IMU&Elevoc System for Far-Field Speaker Verification Challenge 2020

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 the IMU&Elevoc submissions to Far-Filed Speaker Verification Challenge 2020 (FFSVC2020) on the task of far-field text-dependent speaker verification from a single microphone array and microphone arrays. We describes our submissions to this challenge including explored feature extractor topologies, pooling methods, classifier alternatives, and data augmentation strategy. The final system, a fusion of five systems, yields minDCF 0.49 on the Task 1 which achieves 6th place, and minDCF 0.42 on the Task 3 which ranks 3rd place on the leaderboard among all participants.

Index Terms: speaker verification, deep embedding, FFSVC 2020

1. Introduction

The goal of the Far-Filed Speaker Verification Challenge 2020 (FFSVC2020) is to assess the state-of-the-art in speaker recognition under noisy and far-field conditions. There are three different tasks under well-defined conditions: far-field text-dependent speaker verification from a single microphone array, far-field text-independent speaker verification from a single microphone array, and far-field text-dependent speaker verification from distributed microphone arrays. To simulate the real-life scenario, enrollment utterances are recorded from close-talk cellphone, while the test utterances are recorded from far-field microphone arrays. Our submissions focus on Task 1 and Task 3, which is about far-field text-dependent speaker verification from a single microphone array and distributed microphone arrays.

In this paper, we describe our submissions to the FFSVC 2020 challenge. All our systems are based on deep neural network (DNN) embeddings. Specifically, we employ DenseNet as embedding learning architecture [1, 2]. For better obtain utterance-level representation aka deep speaker embedding or embedding for short, we investigate three pooling methods to improve the attentive pooling [3, 4, 5], i.e., the bidirectional attentive pooling, multi-head bidirectional attentive pooling and multi-resolution multi-head bidirectional attentive pooling. The goal of training the network is to produce embeddings that generalize well to speakers beyond those in the training set, aiming to maximize the within-class similarity and minimize the between-class similarity. In general, there are two deep feature learning paradigms, learning with class-level labels and learning with pair-wise labels. The former employs a classification loss function (e.g., softmax cross-entropy loss [6, 7, 8] to optimize the similarity between samples and weight vectors. The latter leverages a metric loss function (e.g., triplet loss [9, 10]) to optimize the similarity between samples. In this end, we explore the performance differences of the speaker system under different deep feature learning paradigms, additive margin softmax loss [11], triplet loss and a unified loss function for two elemental learning paradigms, respectively.

In FFSVC2020, noisy and far-field conditions remain major challenge the models. In order to address these issues, we investigate online data augmentation to improve the robustness of the model. Finally, fusion plays an important role in our final submissions. We use a greedy search to find the five best systems and fuse them to get the primary system.

The rest of this paper is organized as following. Section 2 describes deep neural network with various pooling methods and loss functions in details, followed by the configurations of data sets, the acoustic features, training details in Section 3. Score strategies are introduced in Section 4. The performance of each single system, as well as the fused primary system, is presented in Section 5. We conclude this paper in Section 6.

2. Deep Embedding System

As shown in Figure 1, the deep embedding system can be decomposed into a frame-level feature extractor, a pooling layer, an utterance-level feature extractor and a classifier. The pooling layer converts variable-length frame-level features into a fixed-length embedding. The classifier takes the embedding and produces a posterior probability over the set of training speakers. For our submissions, we employ the DenseNet [1, 2] as frame-level feature extractor and explore three pooling layer topologies and three classifier alternatives.

2.1. Pooling

The aforementioned pooling layer is an essential component to capture long term speaker characteristic for speaker recognition systems [12]. In light of the recent advent of using attention mechanism [13], we describe our proposed pooling methods mainly from the view of attentive pooling [3, 4, 5].

Suppose a speech segment of duration N produces a sequence of N outputs vectors, $\{\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_N\}$. The hidden state at frame index t is a vector $\mathbf{h}_t \in \mathbb{R}^d$ with $t \in [1, N]$ and d representing feature dimension. We first define a self-attentive scoring function, index by i. It computes a score $s_l^{(i)}(x)$ represents l-dimension vector of $\mathbf{x}$. Usually, this is achieved through a fully connected layer as following,

$$s_l^{(i)}(x) = \mathbf{v}_i^T g(\mathbf{W}_i^T x + \mathbf{b}_i), \tag{1}$$

where $g(\cdot)$ is a non-linear activation function and Tanh is chosen here, $\mathbf{W}_i \in \mathbb{R}^{l \times l}$, $\mathbf{V}_i \in \mathbb{R}^l$ and $\mathbf{b}_i \in \mathbb{R}$ are parameters to learn.

The attentive pooling (AP) [3] is utilized for calculating the weighted mean of frame-level feature vectors, which computes

The first two authors contributed equally to this work.

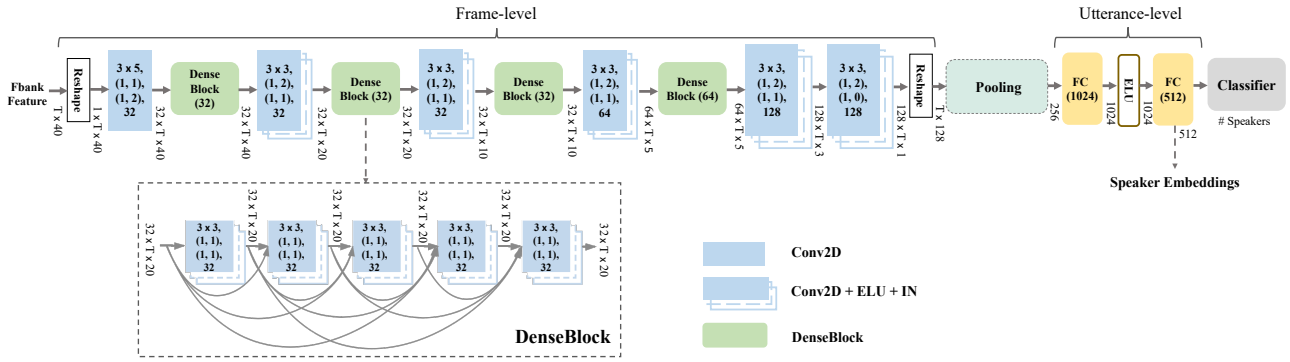

Figure 1: *A schematic of the proposed deep embedding system architecture. For the frame-level stage, each DenseBlock is made up of 5 convolution layers (Conv2D), exponential linear units (ELU) and instance normalizations (IN). The tensor shape after each DenseBlock is in the format: featureMaps × timeSteps × 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 implementation.*

the importance of each frame [4, 5] using softmax as follows,

$$\alpha_t = \frac{e^{s_d(\mathbf{h}_t)}}{\sum_{j=1}^{N} e^{s_d(\mathbf{h}_j)}}, \tag{2}$$

$$U = \sum_{t=1}^{N} \alpha_t \mathbf{h}_t. \tag{3}$$

Notice that we have dropped the superscript i of $s_l^{(i)}$ to emphasize a single-head attention function as used in [4, 5].

2.1.1. Bidirectional attentive pooling

To get more discriminative fixed-dimensional utterance-level representation and capture long term sequence information, Cai et al. [14] proposed an attention-based CNN-BLSTM framework, which combine CNN-BLSTM model with a attentive pooling layer together. Different from [14] 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 BGRU layers. Then the bidirectional utterance-level features $\overrightarrow{U}$ and $\overleftarrow{U}$ are cross concatenated to form an utterance-level speaker embedding as this following:

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

2.1.2. Multi-head bidirectional attentive pooling

We extend the BAP architecture with multi-head, called multi-head bidirectional attentive pooling (MH-BAP), which applies attentive pooling on sub-vectors of each directional frames. Specifically, if a number of K heads are taken into account, it first splits the encoded frame $\mathbf{h}_t$ into K non-overlapping homogenous sub-vectors, $\mathbf{h}_t^{(1)}, \mathbf{h}_t^{(2)}, ..., \mathbf{h}_t^{(K)}$ with $\mathbf{h}_t^{(i)} \in \mathbb{R}^{d/K}$. Then, the sequence of each directional sub-vectors $\mathbf{h}^{(i)} = [\mathbf{h}_1^{(i)}, \mathbf{h}_2^{(i)}, ..., \mathbf{h}_N^{(i)}]$ is processed with its specific attentive pooling, extracting speech characteristics from the temporal vector sequence $\mathbf{h}^{(i)}$ with index $i \in \{1, 2, ..., K\}$ as this following:

$$\alpha_t^{(i)} = \frac{e^{s_d^{(i)}(\mathbf{h}_t^{(i)})}}{\sum_{j=1}^{N} e^{s_{d/K}^{(i)}(\mathbf{h}_j^{(i)})}}, \tag{7}$$

$$U^{(i)} = \sum_{t=1}^{N} \alpha_t^{(i)} \mathbf{h}_t^{(i)}. \tag{8}$$

Finally, the utterance-level speaker embedding U is formed as a concatenation of the bidirectional sub-embeddings as:

$$U = CrossCat([\overrightarrow{U}^{(1)}, ..., \overrightarrow{U}^{(K)}], [\overleftarrow{U}^{(1)}, ..., \overleftarrow{U}^{(K)}]). \tag{9}$$

2.1.3. Multi-resolution multi-head bidirectional attentive pooling

As the speech characteristics are obtained through aggregation with attentive weights, Wang et al. [15] proposed the multi-resolution multi-head attention which increase or decrease the resolution of the attentive wights with a temperature parameter. We extend this idea to incorporate the MH-BAP architecture, called multi-resolution multi-head bidirectional attentive pooling (MRMH-BAP). The Eq.(7) is changed to be

$$\alpha_t^{(i)} = \frac{e^{s_d^{(i)}(\mathbf{h}_t^{(i)})/T_i}}{\sum_{j=1}^{N} e^{s_{d/K}^{(i)}(\mathbf{h}_j^{(i)})/T_i}}, \tag{10}$$

where $T_i = max(1, \lfloor \frac{i-1}{2} \rfloor \times 5), i = 1, 2, ..., K$.

2.2. Loss function

The goal of the loss function is to define a task that matches the deployment setup as closely as possible while allowing for efficient deep neural network training through back-propagation. In general, there are two deep feature learning paradigms, learning with class-level labels and learning with pair-wise labels.

In our submissions, we explore different deep feature learning paradigms, compare the performance difference of different loss functions.

Given a single sample $\mathbf{x}$ in the feature space, assumes that there are K within-class similarity scores and L between-class similarity scores associated with $\mathbf{x}$. We denote these similarity scores as $\{\mathbf{s}_p^i\}(i = 1, 2, ..., K)$ and $\{\mathbf{s}_n^j\}(j = 1, 2, ..., L)$, respectively.

2.2.1. Additive margin softmax loss

Given class-level labels, we use additive margin softmax loss (AM-Softmax), proposed in [16], as the loss function. We calculate similarity scores between $\mathbf{x}$ and weights vector $\mathbf{w}_i(i = 1, 2, ...N)$ (N is the number of training classes) in the classification layer. Specifically, we get $(N-1)$ between-class similarity scores by: $\mathbf{s}_n^j = \mathbf{w}_j^T\mathbf{x}/(\|\mathbf{w}_j\| \|\mathbf{x}\|)$($\mathbf{w}_j$ is the j-th non-target weight vector). Additionally, we get a single within-class similarity core (with the superscript omitted) $\mathbf{s}_p = \mathbf{w}_y^T\mathbf{x}/(\|\mathbf{w}_y\| \|\mathbf{x}\|)$. With these prerequisites, the AM-Softmax loss as this following:

$$\begin{aligned} L_{am} &= log[1 + \sum_{j=1}^{N-1} exp(\gamma(\mathbf{s}_n^j + m))exp(-\gamma\mathbf{s}_p)] \\ &= -log\frac{exp(\gamma(\mathbf{s}_p - m))}{exp(\gamma(\mathbf{s}_p - m)) + \sum_{j=1}^{N-1} exp(\gamma\mathbf{s}_n^j)}, \end{aligned} \tag{11}$$

in which γ is a scale factor and m is a margin for better similarity separation.

2.2.2. Triplet loss

Given pair-wise labels, we use Triplet loss, proposed in [10], as the loss function. We calculate the similarity scores between $\mathbf{x}$ and the other features in the mini-batch. Specifically, $\mathbf{s}_n^j = (\mathbf{x}_n^j)^T\mathbf{x}/(\left\|\mathbf{x}_n^j\right\| \|\mathbf{x}\|)$ ($\mathbf{x}_n^j$ is the j-th sample in the negative sample set $\mathcal{N}$) and $\mathbf{s}_p^i = (\mathbf{x}_p^i)^T\mathbf{x}/(\left\|\mathbf{x}_p^i\right\| \|\mathbf{x}\|)$ ($\mathbf{x}_p^i$ is the i-th sample in the positive sample in the positive sample set $\mathcal{P}$). Correspondingly, $K = |\mathcal{P}|$, $L = |\mathcal{N}|$. The triplet loss as this following:

$$L_{tri} = [\mathbf{s}_n^j - \mathbf{s}_p^i + m]_+, \tag{12}$$

where the operator $[x]_+ = max(x, 0)$ represents triplet selection.

2.2.3. Circle loss

Given class-level labels and pair-wise labels, a unified formula for two elemental deep feature learning paradigms proposed in [17], called Circle loss, as the loss function. To minimize each $\mathbf{s}_n^j$ as well as to maximize $\mathbf{s}_p^i$, ($\forall i \in 1, 2, ..., K$, $\forall j \in 1, 2, ..., L$), the corresponding loss function is:

$$\begin{aligned} L_{cir} &= log[1 + \sum_{i=1}^{K}\sum_{j=1}^{L} exp(\gamma(\mathbf{s}_n^j - \mathbf{s}_p^i + m))] \\ &= log[1 + \sum_{j=1}^{L} exp(\gamma(\mathbf{s}_n^j + m)) \sum_{i=1}^{K} exp(\gamma(-\mathbf{s}_p^i))]. \end{aligned} \tag{13}$$

3. Training

3.1. Data preparation

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. 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. For all of our submissions, we train our DNN on the first 30 utterances of FFSVC2020 training dataset, and SLR85 HI-MIA dataset from `openslr.org`. 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.

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.

3.2. Online Data Augmentation

We use the online data augmentation strategy in [18] to make the speaker embeddings more robust. The public MUSAN [19] and RIR_NOISES [20] 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.

3.3. DNN training with PyTorch

All of the submission models are trained using PyTorch [21] 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.

4. Scoring

We used both Probabilistic Linear Discriminative Analysis (PLDA) [23, 24] 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 [12]. In the cosine scoring, we simply evaluate the pairwise comparisons using the cosine distance. Recordings from the iPhone at 25cm distance are selected for enrollment. 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.

5. Results

We list the performance of systems on FFSVC2020 in Table 1. The primary measure metric is the minimum detection cost function (minDCF) with P_{target} = 0.01. It should be mentioned that TDNN is used as the feature extractor in system D. The structure of this extractor is the same as that in the x-vector system [25], and ResNet of the system E is a reimplementation

Table 1: *Performance of our systems on the FFSVC2020 development set, and the official evaluation.* ***Boldface*** *values are the best results each in PLDA backend and cosine distance. DenseNet system as shown in Figure 1. AP, BAP, MH-BAP and MRMH-BAP pooling methods are described in Section 2.1.*

System	Loss function	Pooling	Scoring	Development Set				Evaluation Set	
				Task 1		Task 3		Task 1	Task3
				minDCF	EER(%)	minDCF	EER(%)	minDCF	minDCF
A1(DenseNet)	AM-Softmax (m=0.35, γ=10)	AP	Cosine	0.54	**4.73**	0.51	4.69	-	-
			PLDA	**0.52**	4.81	**0.49**	**4.57**	-	-
A2(DenseNet)	AM-Softmax (m=0.35, γ=10)	BAP	Cosine	**0.48**	4.60	0.46	4.27	-	-
			PLDA	0.49	**4.57**	**0.45**	**4.12**	0.57	0.53
A3(DenseNet)	AM-Softmax (m=0.35, γ=10)	MH-BAP	Cosine	0.47	4.59	0.43	4.23	-	-
			PLDA	**0.45**	**4.43**	**0.42**	**4.01**	-	-
A4(DenseNet)	AM-Softmax (m=0.35, γ=10)	MRMH-BAP	Cosine	0.43	4.06	0.42	3.97	-	-
			PLDA	**0.41**	**3.93**	**0.39**	**3.85**	0.53	0.48
B1(DenseNet)	TripletLoss (m=0.25)	MH-BAP	Cosine	0.53	**4.65**	0.50	4.45	-	-
			PLDA	**0.51**	4.70	**0.47**	**4.36**	-	-
B2(DenseNet)	TripletLoss (m=0.25)	MRMH-BAP	Cosine	0.49	4.52	0.48	**4.29**	-	-
			PLDA	**0.46**	**4.48**	**0.44**	4.34	0.59	0.58
C1(DenseNet)	CircleLoss (m=0.35, γ=256)	MH-BAP	Cosine	0.37	**3.39**	0.38	3.31	-	-
			PLDA	**0.35**	3.47	**0.36**	**3.26**	-	-
C2(DenseNet)	CircleLoss (m=0.35, γ=256)	MRMH-BAP	Cosine	0.36	3.29	0.37	3.21	-	-
			PLDA	**0.34**	**3.16**	**0.35**	**3.09**	**0.52**	**0.47**
D(TDNN)	CircleLoss (m=0.35, γ=256)	MRMH-BAP	Cosine	0.45	3.86	0.43	3.61	-	-
			PLDA	**0.42**	**3.81**	**0.40**	**3.58**	0.56	0.52
E(ResNet)	CircleLoss (m=0.35, γ=256)	MRMH-BAP	Cosine	0.38	**3.43**	**0.38**	**3.37**	-	-
			PLDA	**0.37**	3.51	0.41	3.40	0.54	0.50
FFSVC2020 Baseline[22]	-	-	-	0.57	6.01	0.59	5.42	0.62	0.66
Fusion(A4+B2+C2+D+E)	-	-	PLDA	0.32	3.05	0.33	2.97	0.49	0.42

of the feature extractor of the official benchmark system [22]. System C2 is our proposed best single system. Comparing systems C2, D and E, we observe that DenseNet outperforms the TDNN and ResNet. 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 17.74%, 28.78% minDCFs relative reductions on the evaluation set of Task 1 and Task 3 respectively.

In Table 1, under same loss function condition, we can see the effect of different pooling methods. Comparing the systems A1, A2, A3 and A4 (B1 and B2, or C1 and C2), we observe that MRMH-BAP performs best. That means that using multiple heads and having multiple resolutions on attention heads with different temperatures lead to improved certainty of attentive weights in the pooling layer. In particular, the bidirectional recurrent layer in the pooling method consists of two BGRU layers with 128 cells in each direction and set the attention head $K = 4$.

From Table 1, we can see clearly the loss function is important for deep feature learning. Compare with A4, B2 and C2, Circle loss marginally outperforms the AM-Softmax and Triplet loss.

The systems noted in bold face in Table 1 are combined by averaging their scores and submitted as our final entry for the Task 1 and Task 3. We have also submitted these individual systems to the leaderboard to assess their performance on the evaluation set. Overall, we observe that there is some performance gap between the development and evaluation sets but the relative ranking of systems is consistent. Finally, we note that combining a set of diverse systems (averaging their scores) outperforms a single network (i.e., C2 vs A4+B2+C2+D+E).

6. Conclusions

In this paper, we provide an overview of IMU&Elevoc systems submitted to FFSVC2020. The details about our systems, including the usage of data sets, various loss functions, pooling methods and fusion strategies, are described. What we learn from FFSVC2020 include: online data augmentation is helpful; densely concatenating the outputs of convolutional layers improves efficiency; adding bidirectional recurrent layers to the attentive pooling layer helps capture long temporal context information; using multiple heads for attentive pooling with an additional temperature hyperparameter for each head, significantly improves the performance; circle loss benefits deep feature learning and helps extract more discriminative speaker embeddings; and finally, fusion almost always helps.

7. 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).

8. References

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

[2] 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.

[3] 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.

[4] 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.

[5] G. Bhattacharya, M. J. Alam, and P. Kenny, "Deep speaker embeddings for short-duration speaker verification." in *Interspeech*, 2017, pp. 1517–1521.

[6] Y. Sun, X. Wang, and X. Tang, "Deep learning face representation from predicting 10,000 classes," in *CVPR*, 2014, pp. 1891–1898.

[7] W. Liu, Y. Wen, Z. Yu, and M. Yang, "Large-margin softmax loss for convolutional neural networks." in *ICML*, vol. 2, no. 3, 2016, p. 7.

[8] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *ECCV*, 2016, pp. 499–515.

[9] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *International Workshop on Similarity-Based Pattern Recognition*, 2015, pp. 84–92.

[10] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *CVPR*, 2015, pp. 815–823.

[11] 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.

[12] 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.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017, pp. 5998–6008.

[14] 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.

[15] Z. Wang, K. Yao, X. Li, and S. Fang, "Multi-resolution multi-head attention in deep speaker embedding," in *ICASSP*, 2020, pp. 6464–6468.

[16] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, "Cosface: Large margin cosine loss for deep face recognition," in *CVPR*, 2018, pp. 5265–5274.

[17] Y. Sun, C. Cheng, Y. Zhang, C. Zhang, L. Zheng, Z. Wang, and Y. Wei, "Circle loss: A unified perspective of pair similarity optimization," in *CVPR*, 2020, pp. 6398–6407.

[18] 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.

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

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

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