional Attentive Pooling

After obtaining the frame-level features, an aggregation algorithm is needed to combine these representations into utterance wise speaker embedding. A simple temporal average pooling is generally shown effective. However, not all frame-level representations provide equal clues to infer speaker identity. Instead of averaging, the attention mechanism [10, 12] is a better alternative to actively select the hidden representations and emphasize speaker discriminative informations.

Suppose a speech segment of duration T produces a sequence of T outputs vectors, $\{\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_{T'}\}$. The attentive pooling method can be formulated as follows:

$$\alpha_t = softmax(\mathbf{v}^T g(\mathbf{W}^T \mathbf{h}_t + \mathbf{b})), \quad (2)$$

$$U = \sum_{t=1}^{T'} \alpha_t \mathbf{h}_t, \quad (3)$$

where we project $\mathbf{h}_t$ into a subspace through a fully connected

layer with parameters **W** and **b**, measuring its similarity with **v**, and get a normalized attention weight α_t using softmax. $g(\cdot)$ is some activation function and Tanh is chosen here. The $softmax(\cdot)$ is performed column-wise.

To get more discriminative fixed-dimensional utterance-level representation and capture long term sequence information, Cai et al. [19] proposed an attention-based CNN-BLSTM framework, which combine CNN-BLSTM model with a attentive pooling layer together. Different from [19] directly connecting the BLSTM to the attentive pooling layer, we employ the attentive pooling to capture the bidirectional temporal information output by the bidirectional recurrent neural network, and then concatenate the bidirectional utterance-level features. The proposed pooling method called bidirectional attentive pooling (BAP), which can be expressed as:

$$\overrightarrow{U} = AttendPool(\overrightarrow{H}), \tag{4}$$

$$\overleftarrow{U} = AttendPool(\overleftarrow{H}), \tag{5}$$

where $AttendPool(\cdot)$ is the operation of the attentive pooling, $\overrightarrow{H}$, $\overleftarrow{H}$ are the bidirectional outputs from the BLTSM/BGRU layers. Then the bidirectional utterance-level features $\overrightarrow{U}$ and $\overleftarrow{U}$ are cross combined to form an utterance-level speaker embedding as this following:

$$U = CrossCat(\overrightarrow{U}, \overleftarrow{U}), \tag{6}$$

where the $CrossCat(\cdot)$ operation is the representations of each dimension of $\overrightarrow{U}$ and $\overleftarrow{U}$ are cross concatenated, not directly stitched. As we can be observed, the BAP layer takes advantage of both bidirectional sequential modeling and attention mechanism to capture long-term temporal context informations.

3. Experiments

3.1. Datasets

FFSVC2020 has three tasks in a research competition under well-defined conditions: 1) far-field TD-SV from a single microphone array; 2) far-field TI-SV from a single microphone array; 3) far-field TD-SV from distributed microphone arrays. We evaluate the proposed architecture on the TD-SV tasks (Task 1 and 3). Audio clips in FFSVC2020 are recorded by close-talking microphone, a mobile phone placed at 25cm distance from speaker and three circular microphone arrays simultaneously. The first 30 utterances are of fixed content: 'ni hao mi ya' in Mandarin Chinese for TD-SV tasks. The remaining utterances are text-independent. We trained our DNN on the first 30 utterances of FFSVC 2020 training dataset, and SLR85 HI-MIA dataset. The SLR 85 HI-MIA from `openslr.org` is a subset of the original HI-MIA database [20]. It contains one close-talking microphone and three microphone arrays located at 1m, 3m and 5m distance right in front of the speaker. In total, training data sets have nearly 1,139,671 utterances and the total duration approximately 950 hours with 374 speakers, there are 120 speakers from the FFSVC2020 dataset, and 254 speakers from the HI-MIA dataset.

3.2. Implementation Details

- **Input features.** All audio clips are first downsampled to 16kHz. Mean normalization is applied using a moving window of 3 seconds speech segment. For each segment, 40-dimensional Mel-filter bank features are generated with a 20-ms window length and 10-ms offset.

- **Online Data Augmentation.** We use the Online data augmentation strategy in [21] to make the speaker embeddings more robust. The public MUSAN [22] and RIR_NOISES [23] are used as interfering noises. For each audio segment, we randomly select a noise clip and mix it with the audio segment at a signal-to-noise ratio (SNR) level. The SNR is uniformly distributed between 0 and 20 dB.

- **Loss function.** We use additive margin softmax, proposed in [24], as the loss function, which has good performance and ease of implementation corresponding loss function is

$$L = -\frac{1}{n}\sum_{i=1}^{n} log\frac{e^{scos(\theta_{y_i}-m)}}{e^{scos(\theta_{y_i}-m)} + \sum_{j\neq y_i} e^{scos\theta_j}}, \tag{7}$$

where $cos\theta_{y_i} = \mathbf{w}_{y_i}^T \mathbf{f}_i / \parallel \mathbf{w}_{y_i} \parallel \parallel \mathbf{f}_i \parallel$, $\mathbf{w}_{y_i}$ is the weight vector of class y_i, and $\mathbf{f}_i$ is the input to the layer for example i. Also, s is an adjustable scale factor and m is the penalty margin. To accelerate convergence, we follow the practice of fixing s to a predefined value. Also, during training, we set $s = 10$, and m is initially set to 0 and then to 0.35 after one epoch.

- **Training and testing.** The configuration of the neural network is illustrated in Figure 1. Models are trained using PyTorch [25] and using the Adam optimizer with a batch size of 128. After some exploration, we observe that 100K training steps are sufficient to train the networks. We also evaluate different learning rate schedulers. The selected strategy uses a starting learning rate of 0.1, keeps it constant for 30K steps, and then it applies an exponential decay every 10K steps with a rate of 1/2. The period of constant learning rate was important for the networks trained with margin penalty. For testing, one microphone array is used in Task 1; 2-4 microphone arrays are randomly selected in Task 3. Different channels from the microphone array(s) are equally weighted at the embedding level before scoring. The performance metrics are equal error rate (EER) and minimum detection cost function (minDCF) with P_{target} = 0.01, and the minDCF as the primary metric.

- **Scoring.** We used both PLDA [2, 3] and cosine for scoring. In the PLDA scoring, the PLDA of the system is trained using embeddings of the whole training set. The post-processing of the speaker embeddings are extracted from the embedding layers, length normalization, centering, whitening and LDA transformation for feature dimensionality reduction has been applied to the embeddings in sequence, finally followed by the PLDA training. Specifically, the PLDA scoring process is based on the Kaldi toolkit [9]. In the cosine scoring, we simply evaluate the pair-wise comparisons using the cosine distance.

4. Results

We compare our algorithm with x-vector system [9] and FFSVC2020 official baseline system [26]. The results are shown in Table 1. Since the maximum number of evaluation set submissions is only five, we list part of the score submission results on the evaluation set. It should be mentioned that the system 1 is actually the same as the x-vector system in Kaldi

Table 1: *Performance of different systems,* ***Boldface*** *values are the best results each in PLDA backend and cosine distance. DenseNet system as described in Section 2.1 and shown in Figure 1. BAP pooling method is described in Section 2.2.*

System	Feature extractor	Pooling	Scoring	Development Set				Evaluation Set			
				Task 1		Task 3		Task 1		Task3	
				minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)
1 (x-vector [9])	TDNN	SP	PLDA	0.60	6.04	0.63	5.47	-	-	-	-
			cosine	0.59	6.21	0.61	5.63	-	-	-	-
2	TDNN	BAP	PLDA	**0.58**	**5.95**	0.59	**5.35**	0.60	6.06	0.63	6.21
			cosine	0.59	6.07	**0.57**	5.56	-	-	-	-
3 (baseline [26])	ResNet	SP	PLDA	0.56	5.87	0.59	5.24	-	-	-	-
			cosine	0.55	5.89	0.57	5.19	-	-	-	-
4	ResNet	BAP	PLDA	0.53	5.08	0.51	**4.59**	0.59	6.14	0.58	6.18
			cosine	**0.52**	**4.97**	**0.49**	4.67	-	-	-	-
5	DenseNet	SP	PLDA	0.54	5.39	0.55	5.08	-	-	-	-
			cosine	0.55	5.27	0.53	4.98	-	-	-	-
6 (proposed)	DenseNet	BAP	PLDA	0.49	**4.57**	**0.45**	**4.12**	**0.57**	**5.78**	**0.53**	**6.02**
			cosine	**0.48**	4.60	0.46	4.27	-	-	-	-
FFSVC2020 Baseline [26]	ResNet	SP	cosine	0.57	6.01	0.59	5.42	0.62	6.37	0.66	7.18
Fusion(2+4+6)	-	-	-	**0.43**	**3.67**	**0.41**	**3.98**	**0.52**	**4.72**	**0.47**	**5.14**

Table 2: *EERs and minDCFs of the DenseNet-BAP models with respect to the number of BLSTM/BGRU layers and the layer size,* ***Boldface*** *value is the best result.*

Recurrent Unit	Layers	Layer Size	minDCF	EER(%)
0	0	0	0.54	5.08
BLSTM	1	128	**0.51**	**4.83**
	2	128	0.53	4.89
	3	128	0.55	4.93
	1	256	0.52	4.85
BGRU	1	128	0.52	4.63
	2	128	**0.49**	**4.57**
	3	128	0.53	4.66
	2	256	0.50	4.61

[27], and system 3 is the re-implementation of the official baseline system [26]. System 6 is our proposed method. Comparing systems 1, 3, and 6, we observe that our proposed method outperforms the x-vector and the official baseline on the development set. Comparing with the FFSVC2020 baseline implementation results, our proposed method is significantly outperform the FFSVC2020 baseline system and get the best performance in PLDA scoring with 8.06%, 19.70% minDCFs and 9.26%, 16.16% EERs relative reductions on the evaluation set of Task 1 and Task 3 respectively.

DenseNet and BAP method are the most important components in our proposed architecture. To further evaluate the performance, as showed in Table 1, we replace the DenseNet with TDNN and ResNet, and also replace the pooling method from BAP to statistic pooling (SP). Comparing the systems 1, 3, 5, we observe that DenseNet performs better than TDNN and ResNet. The main reason is that the DenseNet combines lots of different aspects and scales of temporal information. These results suggest that DenseNet is an efficient architecture for SV tasks. Comparing systems 2, 4, and 6, we observe that BAP outperforms SP under all three feature extractors. We further explore the impact of bidirectional recurrent unit architecture as shown in Table 2. It can yield a decent performance when the BAP layer consists of 2 BGRU layers with 128 cells in each direction.

For the best system, we use simple averaging method to fuse the systems 2, 4 and 6 with PLDA as the backend, and submit the average-fused system to the mid-term leaderboard of FFSVC2020. Our fused system improves the minDCFs of 16.13%, 28.79% and the EERs of 25.90%, 28.41% over the FFSVC2020 baseline, respectively in Task 1 and Task 3.

5. Conclusions

In this paper, we propose a deep embedding learning architecture for text-dependent speaker verification. The architecture consists of a stack of DenseBlocks to capture the speaker identity at frame level and a bidirectional attentive pooling structure to form speaker embedding at utterance level. By densely concatenating the outputs of convolutional layers, a more meaningful frame-level representation, with various aspects of time-frequency context information, is generated. The bidirectional attentive pooling layer further capture temporal context information from the bidirectional with the combination of the BGRU layer and attentive pooling.

We conduct extensive experiments on the FFSVC2020 challenge. For our score submission in the FFSVC2020, our proposed method achieves 0.52, 4.72% and 0.47, 5.14% in the minDCFs and EERs for Task 1 and Task 3 on the evaluation set, respectively. Also, the results demonstrate significant performance gains over x-vector and ResNet baseline systems.

6. Acknowlegement

We gratefully acknowledge many fruitful discussions with all the members from Elevoc's Speech team and Dr. Zhong-Qiu Wang. This research work is supported by the National Natural Science Foundation of China (No. 61876214).

7. References

[1] J. H. Hansen and T. Hasan, "Speaker recognition by machines and humans: A tutorial review," *IEEE Signal Processing Magazine*, vol. 32, no. 6, pp. 74–99, 2015.

[2] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2010.

[3] P. Kenny, "Bayesian speaker verification with heavy-tailed priors." in *Odyssey*, 2010, p. 14.

[4] P. Kenny, T. Stafylakis, P. Ouellet, V. Gupta, and M. J. Alam, "Deep neural networks for extracting baum-welch statistics for speaker recognition." in *Odyssey*, 2014, pp. 293–298.

[5] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer, "End-to-end text-dependent speaker verification," in *ICASSP*, 2016, pp. 5115–5119.

[6] D. Snyder, P. Ghahremani, D. Povey, D. Garcia-Romero, Y. Carmiel, and S. Khudanpur, "Deep neural network-based speaker embeddings for end-to-end speaker verification," in *2016 IEEE Spoken Language Technology Workshop*. IEEE, 2016, pp. 165–170.

[7] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep neural network embeddings for text-independent speaker verification." in *Interspeech*, 2017, pp. 99–1003.

[8] S. Novoselov, A. Shulipa, I. Kremnev, A. Kozlov, and V. Shchemelinin, "On deep speaker embeddings for text-independent speaker recognition," in *Odyssey*, 2018, pp. 378–385.

[9] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *ICASSP*, 2018, pp. 5329–5333.

[10] Y. Zhu, T. Ko, D. Snyder, B. Mak, and D. Povey, "Self-attentive speaker embeddings for text-independent speaker verification." in *Interspeech*, 2018, pp. 3573–3577.

[11] C. Li, X. Ma, B. Jiang, X. Li, X. Zhang, X. Liu, Y. Cao, A. Kannan, and Z. Zhu, "Deep speaker: an end-to-end neural speaker embedding system," *arXiv preprint arXiv:1705.02304*, 2017.

[12] F. R. rahman Chowdhury, Q. Wang, I. L. Moreno, and L. Wan, "Attention-based models for text-dependent speaker verification," in *ICASSP*, 2018, pp. 5359–5363.

[13] K. Okabe, T. Koshinaka, and K. Shinoda, "Attentive statistics pooling for deep speaker embedding," in *Interspeech*, 2018, pp. 2252–2256.

[14] Y. Tang, G. Ding, J. Huang, X. He, and B. Zhou, "Deep speaker embedding learning with multi-level pooling for text-independent speaker verification," in *ICASSP*, 2019, pp. 6116–6120.

[15] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *CVPR*, 2017, pp. 4700–4708.

[16] N. Takahashi and Y. Mitsufuji, "Multi-scale multi-band densenets for audio source separation," in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE, 2017, pp. 21–25.

[17] Y. Liu and D. Wang, "Divide and conquer: A deep casa approach to talker-independent monaural speaker separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 12, pp. 2092–2102, 2019.

[18] Y. Jiang, Y. Song, I. McLoughlin, Z. Gao, and L. Dai, "An effective deep embedding learning architecture for speaker verification," in *Interspeech*, 2019, pp. 4040–4044.

[19] W. Cai, D. Cai, S. Huang, and M. Li, "Utterance-level end-to-end language identification using attention-based cnn-blstm," in *ICASSP*, 2019, pp. 5991–5995.

[20] X. Qin, H. Bu, and M. Li, "Hi-mia: A far-field text-dependent speaker verification database and the baselines," in *ICASSP*, 2020, pp. 7609–7613.

[21] D. Cai, W. Cai, and M. Li, "Within-sample variability-invariant loss for robust speaker recognition under noisy environments," in *ICASSP*, 2020, pp. 6469–6473.

[22] D. Snyder, G. Chen, and D. Povey, "Musan: A music, speech, and noise corpus," *arXiv preprint arXiv:1510.08484*, 2015.

[23] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *ICASSP*, 2017, pp. 5220–5224.

[24] F. Wang, J. Cheng, W. Liu, and H. Liu, "Additive margin softmax for face verification," *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 926–930, 2018.

[25] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[26] X. Qin, M. Li, H. Bu, W. Rao, R. K. Das, S. Narayanan, and H. Li, "The interspeech 2020 far-field speaker verification challenge," *arXiv preprint arXiv:2005.08046*, 2020.

[27] D. Povey, A. Ghoshal, and G. Boulianne, "The kaldi speech recognition toolkit," in *2011 IEEE Workshop on Automatic Speech Recognition and Understanding*, 2011.