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

Densely Connected Time Delay Neural Network for Speaker Verification

Ya-Qi Yu, Wu-Jun Li

National Key Laboratory for Novel Software Technology,
Department of Computer Science and Technology, Nanjing University, China
yuyq@lamda.nju.edu.cn, liwujun@nju.edu.cn

Abstract

Time delay neural network (TDNN) has been widely used in speaker verification tasks. Recently, two TDNN-based models, including extended TDNN (E-TDNN) and factorized TDNN (F-TDNN), are proposed to improve the accuracy of vanilla TDNN. But E-TDNN and F-TDNN increase the number of parameters due to deeper networks, compared with vanilla TDNN. In this paper, we propose a novel TDNN-based model, called densely connected TDNN (D-TDNN), by adopting bottleneck layers and dense connectivity. D-TDNN has fewer parameters than existing TDNN-based models. Furthermore, we propose an improved variant of D-TDNN, called D-TDNN-SS, to employ multiple TDNN branches with short-term and long-term contexts. D-TDNN-SS can integrate the information from multiple TDNN branches with a newly designed channel-wise selection mechanism called statistics-and-selection (SS). Experiments on VoxCeleb datasets show that both D-TDNN and D-TDNN-SS can outperform existing models to achieve state-of-the-art accuracy with fewer parameters, and D-TDNN-SS can achieve better accuracy than D-TDNN.

Index Terms: speaker verification, time delay neural network, dense connectivity, attention

1. Introduction

The task of speaker verification is to decide whether or not to accept a speaker's claim of identity, usually by scoring the similarity between enrollment and test utterances. Existing speaker verification methods typically have two steps. The first step is called speaker embedding, which aims to extract fixed-length vectors from variable-length utterances. The second step is called scoring, which aims to calculate the similarity between speaker embedding vectors. Representative scoring methods include cosine similarity and probabilistic linear discriminant analysis (PLDA) [1].

In the last decade, i-vector [2] used to be predominant for speaker embedding. Recently, more and more researches focus on deep neural network (DNN)-based speaker embedding models [3, 4, 5, 6, 7]. Some works [5, 8] have empirically verified that DNN-based speaker embedding models trained on large-scale datasets can outperform conventional speaker embedding models like i-vector.

Time delay neural network (TDNN) is one of the most popular DNNs in speaker embedding. TDNN has shown state-of-the-art performance on a large range of datasets [4, 5]. The vanilla TDNN [5] consists of three consecutive TDNN layers each of which is a temporal one-dimensional convolutional neural network, followed by two consecutive feed-forward neural network (FNN) layers and then a statistics pooling layer. Recently, two new TDNN-based models, including extended TDNN (E-TDNN) [9] and factorized TDNN (F-TDNN) [10], are proposed for speaker embedding. E-TDNN extends the network depth of vanilla TDNN by adopting four TDNN layers and adding one FNN layer after each TDNN layer. F-TDNN factorizes the weight matrix of each TDNN layer as a product of two smaller matrices to reduce the number of parameters in each layer. Then it adopts more channels and a deeper network than vanilla TDNN. Compared with vanilla TDNN, E-TDNN and F-TDNN can achieve better accuracy but they substantially increase the number of parameters.

To reduce the number of parameters and further improve the accuracy of TDNN, in this paper we propose a novel TDNN-based model, called densely connected TDNN (D-TDNN)[1], for speaker embedding. The contributions of this paper are outlined as follows:

- D-TDNN has fewer parameters than existing TDNN-based models, by adopting bottleneck layers and dense connectivity.
- An improved variant of D-TDNN, called D-TDNN-SS, is proposed to employ multiple TDNN branches with short-term and long-term contexts. D-TDNN-SS can integrate the information from multiple TDNN branches with a newly designed channel-wise selection mechanism called statistics-and-selection (SS).
- Experiments on VoxCeleb datasets demonstrate that both D-TDNN and D-TDNN-SS can outperform existing TDNN-based models to achieve state-of-the-art accuracy with fewer parameters, and D-TDNN-SS can achieve better accuracy than D-TDNN.

2. Related Works

In this section, we briefly review the related works including two TDNN-based speaker embedding models and skip connection related methods.

2.1. Extended TDNN

Compared with vanilla TDNN, extended TDNN (E-TDNN) has one more TDNN layer and interleaves FNN layers between the TDNN layers. Layer interleaving creates a repeated composite structure in E-TDNN and we denote it as E-TDNN layer. The components of an E-TDNN layer are shown in Table 1. Note that batch normalization (BN) is employed before ReLU but omitted in the table.

2.2. Factorized TDNN

Singular value decomposition (SVD) is a commonly used approach for reducing the number of parameters in DNN. Instead of performing SVD after pre-training and then fine-tuning the factorized networks, factorized TDNN (F-TDNN) [11] adopts

[1]D-TDNN is publicly available at https://github.com/yuyq96/D-TDNN.

http://dx.doi.org/10.21437/Interspeech.2020-1275

Table 1: *The components of an E-TDNN layer. t_o denotes the frame offset. d_{out} denotes the output size.*

#	Component Type	Context	Output
1	TDNN-ReLU	t - t_o, t, t + t_o	d_{out}
2	FNN-ReLU	t	d_{out}

Table 2: *The components of a F-TDNN layer. d_{bn} denotes the bottleneck size and $d_{bn} << d_{out}$.*

#	Component Type	Context	Output
1	TDNN (Semi-orthogonal)	t - t_o, t	d_{bn}
2	TDNN-ReLU	t, t + t_o	d_{out}

the factorized architecture but is trained from scratch. Intuitively, F-TDNN adopts bottleneck layers with semi-orthogonal constraint. We denote the factorized TDNN layer as F-TDNN layer. The components of a F-TDNN layer are shown in Table 2.

2.3. Skip Connection

Skip connection helps to create fluent information flow in DNNs. There are two widely used skip connection types including residual connection and direct connection. Residual connection sums up the input and output features. Thus, a transition layer is required if the input and output sizes are different. Direct connection concatenates the input and output features. In F-TDNN [10], direct connections are introduced among some of the bottleneck layers. In DenseNet [12, 13] which adopts dense connectivity, direct connections are introduced among all of the layers in a feed-forward manner.

2.4. Multi-stage Aggregation

Experiments in [14, 15] show that integrating information from different layers can improve the accuracy of speaker embedding models. Features from different layers are transited to match the size at first. Then the features can be summed up [15] or concatenated [14] to aggregate multi-stage information.

3. Methods

This section presents the details of the proposed speaker embedding model called densely connected TDNN (D-TDNN) and its variant D-TDNN-SS.

3.1. Densely Connected TDNN

The basic unit of D-TDNN is called D-TDNN layer. D-TDNN layer is similar to the dense layer of DenseNet but the two-dimensional convolutional neural network (2D CNN) layers in DenseNet are replaced with FNN and TDNN layers. The advantages of FNN and TDNN layers compared to 2D CNN layers include fewer parameters, less computational cost and a total receptive field in frequency-domain, which are preferred in speaker verification tasks.

Specifically, a D-TDNN layer mainly consists of a FNN-based bottleneck layer and a TDNN layer. Let g denote the output size of the TDNN layer, also called growth rate. We set the output size of the bottleneck layer to be twice of the growth rate, i. e., $2g$. Finally, we concatenate the input of the D-TDNN layer and the output of the TDNN layer. The components of a D-TDNN layer are shown in Table 3.

Table 3: *The components of a D-TDNN layer. d_{in} denotes the input size.*

#	Component Type	Context	Output
1	ReLU-FNN	t	$2g$
2	ReLU-TDNN	t - t_o, t, t + t_o	g
	Concatenation		$d_{\text{in}} + g$

Table 4: *The architecture of D-TDNN. K denotes the number of classes. For D-TDNN units, 'size' refers to the growth rate, 'output' refers to the output size after concatenation.*

#	Component Type	Context	Size	Output
1	TDNN-ReLU	t - 2 : t : t + 2	128	128
2	D-TDNN	t - 1, t, t + 1	64	192
3	D-TDNN	t - 1, t, t + 1	64	256
4	D-TDNN	t - 1, t, t + 1	64	320
5	D-TDNN	t - 1, t, t + 1	64	384
6	D-TDNN	t - 1, t, t + 1	64	448
7	D-TDNN	t - 1, t, t + 1	64	512
8	ReLU-FNN	t	256	256
9	D-TDNN	t - 3, t, t + 3	64	320
10	D-TDNN	t - 3, t, t + 3	64	384
11	D-TDNN	t - 3, t, t + 3	64	448
12	D-TDNN	t - 3, t, t + 3	64	512
13	D-TDNN	t - 3, t, t + 3	64	576
14	D-TDNN	t - 3, t, t + 3	64	640
15	D-TDNN	t - 3, t, t + 3	64	704
16	D-TDNN	t - 3, t, t + 3	64	768
17	D-TDNN	t - 3, t, t + 3	64	832
18	D-TDNN	t - 3, t, t + 3	64	896
19	D-TDNN	t - 3, t, t + 3	64	960
20	D-TDNN	t - 3, t, t + 3	64	1024
21	ReLU-FNN	t	512	512
22	Pooling (mean+stddev)	full-seq		1024
23	FNN-BN (Emb.)		512	512
	Softmax		K	K

We use multiple consecutive D-TDNN layers followed by a FNN layer to construct a D-TDNN block. The FNN layer aims to aggregate multi-stage information from different layers. The formulation of the l-th D-TDNN layer is:

$$\mathbf{d}^l = D^l([\mathbf{d}^0, \mathbf{d}^1, \ldots, \mathbf{d}^{l-1}]), \tag{1}$$

where $\mathbf{d}^0$ denotes the input of the D-TDNN block, $\mathbf{d}^l$ denotes the output of the l-th D-TDNN layer, $[\cdot]$ denotes concatenation, $D^l(\cdot)$ denotes the non-linear transformation of the l-th D-TDNN layer.

The whole architecture of D-TDNN is shown in Table 4, which consists of five parts. The first part is component 1, which initializes the number of channels. The second part is components 2 to 8, which is a D-TDNN block whose frame offset is set to 1 with the purpose of learning local features. The third part is components 9 to 21, which is another D-TDNN block whose frame offset is set to 3 with the purpose of catching long-term dependence. The fourth part is component 22, which aggregates the frame-level features and outputs the utterance-level features. The last part is component 23, whose output vectors are used as speaker embeddings.

3.2. Multi-branch Extension

In the base D-TDNN presented at the above subsection, we manually set the frame offsets of D-TDNN layers, which might result in suboptimal solution. Here, we propose a multi-branch extension of D-TDNN to employ multiple TDNN branches with different contexts. Specifically, different TDNN branches can adopt different kernel sizes or different frame offsets so that the multi-branch extension of D-TDNN can adaptively switch between short-term context and long-term context. To effectively combine the information from different branches, we need to introduce a selection mechanism.

Selective kernel (SK) [16] is a dynamic channel-wise selection mechanism based on softmax attention. Specifically, the selection module in SK is structured as GAP-FNN-FNNs-Softmax, where GAP denotes global average pooling. The pooling layer aims to gather the channel-wise information from space dimensions and the following layers aim to estimate the importance of different branches.

We propose to replace the GAP layer with a high-order statistics pooling (HOSP) layer which collects mean as well as high-order statistics. We call the new selection mechanism as statistics-and-selection (SS). Specifically, a SS module is structured as HOSP-FNN-FNNs-Softmax. First, we combine the information from different branches by summing up the features from different branches:

$$\tilde{\mathbf{h}}_t = \sum_{i=1}^{B} \mathbf{h}_t^i, \tag{2}$$

where B denotes the number of branches, $\mathbf{h}_t^i \in \mathbb{R}^C$ denotes the feature vector from the i-th branch at the t-th frame, C denotes the number of channels.

The HOSP layer collects mean, standard deviation, skewness and kurtosis information for each channel:

$$\boldsymbol{\mu} = \frac{1}{T}\sum_{t=1}^{T} \tilde{\mathbf{h}}_t, \tag{3}$$

$$\boldsymbol{\sigma} = \sqrt{\frac{1}{T}\sum_{t=1}^{T} \tilde{\mathbf{h}}_t \odot \tilde{\mathbf{h}}_t - \boldsymbol{\mu} \odot \boldsymbol{\mu}}, \tag{4}$$

$$\mathbf{s} = \frac{1}{T}\sum_{t=1}^{T} \left(\frac{\tilde{\mathbf{h}}_t - \boldsymbol{\mu}}{\boldsymbol{\sigma}}\right)^3, \tag{5}$$

$$\mathbf{k} = \frac{1}{T}\sum_{t=1}^{T} \left(\frac{\tilde{\mathbf{h}}_t - \boldsymbol{\mu}}{\boldsymbol{\sigma}}\right)^4, \tag{6}$$

where T denotes the total number of frames, $\boldsymbol{\mu} \in \mathbb{R}^C$, $\boldsymbol{\sigma} \in \mathbb{R}^C$, $\mathbf{s} \in \mathbb{R}^C$ and $\mathbf{k} \in \mathbb{R}^C$ denote the mean, standard deviation, skewness and kurtosis vectors respectively.

The first FNN layer compresses the channel-wise information into a smaller vector:

$$\mathbf{z} = \mathbf{U}^\top [\boldsymbol{\mu}, \boldsymbol{\sigma}, \mathbf{s}, \mathbf{k}] + \mathbf{p}, \tag{7}$$

where $\mathbf{U} \in \mathbb{R}^{4C \times C/r}$ and $\mathbf{p} \in \mathbb{R}^{C/r}$ denote the weight matrix and bias vector of the first FNN layer, r denotes the reduction factor. In our experiments, $C = g = 64$, r is set to 2.

The following FNN layers and softmax function are used to calculate the attention value for each branch:

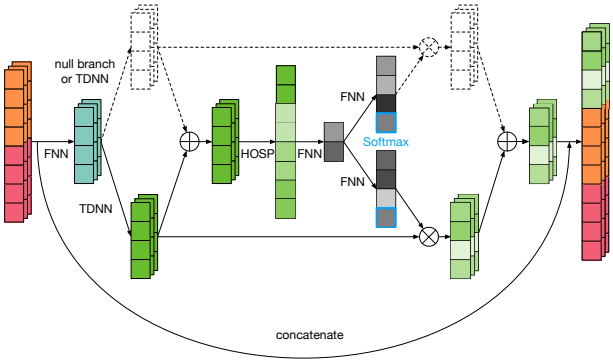

Figure 1: *The structure of D-TDNN-SS layer.*

$$\mathbf{b}^i = \mathbf{V}^{i\top}\mathbf{z} + \mathbf{q}^i, \tag{8}$$

$$a_c^i = \frac{e^{b_c^i}}{\sum_{j=1}^{B} e^{b_c^j}}, \tag{9}$$

$$\mathbf{h}_t = \sum_{i=1}^{B} \mathbf{a}^i \odot \mathbf{h}_t^i, \tag{10}$$

where $\mathbf{V}^i \in \mathbb{R}^{C/r \times C}$ and $\mathbf{q}^i \in \mathbb{R}^C$ denote the weight matrix and bias vector of the FNN layer corresponding to the i-th branch, a_c^i denotes the attention value of the i-th branch for the c-th channel, $\mathbf{h}_t$ denotes the feature vector after selection.

Except for using different kernel sizes or different frame offsets in different TDNN branches, we also propose a null branch for channel suppression. Specifically, we introduce a null feature vector, denoted as $\mathbf{h}_t^0$. Then we only need to change the calculation of attention values in (8) and (9) by letting i and j start from 0 instead of 1.

We call the multi-branch extension of D-TDNN with SS as D-TDNN-SS. The structure of a D-TDNN-SS layer is shown in Figure 1, where we adopt two branches for demonstration. The lower branch is a TDNN branch. The upper branch can be either a null branch or another TDNN branch. Dash line means we do not need to calculate it if we adopt null branch.

3.3. Cosine-based Softmax Loss

Softmax-based cross-entropy loss, also called softmax loss, is usually adopted for training multi-class classifiers. Recently, there have appeared some variants of the softmax loss called cosine-based softmax loss [17, 18], which further employs cosine normalization and introduces an intra-class margin during training. The formulation of cosine-based softmax loss is:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{e^{s\cdot\psi(\cos\theta_{y_i,i})}}{e^{s\cdot\psi(\cos\theta_{y_i,i})} + \sum_{j=1;j\neq y_i}^{K} e^{s\cdot\cos\theta_{j,i}}},$$

where N denotes the number of samples, s denotes the scaling factor, $\psi(\cdot)$ denotes the function for adding margin, $\theta_{j,i}$ denotes the angle between weight vector $\mathbf{w}_j$ and feature $\mathbf{f}_i = f(\mathbf{x}_i;\Theta)$, $\mathbf{x}_i$ denotes the i-th input, $f(\cdot;\Theta)$ denotes the non-linear transformation of the networks and Θ denotes the corresponding parameters.

Angular additive margin softmax (AAM-Softmax) loss [18] adds a margin m on the angular values and $\psi(\cdot)$ is defined as:

$$\psi(\cos\theta) = \cos(\theta + m).$$

Table 5: *Results on the VoxCeleb1 test set.*

Model	Embedding Size	Parameters (M)	Loss Function	Scoring	EER (%)	$DCF_{0.01}$	$DCF_{0.001}$
TDNN	512	4.2	Softmax	Cosine	5.20	0.44	0.60
				PLDA	2.34	0.28	0.38
E-TDNN	512	6.1	Softmax	Cosine	4.65	0.43	0.53
				PLDA	2.08	0.26	0.41
F-TDNN	512	12.4	Softmax	Cosine	4.66	0.41	0.57
				PLDA	1.89	0.21	0.29
D-TDNN	512	2.8	Softmax	Cosine	1.81	0.20	0.28
				PLDA	2.02	0.25	0.30
D-TDNN-SK	512	3.4	Softmax	Cosine	1.63	0.18	0.31
D-TDNN-SS (0)	512	3.0	Softmax	Cosine	1.55	0.20	0.30
D-TDNN-SS	512	3.5	Softmax	Cosine	1.41	0.19	0.24
D-TDNN-SS	128	3.1	AAM-Softmax	Cosine	**1.22**	**0.13**	**0.20**

Table 6: *Dataset for training and evaluation.*

#	Training	Validation	Test
speakers	7291	32	40
utterances	1,198,633	1,481	4,874

4. Experiment

In this section, we conduct experiment to compare the proposed D-TDNN and D-TDNN-SS to TDNN, E-TDNN and F-TDNN.

4.1. Dataset

We conduct experiments on the VoxCeleb1 [19] and VoxCeleb2 [5]. Each dataset contains a development set and a test set. We follow the Kaldi [20] recipe to generate the training examples, but we do not augment the data. Specifically, the whole VoxCeleb2 dataset and the development set of VoxCeleb1 are combined to form a lager development set. We select utterances of 32 speakers from the development set as the validation set and other utterances are used as training set. The detailed information of the combined dataset is shown in Table 6.

We calculate 30-dimensional Mel-frequency cepstrum coefficients (MFCCs) over a 25ms long window every 10ms. Cepstral mean normalization (CMN) is adopted over a 3 seconds long sliding window and energy-based voice activity detection (VAD) is used to remove silent frames. The spectrograms are randomly split into 200 to 400 frames long.

4.2. Implementation Detail

The detailed architectures of TDNN, E-TDNN and F-TDNN can be found in [5, 9, 10]. We implement all models in PyTorch. Specifically, we optimize models using stochastic gradient descent (SGD) with momentum set to 0.95 and weight decay set to 5e-4. The size of mini-batch is 128. The learning rate is initialized as 0.01 and divided by 10 at the 120K-th and 180K-th iterations. Training terminates at the 240K-th iteration.

For AAM-Softmax loss, we set the margin m and scaling factor s to 0.4 and 64.0 respectively. ReLU is replaced with parametric ReLU (PReLU) to stabilize training.

4.3. Result

We adopt two widely used metrics for evaluation, including the equal error rate (EER) [9] and the minimum of detection cost function (DCF) [9] with target probabilities set to 0.01 and 0.001.

The results on the VoxCeleb1 test set are listed in Table 5. Results demonstrate that TDNN, E-TDNN and F-TDNN with PLDA scoring can achieve better accuracy than those with cosine similarity scoring. On the contrary, D-TDNN with cosine similarity scoring can achieve better accuracy than that with PLDA scoring.

Comparing the results of D-TDNN with those of TDNN, E-TDNN and F-TDNN, we find that D-TDNN can outperform existing TDNN-based models to achieve state-of-the-art accuracy. Note that D-TDNN only contains 2.8 million parameters, which is substantially fewer than existing TDNN-based models.

In Table 5, D-TDNN-SS combines a short-term TDNN branch and a long-term TDNN branch, whose frame offsets are set to 1 and 3 respectively. D-TDNN-SS (0) denotes a variant of D-TDNN-SS which adopts a null branch and a TDNN branch. We find that D-TDNN-SS can achieve higher accuracy than D-TDNN-SS (0). Comparing the results between D-TDNN and D-TDNN-SS (0), we find that channel-wise suppression can improve the accuracy of D-TDNN.

D-TDNN-SK is also a multi-branch extension of D-TDNN which has the same branch settings as D-TDNN-SS but adopts selective kernel (SK) for channel-wise selection. Comparing the results of D-TDNN, D-TDNN-SK and D-TDNN-SS, we find that multi-branch extension can improve the accuracy of D-TDNN, and our newly proposed channel-wise selection mechanism SS is better than SK.

We also find that D-TDNN-SS with AAM-Softmax loss can achieve the best results on all metrics.

5. Conclusion

In this paper, we propose a novel TDNN-based model, called D-TDNN, for speaker embedding. Furthermore, we propose an improved variant of D-TDNN, called D-TDNN-SS, to employ multiple TDNN branches with short-term and long-term contexts. Experiments on VoxCeleb datasets demonstrate that our proposed methods can outperform existing TDNN-based models to achieve state-of-the-art accuracy, with fewer parameters.

6. Acknowledgements

This work was supported by the NSFC-NRF Joint Research Project (No. 61861146001). Wu-Jun Li is the corresponding author.

7. References

[1] S. J. D. Prince and J. H. Elder, "Probabilistic linear discriminant analysis for inferences about identity," in *IEEE International Conference on Computer Vision (ICCV)*, 2007, pp. 1–8.

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

[3] 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," *CoRR*, vol. abs/1705.02304, 2017.

[4] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep neural network embeddings for text-independent speaker verification," in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2017, pp. 999–1003.

[5] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 5329–5333.

[6] Y.-Q. Yu, L. Fan, and W.-J. Li, "Ensemble additive margin softmax for speaker verification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 6046–6050.

[7] L. Fan, Q.-Y. Jiang, Y.-Q. Yu, and W.-J. Li, "Deep hashing for speaker identification and retrieval," in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2019, pp. 2908–2912.

[8] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2018, pp. 1086–1090.

[9] D. Snyder, D. Garcia-Romero, G. Sell, A. McCree, D. Povey, and S. Khudanpur, "Speaker recognition for multi-speaker conversations using x-vectors," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 5796–5800.

[10] J. Villalba, N. Chen, D. Snyder, D. Garcia-Romero, A. McCree, G. Sell, J. Borgstrom, F. Richardson, S. Shon, F. Grondin, R. Dehak, L. P. García-Perera, D. Povey, P. A. Torres-Carrasquillo, S. Khudanpur, and N. Dehak, "State-of-the-art speaker recognition for telephone and video speech: The JHU-MIT submission for NIST SRE18," in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2019, pp. 1488–1492.

[11] D. Povey, G. Cheng, Y. Wang, K. Li, H. Xu, M. Yarmohammadi, and S. Khudanpur, "Semi-orthogonal low-rank matrix factorization for deep neural networks," in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2018, pp. 3743–3747.

[12] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2261–2269.

[13] Y. Jiang, Y. Song, I. McLoughlin, Z. Gao, and L.-R. Dai, "An effective deep embedding learning architecture for speaker verification," in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2019, pp. 4040–4044.

[14] Z. Gao, Y. Song, I. McLoughlin, P. Li, Y. Jiang, and L.-R. Dai, "Improving aggregation and loss function for better embedding learning in end-to-end speaker verification system," in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2019, pp. 361–365.

[15] A. Hajavi and A. Etemad, "A deep neural network for short-segment speaker recognition," in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2019, pp. 2878–2882.

[16] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7132–7141.

[17] 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.

[18] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4690–4699.

[19] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: A large-scale speaker identification dataset," in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2017, pp. 2616–2620.

[20] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The kaldi speech recognition toolkit," in *IEEE Workshop on Automatic Speech Recognition and Understanding*, 2011.