INTERSPEECH 2019
September 15–19, 2019, Graz, Austria

An Effective Deep Embedding Learning Architecture for Speaker Verification

Yiheng Jiang[1], *Yan Song*[1], *Ian McLoughlin*[2], *Zhifu Gao*[1], *Lirong Dai*[1]

[1]National Engineering Laboratory of Speech and Language Information Processing, University of Science and Technology of China, Hefei, China.
[2]School of Computing, University of Kent, Medway, UK.

{jiangyh,gaozf}@mail.ustc.edu.cn, {songy,lrdai}@ustc.edu.cn, ivm@kent.ac.uk

Abstract

In this paper we present an effective deep embedding learning architecture, which combines a dense connection of dilated convolutional layers with a gating mechanism, for speaker verification (SV) tasks. Compared with the widely used time-delay neural network (TDNN) based architecture, two main improvements are proposed: (1) The dilated filters are designed to effectively capture time-frequency context information, then the convolutional layer outputs are utilized for effective embedding learning. Specifically, we employ the idea of the successful DenseNet to collect the context information by dense connections from each layer to every other layer in a feed-forward fashion. (2) A gating mechanism is further introduced to provide channel-wise attention by exploiting inter-dependencies across channels. Motivated by squeeze-and-excitation networks (SENet), the global time-frequency information is utilized for this feature calibration. To evaluate the proposed network architecture, we conduct extensive experiments on noisy and unconstrained SV tasks, *i.e.*, Speaker in the Wild (SITW) and Voxceleb1. The results demonstrate state-of-the-art SV performance. Specifically, our proposed method reduces equal error rate (EER) from TDNN based method by 25% and 27% for SITW and Voxceleb1, respectively.

Index Terms: speaker verification, dilated convolution, dense connections, gating mechanism

1. Introduction

Speaker verification (SV) is the task of automatically determining whether a given speech utterance belongs to a specific speaker or not. SV systems typically consists of two main stages: (1) frontend embedding learning which extracts a low-dimensional speaker embedding, and (2) backend modeling to calculate the similarity between speaker embeddings.

For the past decade, most popular SV methods have been based on i-vectors with a Probabilistic Linear Discriminative Analysis (PLDA) backend [1, 2]. In these systems, the i-vector is learned via a pipeline of generative modeling which first trains a Gaussian Mixture Model-Universal Background Model (GMM-UBM) to collect sufficient statistics, and then trains a large loading matrix to project the high-dimensional supervector from collected sufficient statistics into a low-dimensional total variability space.

Recently, more attention has been given to deep speaker embedding learning methods [3, 4, 5, 6, 7, 8, 9]. Early systems use deep neural networks (DNN) that are trained as the acoustic models for automatic speech recognition (ASR) to enhance the modeling of the i-vectors: either replacing the GMM-UBM to collect sufficient statistics [3], or acting as a frame-level feature extractor [4]. By exploiting additional transcribed information from the in-domain ASR corpus, these methods have demonstrated superior SV performance.

More recently, end-to-end SV methods have been proposed [5, 6, 7, 8, 9], in which embeddings are extracted using DNNs that are trained to directly discriminate between speakers. These methods first deal with a local short span of acoustic features to obtain more effective frame-level representations. It can be done via several layers in a time-delay neural network (TDNN) [5, 6], convolutional neural network (CNN) [7], or Long Short-Term Memory Network (LSTM) [8]. A pooling layer follows to aggregate frame-level outputs, and fully-connected (FC) layers then map the aggregation to speaker embeddings. Average-pooling, max-pooling [10], statistics pooling [6], attentive pooling [11], and cross-layer bilinear pooling [12] are popular choices.

Comparing with traditional i-vector systems, deep embedding learning may enjoy benefits from both the discriminative perspective of DNNs, and the span of acoustic features for exploiting time-frequency context information. It has been shown that the effectiveness of embedding learning can be improved by exploiting information from multiple layers [12, 13], or by introducing an attention mechanism [11]. Tang *et al.* [13] proposed a pooling strategy which collects speaker information from both TDNN and LSTM layers. Bilinear pooling of two consecutive layers [12] has also been proposed. In [11], an attentive pooling mechanism was employed to capture long-term variation in speaker characteristics by providing different frame weights. However, these methods mainly focus on local feature aggregation.

This paper advances in an orthogonal direction by proposing an effective architecture for SV, as shown in Fig. 1 and detailed in Section 2. Specifically, we focus on improving the effectiveness of frame-level features by designing an architecture consisting of dilated dense blocks (DDB), gate blocks and transition layers, where the DDB functions similarly to a TDNN based system. The dilated filters are designed for convolutional layers to cover local features of different spans. The outputs of convolutional layers are then concatenated, leading to more meaningful frame-level features with various time-frequency context information. Furthermore, a gating mechanism is proposed to introduce a channel-wise attention. It is worth noting that incorporating the pooling operations of methods like [11, 12, 13] into our architecture is straightforward.

The proposed architecture is evaluated on two benchmark datasets, Speaker in the Wild (SITW) [14] and Voxceleb1 [15]. Results show relative improvements over conventional methods of 25% and 27% in these two datasets, respectively.

http://dx.doi.org/10.21437/Interspeech.2019-1606

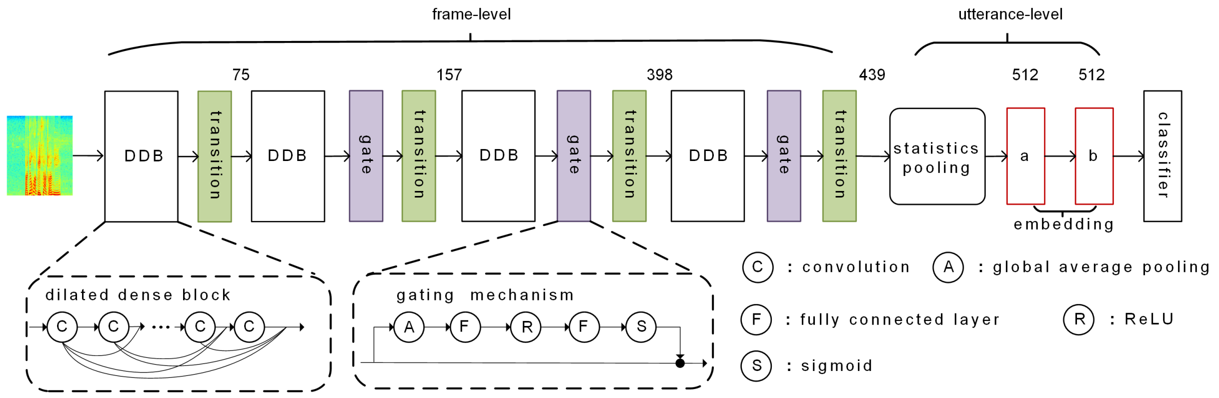

Figure 1: *The proposed deep architecture comprises three basic components: (1) Dilated dense blocks (DDB), each one is made up of convolution layers, batch normalizations (BN) and rectified linear units (ReLU). (2) Gate blocks, each one provides a gating mechanism for channel-wise attention, and (3) Transition layers, each one comprises convolution, BN and ReLU to limit the number of feature maps. Note: numbers denote the channels of output feature maps or embedding dimensions in our implementation.*

2. Overview of deep embedding learning architecture

In this section, we briefly introduce the widely used TDNN based deep embedding learning architecture, *i.e.*, x-vector system [6]. and then describe our proposed method based on it.

The x-vector can be roughly divided into frame-level and utterance-level parts. The frame-level part consists of 5 TDNN layers which deal with a local short span of acoustic features, while the utterance-level part computes the speaker embeddings via statistics pooling and FC layers. The x-vector is trained to discriminate between speakers via a cross-entropy (CE) criterion. For SV tasks, the embeddings from FC layers can be used for backend modeling such as PLDA to calculate a similarity score.

The architecture proposed in this paper aims to exploit the outputs of multiple layers to improve the effectiveness of the frame-level representation under an x-vector framework, other operations at utterance-level are the same as for the x-vector system. Since there exists a gradual transition from low to high layers in a DNN [16], exploiting information from multiple layers may be beneficial for an effective representation. A straightforward extension would be to concatenate the outputs of different TDNN layers. However, this would drastically increase model size and computational complexity, unless the TDNN layers are very slim – but such a slim network may be unable to effectively represent specific speaker information.

Inspired by the recent DenseNet [17], we propose an architecture consisting of dilated dense blocks (DDB), gate blocks and transition layers, shown in Fig 1. The dilated filter used in the DDBs is motivated from WaveNet [18], and it is designed to capture long time-frequency context information in an efficient way. Conceptually, each DDB acts as a small TDNN system. The difference is that the outputs from layers are densely connected in a feed-forward manner, implemented by concatenation of outputs in the channel dimension. Here, channel is defined as the number of feature maps in each convolutional layer. This allows the resulting network to enjoy the benefits of feature reuse from preceding layers without increasing number of parameters, and alleviate the vanishing-gradient problem.

A gate block is introduced which further enhances the representational power of DDB outputs. It could exploit global time-frequency information to increase the sensitivity of informative features while suppressing less useful ones. Gates can be applied to any DDB outputs, but in practice, they only provide improvement for higher layers. Transition layers are used to control the number of feature maps, implemented by a sequence of convolution, batch normalization (BN) and rectified linear unit (ReLU). In order to make a fair comparison, we limit the number of network parameters to be similar to that of x-vector [6].

In summary, the proposed architecture is a stack of basic components. It aims to improve the representational power of learned features by combining dense connections and gating mechanism. Specifically, each DDB in this network operates similarly to a small TDNN system.

3. Methods

3.1. Dilated dense block

For a DDB, Let $H_l(\cdot)$ denote a general transformation of the l-th layer, $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{l-1}$ are the outputs of the preceding layers, and $\mathbf{x}_0$ is the input of this DDB. All convolutional layers in the same DDB can be designed to output feature maps with matching time and frequency dimensions. That is, $\mathbf{x}_l \in \mathbf{R}^{T\times F\times k}$, where T, F, k are the time, frequency, and channel 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 $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{l-1}$ as

$$\mathbf{x}_l = H_l\left([\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{l-1}]\right) \qquad (1)$$

where $[\cdot]$ is concatenation in the channel dimension. The l-th convolutional layer takes feature maps of size $k \times (l-1) + k_0$ as input, where k_0 is the channel dimension of feature maps corresponding to the DDB's input $\mathbf{x}_0$. To limit the number of DDB parameters, k is set in practice to a small integer, *i.e.*, $k = 20$.

3.2. Gating mechanism

A gating mechanism, inspired by SENet [19], is further utilized to force the model to pay more attention to useful information. It calibrates the output feature maps using learnable parameters to provide a channel-wise attention mechanism.

For a DDB with n layers, its output is $\mathbf{X} = [\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]$, where $\mathbf{X} \in \mathbf{R}^{T\times F\times(k\times n+k_0)}$. Firstly, an aggregation strategy is used to collect global time-frequency statistics of each channel. In our implementation, the simplest global average pooling (GAP) [20] is applied. Note that more sophisticated pooling methods could be utilized for this aggregation operation. Let $\mathbf{s}$ denote the GAP output, the c-th entry of $\mathbf{s}$ is calculated as

$$s_c = \frac{1}{TF} \sum_{t=1}^{T} \sum_{f=1}^{F} \mathbf{X}_c(t, f), \quad c = 1, 2, ..., k \times n + k_0 \tag{2}$$

Then, a structure with two FC layers is employed to capture the inter-dependencies across channels. In this case, the gating vector $\mathbf{g}$ can be expressed as

$$\mathbf{g} = \sigma\left(\mathbf{W}_1 \delta\left(\mathbf{W}_2 \mathbf{s} + \mathbf{b}_2\right) + \mathbf{b}_1\right) \tag{3}$$

where $\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2$ are the parameters of FC layers, δ is ReLU function, and σ is sigmoid function. Finally, the gate block' output $\widetilde{\mathbf{X}}$ is scaled from $\mathbf{X}$ by $\mathbf{g}$

$$\widetilde{\mathbf{X}} = \mathbf{g} \cdot \mathbf{X} \tag{4}$$

4. Experimental setup

4.1. Datasets and acoustic features

We evaluate performance on the test portion of Voxceleb1, and SITW core-core condition. The training dataset includes Voxceleb2 [21] and the development portion of Voxceleb1, without data augmentation. There are 59 speakers included in both SITW and the development portion of Voxceleb1. These speakers are removed from the training dataset. Due to sampled from the real world, these datasets include background noise such as laughter and music. It is therefore challenging to design a robust model to handle this complex environment.

The feature extraction process follows Kaldi [22] i-vector and x-vector baselines. For the i-vector baseline, MFCCs of 24 dimensions with deltas and delta-deltas (*i.e.*, 72 dimensions in total) are used as the input feature, while for the x-vector baseline and our proposed method, MFCCs of 30 dimensions are used. All features are obtained from 25ms windows with 10ms shift between frames. Furthermore, we apply mean-normalization over a sliding window of 3s, and use voice activity detection (VAD) to remove silent segments.

4.2. Model configuration

For neural network based methods, the utterances from the training dataset are randomly cropped to lengths of 2-4s. And utterances with the same duration are grouped into a mini-batch with batch-size=128. All neural networks are implemented using the PyTorch framework [23]. The network is optimized using stochastic gradient descent (SGD) [24] with momentum of 0.95 and weight decay of 5e-4. The learning rate gradually decreases from 1e-2 to 1e-5. Other configurations of each system are listed as follows:

i-vector: This is a baseline system implemented using the Kaldi toolkit. The UBM is a full-covariance GMM with 2048 components, and the dimension of the i-vector is 400. For PLDA backend scoring, the i-vector dimension is further reduced to 200 by LDA.

x-vector: This is a deep embedding learning baseline system. Embeddings are extracted based on a TDNN architecture. There are 5 TDNN layers with BN and ReLU for frame-level processing. These layers generate a 15-frame span of local features. For the utterance-level learning, the output dimensions of statistics pooling layer and two embedding layers (*i.e.*, embedding ***a***, embedding ***b***) are 1500, 512, 512, respectively. Embedding ***a*** is used as speaker representation in our experiments. For PLDA backend scoring, LDA is applied to reduce the dimension of embeddings to 180. More detail can be found in [6].

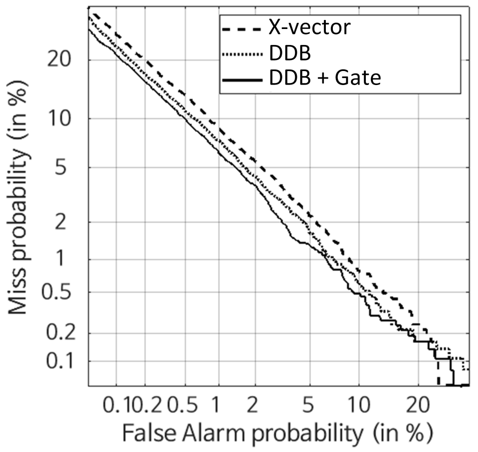

Figure 2: *DET curve for SITW using PLDA backend.*

DDB: This is a comparative SV system using an architecture with DDBs and transition layers. There are four DDBs including 6, 12, 32 and 24 basic units respectively. Each basic unit has 2 convolutional layers, with filter sizes of 1 and 3, dilation sizes of 1 and 2, and channel dimensions of 80 and 20, respectively ($k = 20$ as described in Section 3.1). The transition layer comprises convolutional layer, BN and ReLU, in which the convolutional filter size is equal to 1. All other configuration settings at the utterance-level are the same as in the x-vector baseline system.

DDB + Gate: This is the SV system using the proposed architecture as shown in Fig. 1, in which the gate blocks operate on outputs of DDBs. As described in Section 3.2, the gate blocks are implemented as a GAP-FC-ReLU-FC-sigmoid sandwich. Given c input channels to the gate block, the output dimensions of these two FC layers are $c/8$ and c, respectively.

5. Results

5.1. Main results

The performance is evaluated in terms of equal error rate (EER) and minimum of detection cost function ($\mathrm{DCF}_{0.01}$ and $\mathrm{DCF}_{0.001}$). The main results are reported in Table 1. It is worth noting that even though the proposed architecture is deeper, the number of parameters is maintained to be similar to the baseline x-vector system (the number of parameters is listed in the second column in Table 1).

For the DDB system, the results improve on both i-vector and x-vector systems in all aspects for both SITW and Voxceleb1. The main reason is that the proposed DDB block not only spans a big enough temporal context but also combines lots of different aspects and scales of temporal information. This result suggests that DDB is an efficient architecture for SV tasks. In both evaluation datasets, we also note that all the DDB results using PLDA backend outperform the cosine distance scores. This observation is consistent with the baseline results. It indicates that our method may be more suitable for PLDA backend instead of for cosine distance.

For the DDB+Gate system, we see that it outperforms the other tested methods in almost all metrics, with the exception of a single $\mathrm{DCF}_{0.001}$ result. Again, PLDA tends to perform best. Thanks to the dense connection, large amounts of dif-

Table 1: *Performance of different systems,* ***Boldface*** *values are the best results each in PLDA backend and cosine distance.*

System	Paramters	Backend	SITW			Voxceleb1		
			EER (%)	$DCF_{0.01}$	$DCF_{0.001}$	EER (%)	$DCF_{0.01}$	$DCF_{0.001}$
i-vector	-	cosine	13.40	0.823	0.970	14.26	0.687	0.852
		PLDA	5.88	0.485	0.680	5.59	0.473	0.667
x-vector	8.22M	cosine	10.42	0.905	0.994	3.94	0.436	0.620
		PLDA	3.42	0.356	0.563	3.17	0.328	0.498
DDB	7.46M	cosine	8.64	0.766	0.955	3.43	0.377	0.493
		PLDA	2.92	0.320	0.522	2.63	0.284	**0.427**
DDB+Gate	8.83M	cosine	**5.41**	**0.498**	**0.720**	**2.91**	**0.331**	**0.430**
		PLDA	**2.54**	**0.265**	**0.467**	**2.31**	**0.268**	0.459

Table 2: *EER (%) for different durations on SITW using PLDA backend,* ***Boldface*** *values are the best results.*

System	<15s	15-25s	25-40s
i-vector	7.22	6.33	5.29
x-vector	4.06	3.13	2.75
DDB	3.84	2.59	2.42
DDB+Gate	**3.39**	**2.52**	**1.54**

ferent temporal information over previous layers can be aggregated to the last layer in a DDB block. It is natural that a gating mechanism, guiding attention to more discriminative information, should improve results, and this is indeed evident.

For the PLDA backend, the EER of DDB+Gate achieves 25% and 27% relative improvements over the x-vector baseline for SITW and Voxceleb1, respectively. The results of $DCF_{0.01}$ and $DCF_{0.001}$ are similarly impressive. Meanwhile the i-vector results are inferior to those of neural network methods – indicating that the latter can extract more efficient speaker embeddings in our experiments. Fig. 2 plots DET curves for various systems using PLDA backend for the SITW dataset. The proposed method clearly outperforms x-vector on this noisy and unconstrained dataset, suggesting that our method is robust to noise environment.

5.2. Evaluation on different durations

To determine the relationship between duration and performance, we select utterances from the SITW dataset having durations less than 40s, and arrange them into 3 groups according to duration. One group for utterances between 25 and 40s, one group for utterances between 15 and 25s, and one group for utterances shorter than 15s. Table 2 lists the EER results of all utterance groups, using PLDA backend.

The results show that DDB+Gate achieves the best performance in all duration conditions. As would be expected, when the duration increases, performance improves in all systems [5]. Compared to x-vector, DDB+Gate improves by between 17% and 44% for all durations. This comparison demonstrates that the proposed architecture is also robust to various durations.

5.3. Comparison with state-of-the-art systems

To demonstrate the efficiency of the proposed architecture, we also compare our network to other state-of-the-art systems which have the same test dataset [1]. This is reported in Table 3, where AP refers to average-pooling and SP refers to statistics pooling. The results for ResNet34 and ResNet50 were posted by publishers of Voxceleb2, and could be regarded as baselines for this comparison. The Thin-ResNet34 systems combining with NetVLAD [26] and GhostVLAD [27], were proposed by Xie *et al*, both outperforming the baselines significantly in terms of EER. DDB and DDB+Gate are our proposed architectures, which obtain clear improvements over above state-of-the-art methods. In particular, DDB+Gate achieves the best result thanks to the combination of DDB blocks and gating mechanism.

[1]Note that our training set includes Voxceleb2 and the development portion of Voxceleb1. It is somewhat different from other systems whose training sets only include Voxceleb2.

Table 3: *EER (%) comparison with current state-of-the-art results on Voxceleb1,* ***Boldface*** *value is the best result.*

System	Loss	Aggregation	EER
ResNet34 [21]	Softmax+Contrastive	AP	5.04
ResNet50 [21]	Softmax+Contrastive	AP	4.19
Thin-ResNet34 [25]	Softmax	NetVLAD	3.57
Thin-ResNet34 [25]	Softmax	GhostVLAD	3.22
DDB (Ours)	Softmax	SP	2.63
DDB+Gate (Ours)	Softmax	SP	**2.31**

6. Conclusion and future work

This paper has proposed an effective deep embedding learning architecture which aims to improve the effectiveness of frame-level features for SV tasks. The architecture is a stack of basic components, consisting of DDBs, gate blocks and transition layers. Each DDB contains a sequence of convolutional layers with dilated filters to capture local information of different spans, it acts similarly as a TDNN based system. 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 gate blocks further introduce a channel-wise attention mechanism to exploit inter-dependencies across channels. The transition layers inserted between DDBs perform nonlinear feature transformation to control the overall number of parameters.

We have conducted extensive experiments on two benchmark datasets, SITW and Voxceleb1, to evaluate the effectiveness of our proposed architecture for SV tasks. The results demonstrate significant performance gains over traditional i-vector and x-vector systems on these noisy datasets and over various utterance durations. In the future, we aim to incorporate the aggregation methods in [11, 12, 13], and evaluate the effects of different architecture configurations.

7. Acknowledgement

The work was supported by National Natural Science Foundation of China grant no U1613211.

8. References

[1] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel *et al.*, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

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

[3] Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," in *Proc. ICASSP*, 2014, pp. 1695–1699.

[4] M. McLaren, Y. Lei, and L. Ferrer, "Advances in deep neural network approaches to speaker recognition," in *Proc. ICASSP*, 2015, pp. 4814–4818.

[5] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep neural network embeddings for text-independent speaker verification," in *Proc. Interspeech*, 2017, pp. 999–1003.

[6] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey *et al.*, "X-vectors: Robust DNN embeddings for speaker recognition," in *Proc. ICASSP*, 2018, pp. 5329–5333.

[7] P. Kenny, V. Gupta, T. Stafylakis, P. Ouellet *et al.*, "Deep neural networks for extracting Baum-Welch statistics for speaker recognition," in *The Speaker and Language Recognition Workshop*, 2014, pp. 293–298.

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

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

[10] S. Novoselov, A. Shulipa, I. Kremnev, A. Kozlov, and V. Shchemelinins, "On deep speaker embedding embeddings for text-independent speaker recognition," in *Proc. Odyssey*, 2018.

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

[12] Z. Gao, Y. Song, I. McLoughlin, W. Guo, and L. Dai, "An improved deep embedding learning method for short duration speaker verification," in *Proc. Interspeech*, 2018, pp. 3578–3582.

[13] Y. Tang, G. Ding, J. Huang, X. He, and B. Zhou, "Deep speaker embedding learning with multi-level pooling for text-independent speaker verification," *arXiv preprint arXiv:1902.07821*, 2019.

[14] M. McLaren, L. Ferrer, D. Castan, and A. Lawson, "The speakers in the wild (sitw) speaker recognition database," in *Proc. Interspeech*, 2016, pp. 818–822.

[15] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: A large-scale speaker identification dataset," in *Proc. Interspeech*, 2017.

[16] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proc. NIPs*, 2014.

[17] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. CVPR*, 2017, pp. 4700–4708.

[18] A. V. D. Oord, S. Dieleman, H. Zen, K. Simonyan *et al.*, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[19] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. CVPR*, 2018, pp. 7132–7141.

[20] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2014.

[21] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," in *Proc. Interspeech*, 2018, pp. 1086–1090.

[22] D. Povey, A. Ghoshal, G. Boulianne, L. Burget *et al.*, "The Kaldi speech recognition toolkit," in *Proc. ASRU*. IEEE Signal Processing Society, 2011.

[23] A. Paszke, S. Gross, S. Chintala, G. Chanan *et al.*, "Automatic differentiation in pytorch," 2017.

[24] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proc. IEEE*, vol. 86, no. 11, 1998, pp. 2278–2324.

[25] W. Xie, A. Nagrani, J. S. Chung, and A. Zisserman, "Utterance-level aggregation for speaker recognition in the wild," *arXiv preprint arXiv:1902.10107*, 2019.

[26] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition," in *Proc. CVPR*, 2016.

[27] Y. Zhong, R. Arandjelovic, and A. Zisserman, "Ghostvlad for set-based face recognition," in *Proc. ACCV*, 2018.