Odyssey 2020 The Speaker and Language Recognition Workshop
1-5 November 2020, Tokyo, Japan

Learning Mixture Representation for Deep Speaker Embedding Using Attention

Weiwei Lin, Man-Wai Mak, and Lu Yi

Dept. of Electronic and Information Engineering,
The Hong Kong Polytechnic University, Hong Kong, China

Abstract

Almost all speaker recognition systems involve a step that converts a sequence of frame-level features to a fixed dimension representation. In the context of deep neural networks, it is referred to as statistics pooling. In state-of-the-art speak recognition systems, statistics pooling is implemented by concatenating the mean and standard deviation of a sequence of frame-level features. However, a single mean and standard deviation are limited descriptive statistics for an acoustic sequence even with a powerful feature extractor such as convolutional neural networks. In this paper, we propose a novel statistics pooling method that can produce more descriptive statistics through a mixture representation. Our approach is inspired by the expectation-maximization (EM) algorithm in Gaussian mixture models (GMMs). However, unlike the GMMs, the mixture assignments are given by an attention mechanism instead of the Euclidean distances between frame-level features and explicit centers. Applying the proposed attention mechanism to a 121-layer Densenet, we achieve an EER of 1.1% in VoxCeleb1 and an EER of 4.77% in the VOiCES 2019 evaluation set.

1. Introduction

Speaker verification has witnessed impressive advances in recent years [1–7]. It has become an increasingly popular choice for biometric authentication. The traditional factor analysis based speaker representation (i-vector) has been replaced by deep neural network speaker embedding, which has shown remarkable robustness to noise, reverberation and domain shift [8]. Among the DNN-based approaches, the x-vector is considered state-of-the-art [5]. Compared with previous DNN-based approaches [2–6], several features of the x-vector stand out: (1) the use of extensive data augmentations with real-life noise and reverberation, (2) the use of short training segments randomly sampled from original utterances, and (3) the statistics pooling of a sequence of frame-level features to produce a fixed dimension representation. It has been shown that the x-vector is superior to the i-vector and other DNN-based speaker embeddings. In both NIST SRE 2016 and 2018 evaluations, the leading teams used x-vector-based systems [8].

The process of converting an acoustic sequence to a fixed-dimensional representation is referred to as statistics pooling in the literature [5]. It has been shown that the statistics pooling layer of an x-vector network can have a significant impact on the performance of speaker embeddings. In [5], it was shown that using both mean and standard deviation of frame-level features has significant improvement over using the mean alone. In [9], the authors argued that each frame should not be treated as equal in computing the statistics as some frames are more discriminative than the others. Therefore, an attention mechanism was introduced to weight each frame. We refer to this statistics pooling as attentive statistics pooling (ASP) in the rest of the paper. The ASP is further extended to multiple-head attention in [10], where multiple attention heads were used to weight each frame to produce multiple means and standard deviations for an input sequence. A regularization term was also introduced to prevent the attention heads from learning the same information [10]. In [11], the idea is further extended to the frequency axis. Instead of weighting the frames alone, the authors proposed to weight both the frames and the frequency bins simultaneously.

Although attentive statistics pooling shows promising results, we believe that weighting frame-level features is not enough to produce ideal speaker representation. We believe that the statistics pooling layer needs to produce a good statistical summary of the frame-level features. However, means and standard deviations are limited statistics for summarizing a distribution. Because mixture models are more powerful models to capture the underlying distribution, we propose to incorporate mixture representation into the statistics pooling layer. Although the attention mechanism in [10] can produce multiple means and standard deviations for an input sequence, it does not produce a valid mixture representation. A valid mixture model requires the probability mass sums to 1 across all mixture assignments for the same input. There is no mechanism in [10] to enforce this requirement. Therefore, the multiple attention heads merely increase the number of parameters in the network without producing a richer representation from a statistical point of view. Instead, our method explicitly uses an attention mechanism to produce valid mixture assignments, from which the means and standard deviations are computed just like the M-step in Gaussian mixture models.

Work similar to ours was proposed in [12], where a dictionary of centers are learned from data and the center assignments are computed using the Euclidean distances between frame-level features and the centers. There are two key differences between our method and that of [12]. First, unlike [12], our method does not explicitly use the *centers* to compute the assignments; instead, we use an attention mechanism to produce mixture assignments. Because the score function in the attention mechanism is more general than Euclidean distances, our method can have more desirable mixture assignments. Second, the means and standard deviations are computed based on the sufficient statistics as in the EM for GMMs, which makes our multi-head attention to have a probabilistic interpretation.

This work was supported by RGC of Hong Kong, Grant No. PolyU 152137/17E and NSFC of China, Grant No. 61971371.

10.21437/Odyssey.2020-30

Model	# of flops	# of parameters
X-vector network	1.060G	4.319M
Wide x-vector network	3.740G	11.726M
Densenet121	0.959G	10.338M

Table 1: A summary of the number of parameters and the number of floating-point operations in a forward pass of the Densenet121, the x-vector network, and the wide x-vector network for an input size of 40×400.

2. Network Architectures

2.1. X-vector Architecture

The x-vector network consists of three parts: Frame-level time delay neural networks (TDNNs), utterance-level fully-connected (FC) layers, and a statistics pooling layer that bridges the frame-level layers and utterance-level layers [5, 13]. A TDNN is a special form of convolutional neural networks (CNNs). It skips the computation at chosen temporal positions while maintaining the same receptive field size as a CNN. A statistics pooling layer concatenates the mean and standard deviation of the activations from the last convolutional layer. The concatenated means and standard deviations are passed to two FC layers. The network is trained to minimize the standard cross-entropy loss. The network is trained using small chunks of acoustic sequences derived from the original utterances. The typical chunk length ranges from 200-ms to 400-ms. After the network is trained, the embedding of each utterance is extracted from the first affine layer after the statistics pooling layer. A backend consisting of LDA and PLDA models is trained using the embeddings as input [14].

2.2. Densenet Architecture for Speaker Embedding

Densenets are proposed in [15] for computer vision. A Densenet comprises two block types, namely, dense block and transition block. In a dense block, each layer is connected by all the output from the previous layers. To prevent the number of feature maps from growing excessively, a transition block is introduced to reduce the feature map size. Suppose each convolutional layer produces k feature maps, then the l-th layer inside the block has $k_0 + k \times (l-1)$ feature maps, where k_0 is the number of channels in the input layer. The parameter k is referred to as the growth rate. In this work, we used a dense network composed of 1-dimensional convolution instead of 2D convolution. We used the same statistics pooling layer as that of the x-vector network. Because max-pooling and average pooling do not work well in speaker recognition, we replaced the max-pooling by stride 2 convolution layers. Table 2 shows our network architecture. Table 1 summarizes the number of parameters and the number of floating-point operations in a forward pass for our Densenet121 and the x-vector network for an input size of 40×400. We also compared our Densenet121 with a wide x-vector network in which the channel size in each convolutional layer are doubled.

2.3. Additive Margin Softmax

Margin-based loss has been very successful in face recognition and speaker recognition [16]. Additive margin loss enforces a minimum margin m between the target class and non-target classes:

$$\begin{aligned}\mathcal{L}_{AMS} &= -\frac{1}{n}\sum_{i=1}^{n}\log\frac{e^{s\cdot\left(\cos\theta_{y_i}-m\right)}}{e^{s\cdot\left(\cos\theta_{y_i}-m\right)}+\sum_{j=1,j\neq y_i}^{c}e^{s\cdot\cos\theta_j}} \\ &= -\frac{1}{n}\sum_{i=1}^{n}\log\frac{e^{s\cdot\left(\mathbf{W}_{y_i}^{\mathsf{T}}\mathbf{x}_i-m\right)}}{e^{s\cdot\left(\mathbf{W}_{y_i}^{\mathsf{T}}\mathbf{x}_i-m\right)}+\sum_{j=1,j\neq y_i}^{c}e^{s\mathbf{W}_j^{\mathsf{T}}\mathbf{x}_i}},\end{aligned} \tag{1}$$

where $\mathbf{W}$ is a weight matrix ($\mathbf{W}_j$ is the j-th column of $\mathbf{W}$) and $\mathbf{x}$ is an embedding vector, both of which are normalized to have unit length. s is a scaling constant.

3. Attention Mechanisms in Statistics Pooling Layers

3.1. Statistics Pooling Layer

Given a sequence of frame-level features $\{\mathbf{h}_t\}_{t=1}^{T}$, a statistics pooling layer computes the mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$ over $\{\mathbf{h}_t\}_{t=1}^{T}$:

$$\boldsymbol{\mu} = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{h}_t \tag{2}$$

$$\boldsymbol{\sigma} = \sqrt{\frac{1}{T}\text{Diag}\left(\sum_{t=1}^{T}\mathbf{h}_t\mathbf{h}_t^{\mathsf{T}} - \boldsymbol{\mu}\boldsymbol{\mu}^{\mathsf{T}}\right)}, \tag{3}$$

where $\text{Diag}(\mathbf{A})$ means the diagonal of $\mathbf{A}$ and the square root is applied element-wise. The utterance-level representation $\mathbf{x}$ is obtained by concatenating the mean and standard deviation:

$$\mathbf{x} = \text{CONCAT}(\boldsymbol{\mu}, \boldsymbol{\sigma}) \tag{4}$$

3.2. Attentive Statistics Pooling

In [9, 10], the authors argued that each frame should not be treated equally when computing the statistics for an input sequence. They proposed to use an attention mechanism to calculating the $\text{score}(\mathbf{h}_t, \mathbf{v})$ for each frame, where $\mathbf{v}$ is an attention vector and $\text{score}(.,.)$ is a score function. A score function parameterized by a single-layer neural network was used:

$$\text{score}(\mathbf{h}_t, \mathbf{v}) = \mathbf{v}^{\mathsf{T}} f(\mathbf{W}\mathbf{h}_t + \mathbf{b}), \tag{5}$$

where $f(.)$ is a non-linear activation function. The score is normalized across frames using a softmax function:

$$\alpha_t = \frac{\exp\left(\text{score}(\mathbf{h}_t, \mathbf{v})\right)}{\sum_{t=1}^{T}\exp\left(\text{score}(\mathbf{h}_t, \mathbf{v})\right)}. \tag{6}$$

The normalized scores are used to weight each frame:

$$\boldsymbol{\mu} = \sum_{t=1}^{T}\alpha_t\boldsymbol{h}_t \tag{7}$$

$$\boldsymbol{\sigma} = \sqrt{\text{Diag}\left(\sum_{t=1}^{T}\alpha_t\mathbf{h}_t\mathbf{h}_t^{\mathsf{T}} - \boldsymbol{\mu}\boldsymbol{\mu}^{\mathsf{T}}\right)}. \tag{8}$$

The above procedure can be easily extended to multi-head attention. Suppose each attention head k is parameterized by $\mathbf{v}_k$, where $k = 1, \ldots, K$. Then the attention scores for each head can be written as:

$$\text{score}(\mathbf{h}_t, \mathbf{v}_k) = \mathbf{v}_k^{\mathsf{T}} f(\mathbf{W}\mathbf{h}_t + \mathbf{b}), \tag{9}$$

Layers	Output Size	DenseNet-121
Convolution	400×40	conv 3
Dense Block (1)	400×80	$\begin{bmatrix} \text{conv } 1 \\ \text{conv } 3 \end{bmatrix} \times 6$
Transition Layer (1)	200×320	conv 2 stride 2
Dense Block (2)	200×320	$\begin{bmatrix} \text{conv } 1 \\ \text{conv } 3 \end{bmatrix} \times 12$
Transition Layer (2)	100×640	conv 2 stride 2
Dense Block (3)	100×640	$\begin{bmatrix} \text{conv } 1 \\ \text{conv } 3 \end{bmatrix} \times 24$
Transition Layer (3)	50×1280	conv 2 stride 2
Dense Block (4)	50×1280	$\begin{bmatrix} \text{conv } 1 \\ \text{conv } 3 \end{bmatrix} \times 16$
Stats-pooling Layer	50×2560	-
FC	1	2560×256 Linear
Classification Layer	1	$256 \times$ # of classes AM-Softmax

Table 2: Densenet architecture for speaker embedding. The growth rate for the networks is 40. Note that each "conv" layer shown in the table corresponds to the sequence BN-ReLU-Conv.

where $\mathbf{v}_k$ is the attention vector for head k. The normalized scores are obtained by:

$$\alpha_{t,k} = \frac{\exp\left(\text{score}(\mathbf{h}_t, \mathbf{v}_k)\right)}{\sum_{t=1}^{T} \exp\left(\text{score}(\mathbf{h}_t, \mathbf{v}_k)\right)}. \tag{10}$$

The normalized scores are used to weight the frame-level features:

$$\boldsymbol{\mu}_k = \sum_{t=1}^{T} \alpha_{t,k} \boldsymbol{h}_t \tag{11}$$

$$\boldsymbol{\sigma}_k = \sqrt{\text{Diag}\left(\sum_{t=1}^{T} \alpha_{t,k} \mathbf{h}_t \mathbf{h}_t^{\mathsf{T}} - \boldsymbol{\mu}\boldsymbol{\mu}^{\mathsf{T}}\right)}. \tag{12}$$

The utterance-level representation $\mathbf{x}$ is obtained by concatenating the means and standard deviations from all attention heads:

$$\mathbf{x} = \text{CONCAT}\left(\boldsymbol{\mu}_1, \boldsymbol{\sigma}_1, \ldots, \boldsymbol{\mu}_K, \boldsymbol{\sigma}_K\right). \tag{13}$$

3.3. Statistics Pooling with Mixture Representation

Even with multiple attention heads, the attention mechanism in [9,10] can not produce valid mixture representations. A valid mixture model requires the probability mass sums to 1 across all mixture assignments for the same input. There is no mechanism in [10] to enforce this requirement. Therefore, the multiple attention heads merely increase the number of parameters of the network without producing a richer representation from a statistical point of view. We propose a novel attention mechanism inspired by the Gaussian mixture model that can produce rich mixture representations.

Gaussian mixture models (GMMs), which are exemplified by the GMM-HMM in ASR and the i-vectors in speak recognition [1], are very popular in the speech community to model complex distributions. A GMM is typically trained by the EM algorithm that alternates between the E-step and the M-step [17]. Assume that the mixture assignments of frame $\mathbf{h}_t$ are $\alpha_{t,k}$, where $k = 1, \ldots, K$; then the parameters of each mixture component can be computed by:

$$N_k = \sum_{t=1}^{T} \alpha_{t,k} \tag{14}$$

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{t=1}^{T} \alpha_{t,k} \mathbf{h}_t \tag{15}$$

$$\boldsymbol{\sigma}_k = \sqrt{\text{Diag}\left(\frac{1}{N_k} \sum_{t=1}^{T} \alpha_{t,k} \left(\mathbf{h}_t - \boldsymbol{\mu}_k\right)\left(\mathbf{h}_t - \boldsymbol{\mu}_k\right)^{\mathsf{T}}\right)}. \tag{16}$$

The attention scores are still produced by the score function in Eq. 9. However, the scores are normalized across the attention heads:

$$\alpha_{t,k} = \frac{\exp\left(\text{score}(\mathbf{h}_t, \mathbf{v}_k)\right)}{\sum_{k=1}^{K} \exp\left(\text{score}(\mathbf{h}_t, \mathbf{v}_k)\right)}. \tag{17}$$

The normalized scores are used to compute the means and standard deviations as in Eqs. 14–16. We refer to this statistics pooling method as mixture representation pooling (MRP) in the rest of the paper.

4. Experiments

4.1. Data Preparation

The training data includes the VoxCeleb1 development set and the VoxCeleb2 development set [18, 19]. We followed the data augmentation strategy in the Kaldi SRE16 receipt. The training data were augmented by adding noise, music, reverb, and babble to the original speech files in the datasets. After filtering out the utterances shorter than 400-ms and the speakers with less than 8 utterances, we are left with 7,302 speakers. We used the filter-bank feature implemented in Kaldi. We used a frame length of 25ms. The number of mel-scale filters is 40 and the lower and upper cutoff frequencies covered by the triangular filters are 20Hz and 7,600Hz respectively. Mean normalization was applied to the filter-bank features using a 3-second sliding window. Non-speech frames were removed by Kaldi's energy-based voice activity detector.

Model	Pooling Method	VoxCeleb1		VOiCES19-dev		VOiCES19-eval	
		EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF
X-vector network	Mean & STD	2.14	0.197	2.66	0.300	6.98	0.520
Wide x-vector network	Mean & STD	2.03	0.219	2.65	0.294	6.62	0.503
Densenet121	Mean & STD	1.37	0.156	**1.53**	0.222	5.53	0.415
Densenet121	ASP	1.22	0.150	1.84	0.197	5.20	0.402
Densenet121	MRP	**1.10**	**0.131**	1.65	**0.184**	**4.77**	**0.390**

Table 3: Systems performance on VoxCeleb1, VOiCES19-dev, and VOiCES19-eval. For VoxCeleb1, we used cosine scoring. For VOiCES19-dev and VOiCES19-eval, we used a PLDA backend.

# of Heads	EER(%)		minDCF	
	ASP	MRP	ASP	MRP
1	**5.20**	5.53	**0.402**	0.415
2	5.50	4.86	0.428	0.397
3	5.69	**4.77**	0.442	**0.390**
4	5.77	5.20	0.453	0.404
5	6.00	5.73	0.430	0.413

Table 4: Performance of attentive statistics pooling (ASP) and our proposed method (MRP) with different numbers of attention heads using Densenet121.

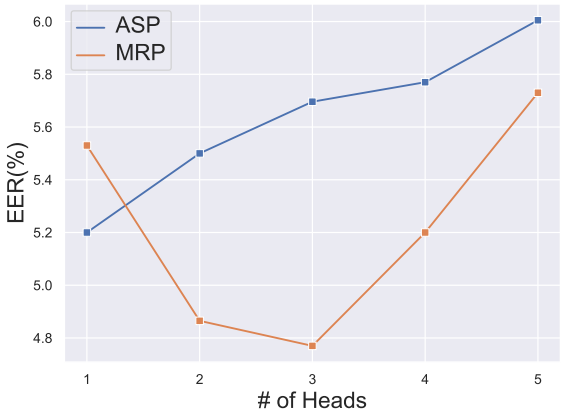

Figure 1: Line plots showing the EER of attentive statistics pooling (ASP) and the proposed mixture representation pooling (MRP) with different numbers of attention heads.

4.2. DNN Training

We experimented with three different networks, namely, the standard x-vector network as in [5], the wide x-vector network, and Densenet121 as mentioned in Section 2.2. The networks were trained using additive margin softmax with a margin of 0.35. The networks were optimized using stochastic gradient descent (SGD). For each mini-batch, we randomly selected 64 utterances from the training set and then randomly cropped 400-ms speech segments from these utterances. We define one epoch as looping through 120,000 such segments. We trained the networks for 320 epochs. The embedding dimension of x-vectors and wide x-vectors are 512. The embedding dimension of DenseNet-121 is 256. The learning rate was set to 0.005 and was divided by 10 at Epoch 80, Epoch 120 , and Epoch 160. All networks were implemented in PyTorch [20].

4.3. Backend Training

We used a standard backend comprised of linear discriminant analysis (LDA), length-normalization, and probabilistic linear discriminant analysis (PLDA). We concatenated speech from the same video session in VoxCeleb1 and VoxCeleb2. These concatenated speech segments were used to train LDA and PLDA models. We used adaptive score normalization [21] in all systems.

4.4. Evaluation

We evaluated the performance of various statistics pooling methods on the VoxCeleb1 test set, VOiCES19 development set, and VOiCES19 evaluation set [22]. We report results in terms of equal error rate (EER) and minimum cost function (DCF) with $P_{target} = 0.01$.

5. Results

5.1. Performance Comparison

We compared the proposed mixture representation pooling (MRP) with the attentive statistics pooling (ASP) and the vanilla single mean and standard deviation pooling. To make a fair comparison with vanilla single mean and standard deviation pooling, we fixed the dimension of the concatenated means and standard deviations in Eq. 13. In other words, increasing the number of attention heads would reduce the dimensionality of the individual mean and standard deviation vectors.

The first three rows of Table 3 show the results of the x-vector network, the wide x-vector network, and Densenet121. Our Densenet121 outperforms the x-vector network and the wide x-vector network significantly. For VoxCeleb1, Densenet121 almost reduces the EER by half when compared with the x-vector network. What's more, our Densenet121 requires even lower flops than the x-vector network, as shown in Table 1. Although the wide x-vector network has more parameters than Densenet121 and the x-vector network, the former only has very small performance gain over the x-vector network, which suggests that increasing the depth is more beneficial than increasing the number of channels.

The last two rows of Table 3 show the results of our Densenet121 with attentive statistics pooling and the proposed method, respectively. The numbers of heads was set to 1 and 3, respectively, as these hyper-parameters lead to the best performance in the two systems. The proposed method outperforms

the attentive statistics pooling in all of the experiments.

5.2. Effect of the Number of Heads

We investigated the effect of the number of attention heads on speaker verification performance. We fixed the dimension of the concatenated means and standard deviations in Eq. 13 to prevent the model from gaining the benefit of having more parameters. The results are shown in Table 4. MRP with number of heads equal to 1 is just the vanilla mean and standard deviation pooling. As can be observed from Table 4, the number of heads influences the performance quite significantly. Actually, attentive statistics pooling with 5 heads performs even worse than without any attention. The performance of the proposed method degrades when the number of heads increases beyond 3, but it is still better than the vanilla single mean and standard deviation pooling.

6. Conclusion

In this paper, we proposed a new way of performing statistics pooling for deep speaker embedding. Our method is inspired by Gaussian mixture models and attention mechanisms. We introduce the concept of mixture representations into statistics pooling by using multi-head attention in a novel way. The experimental results show that our mixture representation pooling is more effective than the previously proposed attentive statistics pooling. More importantly, the proposed method is less prone to overfitting when the number of heads increases. In future work, we will explore how to regularize the attention mechanism when the number of the attention heads increases further.

7. References

[1] Najim Dehak, Patrick Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[2] Yun Lei, Nicolas Scheffer, Luciana Ferrer, and Mitchell McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," in *Proc. ICASSP*, 2014, pp. 1695–1699.

[3] Ehsan Variani, Xin Lei, Erik McDermott, Ignacio Lopez Moreno, and Javier Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *Proc. ICASSP*, 2014, pp. 4052–4056.

[4] Chunlei Zhang and Kazuhito Koishida, "End-to-end text-independent speaker verification with triplet loss on short utterances.," in *Proc. Interspeech*, 2017, pp. 1487–1491.

[5] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *Proc. ICASSP*, 2018, pp. 5329–5333.

[6] Hossein Zeinali, Shuai Wang, Anna Silnova, Pavel Matějka, and Oldřich Plchot, "BUT system description to Voxceleb speaker recognition challenge 2019," *arXiv preprint arXiv:1910.12592*, 2019.

[7] Chunlei Zhang, Kazuhito Koishida, and John HL Hansen, "Text-independent speaker verification based on triplet convolutional neural network embeddings," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 9, pp. 1633–1644, 2018.

[8] Oldřich Plchot, Pavel Matějka, Anna Silnova, Ondřej Novotný, Mireia Diez Sánchez, Johan Rohdin, Ondřej Glembek, Niko Brümmer, Albert Swart, Jesús Jorrín-Prieto, Paola García, Luis Buera, Patrick Kenny, Jahangir Alam, and Gautam Bhattacharya, "Analysis and description of ABC submission to NIST SRE 2016," in *Proc. Interspeech*, 2017, pp. 1348–1352.

[9] Koji Okabe, Takafumi Koshinaka, and Koichi Shinoda, "Attentive statistics pooling for deep speaker embedding," in *Proc. Interspeech*, 2018, pp. 2252–2256.

[10] Yingke Zhu, Tom Ko, David Snyder, Brian Mak, and Daniel Povey, "Self-attentive speaker embeddings for text-independent speaker verification," in *Proc. Interspeech*, 2018, pp. 3573–3577.

[11] Xiaoxiao Miao, Ian McLoughlin, and Yonghong Yan, "A new time-frequency attention mechanism for TDNN and CNN-LSTM-TDNN, with application to language identification," *Proc. Interspeech*, pp. 4080–4084, 2019.

[12] Weicheng Cai, Jinkun Chen, and Ming Li, "Exploring the encoding layer and loss function in end-to-end speaker and language recognition system," in *Proc. Odyssey*, 2018, pp. 74–81.

[13] Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Proc. Interspeech*, 2015.

[14] Simon Prince, Peng Li, Yun Fu, Umar Mohammed, and James Elder, "Probabilistic models for inference about identity," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 1, pp. 144–157, 2011.

[15] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger, "Densely connected convolutional networks," in *Proc. Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4700–4708.

[16] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu, "Additive margin softmax for face verification," *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 926–930, 2018.

[17] Christopher Bishop, "Pattern recognition and machine learning," *Springer, New York*, 2007.

[18] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman, "Voxceleb: A large-scale speaker identification dataset," *Proc. Interspeech*, pp. 2616–2620, 2017.

[19] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman, "Voxceleb2: Deep speaker recognition," *Proc. Interspeech*, pp. 1086–1090, 2018.

[20] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, "Automatic differentiation in PyTorch," 2017.

[21] Pavel Matejka, Ondrej Novotnỳ, Oldrich Plchot, Lukáš Burget, and JH Cernockỳ, "Analysis of score normalization in multilingual speaker recognition," in *Proc. Interspeech*, 2017.

[22] Colleen Richey, Maria A Barrios, Zeb Armstrong, Chris Bartels, Horacio Franco, Martin Graciarena, Aaron Lawson, Mahesh Kumar Nandwana, Allen Stauffer, Julien van Hout, et al., "Voices obscured in complex environmental settings (VOiCES) corpus," *arXiv preprint arXiv:1804.05053*, 2018.