In defence of metric learning for speaker recognition

Joon Son Chung, Jaesung Huh, Seongkyu Mun, Minjae Lee, Hee Soo Heo,
Soyeon Choe, Chiheon Ham, Sunghwan Jung, Bong-Jin Lee, Icksang Han

Naver Corporation, South Korea
joonson.chung@navercorp.com

Abstract

The objective of this paper is 'open-set' speaker recognition of unseen speakers, where ideal embeddings should be able to condense information into a compact utterance-level representation that has small intra-speaker and large inter-speaker distance.

A popular belief in speaker recognition is that networks trained with classification objectives outperform metric learning methods. In this paper, we present an extensive evaluation of most popular loss functions for speaker recognition on the VoxCeleb dataset. We demonstrate that the vanilla triplet loss shows competitive performance compared to classification-based losses, and those trained with our proposed metric learning objective outperform state-of-the-art methods.

Index Terms: speaker recognition, speaker verification, metric learning.

1. Introduction

Research on speaker recognition has a long history and has received an increasing amount of attention in recent years. Large-scale datasets for speaker recognition such as the VoxCeleb [1, 2] and Speakers in the Wild [3] have become freely available, facilitating fast progress in the field.

Speaker recognition can be categorised into closed-set or open-set settings. For closed-set setting, all testing identities are predefined in training set, therefore can be addressed as a classification problem. For open-set setting, the testing identities are not seen during training, which is close to practice. This is a metric learning problem in which voices must be mapped to a discriminative embedding space. The focus of this research, and most others, are on the latter problem.

Pioneering work on speaker recognition using deep neural networks have learnt speaker embeddings via the classification loss [1, 4, 5]. Since then, the prevailing method has been to use softmax classifiers to train the embeddings [6, 7, 8]. While the softmax loss can learn separable embeddings, they are not discriminative enough since it is not explicitly designed to optimise embedding similarity. Therefore, softmax-trained models have often been combined with PLDA [9] back-ends to generate scoring functions [5, 10].

This weakness has been addressed by [11] who have proposed angular softmax (A-Softmax) where cosine similarity is used as logit input to the softmax layer, and a number of works have demonstrated its superiority over vanilla softmax in speaker recognition [6, 7, 8, 12, 13]. Additive margin variants, AM-Softmax [14, 15] and AAM-Softmax [16], have been proposed to increase inter-class variance by introducing a cosine margin penalty to the target logit, and these have been very popular due to their ease of implementation and good performance [17, 18, 19, 20, 21, 22, 23, 24]. However, training with AM-Softmax and AAM-Softmax has proven to be challenging since they are sensitive to the value of scale and margin in the loss function.

Metric learning objectives present strong alternatives to the prevailing classification-based methods, by learning embeddings directly. Since open-set speaker recognition is essentially a metric learning problem, the key is to learn features that have small intra-class and large inter-class distance. Contrastive loss [25] and triplet loss [26] have been demonstrated promising performance on speaker recognition [27, 28] by optimising the distance metrics directly, but these methods require careful pair or triplet selection which can be time consuming and performance sensitive.

Of closest relevance to our work is prototypical networks [29] that learn a metric space in which open-set classification can be performed by computing distances to prototype representations of each class, with a training procedure that mimics the test scenario. The use of multiple negatives helps to stabilise learning since loss functions can enforce that an embedding is far from all negatives in a batch, rather than one particular negative in the case of triplet loss. [30, 31] have adopted the prototypical framework for speaker recognition. Generalised end-to-end loss [32], originally proposed for speaker recognition, is also closely related to this setup.

Comparing different loss functions from prior works can be challenging and unreliable, since speaker recognition systems can vary widely in their design. Popular trunk architectures include TDNN-based systems such as x-vector [5] and its deeper counterparts [8], as well as network architectures from the computer vision community such as the ResNet [33]. A range of encoders have been proposed to aggregate frame-level informations into utterance-level embeddings, from simple averaging [1] to statistical pooling [4, 7] and dictionary-based encodings [17, 34]. [5] has proven that data augmentation can significantly boost speaker recognition performance, but the augmentation methods can range from adding noise [35] to room impulse response (RIR) simulation [36].

Therefore, in order to directly compare a range of loss functions, we conduct over 20,000 GPU-hours of careful experiments while keeping other training details constant. Against popular belief, we demonstrate that the networks trained with vanilla triplet loss show competitive performance compared to most AM-Softmax and AAM-Softmax trained networks, and those trained with our proposed angular objective outperform all comparable methods.

2. Training functions

This section describes the loss functions used in our experiments, including a new angular variant of the prototypical loss.

The code for this paper can be found at:
https://github.com/clovaai/voxceleb_trainer

2.1. Classification objectives

The VoxCeleb2 development set contains $C = 5,994$ speakers or classes. During training, each mini-batch contains N utterances each from different speakers, whose embeddings are $\mathbf{x}_i$ and the corresponding speaker labels are y_i where $1 \leq i \leq N$ and $1 \leq y \leq C$.

Softmax. The softmax loss consists of a softmax function followed by a multi-class cross-entropy loss. It is formulated as:

$$L_{\mathrm{S}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{\mathbf{W}_{y_i}^T\mathbf{x}_i+b_{y_i}}}{\sum_{j=1}^{C}e^{\mathbf{W}_j^T\mathbf{x}_i+b_j}} \tag{1}$$

where $\mathbf{W}$ and b are the weights and bias of the last layer of the trunk architecture, respectively. This loss function only penalises classification error, and does not explicitly enforce intra-class compactness and inter-class separation.

AM-Softmax (CosFace). By normalising the weights and the input vectors, softmax loss can be reformulated such that the posterior probability only relies on cosine of angle between the weights and the input vectors. This loss function, termed by the authors as Normalised Softmax Loss (NSL), is formulated as:

$$L_{\mathrm{N}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{\cos(\theta_{y_i,i})}}{\sum_j e^{\cos(\theta_{j,i})}} \tag{2}$$

where $\cos(\theta_{j,i})$ is the dot product of normalised vector $\mathbf{W}_j$ and $\mathbf{x}_i$.

However, embeddings learned by the NSL are not sufficiently discriminative because the NSL only penalises classification error. In order to mitigate this problem, cosine margin m is incorporated into the equation:

$$L_{\mathrm{C}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cos(\theta_{y_i,i})-m)}}{e^{s(\cos(\theta_{y_i,i})-m)}+\sum_{j\neq y_i}e^{s(\cos(\theta_{j,i}))}} \tag{3}$$

where s is a fixed scale factor to prevent gradient from getting too small in training phase.

AAM-Softmax (ArcFace). This is equivalent to CosFace except that there is additive *angular* margin penalty m between $\mathbf{x}_i$ and $\mathbf{W}_{y_i}$. The additive angular margin penalty is equal to the geodesic distance margin penalty in the normalised hypersphere.

$$L_{\mathrm{A}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cos(\theta_{y_i,i}+m))}}{e^{s(\cos(\theta_{y_i,i}+m))}+\sum_{j\neq y_i}e^{s(\cos(\theta_{j,i}))}} \tag{4}$$

2.2. Metric learning objectives

For metric learning objectives, each mini-batch contains M utterances from each of N different speakers, whose embeddings are $\mathbf{x}_{j,i}$ where $1 \leq j \leq N$ and $1 \leq i \leq M$.

Triplet. Triplet loss minimises the $L2$ distance between an anchor and a positive (same identity), and maximises the distance between an anchor and a negative (different identity).

$$L_{\mathrm{T}} = \frac{1}{N}\sum_{j=1}^{N}\max(0, \|\mathbf{x}_{j,0}-\mathbf{x}_{j,1}\|_2^2 - \|\mathbf{x}_{j,0}-\mathbf{x}_{k\neq j,1}\|_2^2 + m) \tag{5}$$

For our implementation, the negative utterances are sampled from different speakers within the mini-batch and the sample $\mathbf{x}_k$ is selected by the hard negative mining function. This requires $M = 2$ utterances from each speaker.

Prototypical. Each mini-batch contains a support set S and a query set Q. For simplicity, we will assume that the query is M-th utterance from every speaker. Then the prototype (or centroid) is:

$$\mathbf{c}_j = \frac{1}{M-1}\sum_{m=1}^{M-1}\mathbf{x}_{j,m} \tag{6}$$

Squared Euclidean distance is used as the distance metric as proposed by the original paper:

$$\mathbf{S}_{j,k} = \|\mathbf{x}_{j,M} - \mathbf{c}_k\|_2^2 \tag{7}$$

During training, each query example is classified against N speakers based on a softmax over distances to each speaker prototype:

$$L_{\mathrm{P}} = -\frac{1}{N}\sum_{j=1}^{N}\log\frac{e^{\mathbf{S}_{j,j}}}{\sum_{k=1}^{N}e^{\mathbf{S}_{j,k}}} \tag{8}$$

Here, $\mathbf{S}_{j,j}$ is the squared Euclidean distance between the query and the prototype of the same speaker from the support set. The softmax function effectively serves the purpose of hard negative mining, since the hardest negative would most affect the gradients. The value of M is typically chosen to match the expected situation at test-time, *e.g.* $M = 5 + 1$ for 5-shot learning, so that the prototype is composed of five different utterances. In this way, the task in training exactly matches the task in test scenario.

Generalised end-to-end (GE2E). In GE2E training, every utterance in the batch except the query itself is used to form centroids. As a result, the centroid that is of the same class as the query is computed from one fewer utterance than centroids of other classes. They are defined as:

$$\mathbf{c}_j = \frac{1}{M}\sum_{m=1}^{M}\mathbf{x}_{j,m} \tag{9}$$

$$\mathbf{c}_j^{(-i)} = \frac{1}{M-1}\sum_{\substack{m=1\\m\neq i}}^{M}\mathbf{x}_{j,m} \tag{10}$$

The similarity matrix is defined as scaled cosine similarity between the embeddings and all centroids:

$$\mathbf{S}_{j,i,k} = \begin{cases} w\cdot\cos(\mathbf{x}_{j,i}, \mathbf{c}_j^{(-i)}) + b & \text{if} \quad k = j \\ w\cdot\cos(\mathbf{x}_{j,i}, \mathbf{c}_k) + b & \text{otherwise.} \end{cases} \tag{11}$$

where $w > 0$ and b are learnable scale and bias. The final GE2E loss is defined as:

$$L_{\mathrm{G}} = -\frac{1}{N}\sum_{j,i}\log\frac{e^{\mathbf{S}_{j,i,j}}}{\sum_{k=1}^{N}e^{\mathbf{S}_{j,i,k}}} \tag{12}$$

Angular Prototypical. The angular prototypical loss uses the same batch formation as the original prototypical loss, reserving one utterance from every class as the query. This has advantages over GE2E-like formation since every centroid is made from the same number of utterances in the support set, therefore it is possible to exactly mimic the test scenario during training.

We use a cosine-based similarity metric with learnable scale and bias, as in the GE2E loss.

$$\mathbf{S}_{j,k} = w\cdot\cos(\mathbf{x}_{j,M}, \mathbf{c}_k) + b \tag{13}$$

Using the angular loss function introduces scale invariance, improving the robustness of objective against feature variance and demonstrating more stable convergence [37].

The resultant objective is the same as the original prototypical loss, Equation 8.

3. Experiments

In this section we describe the experimental setup, which is identical across all objectives described in Section 2.

3.1. Input representations

During training, we use a fixed length 2-second temporal segment, extracted randomly from each utterance. Spectrograms are extracted with a hamming window of width 25ms and step 10ms. For the Thin ResNet model, the 257-dimensional raw spectrograms are used as the input to the network. For the VGG-M-40 and the Fast ResNet, 40-dimensional Mel filterbanks are used as the input. Mean and variance normalisation (MVN) is performed by applying instance normalisation [38] to the network input. Since the VoxCeleb dataset consists mostly of continuous speech, voice activity detection (VAD) is not used in training and testing.

3.2. Trunk architecture

Experiments are performed on the trunk architectures described below. The first two are identical to the models used and described in [39], while the last is a variation of the ResNet model to reduce computation requirement. The architectures are compared in Table 1.

VGG-M-40. The VGG-M model has been proposed for image classification [40] and adapted for speaker recognition by [1]. The network is known for high efficiency and good classification performance. VGG-M-40 is a modification of the network proposed by [1] to take 40-dimensional filterbanks as inputs instead of the 513-dimensional spectrogram. The temporal average pooling (TAP) layer takes the mean of the features along the time domain in order to produce utterance-level representation.

Thin ResNet-34. Residual networks [33] are widely used in image recognition and have recently been applied to speaker recognition [2, 34, 17, 39]. Thin ResNet-34 is the same as the original ResNet with 34 layers, except using only one-quarter of the channels in each residual block in order to reduce computational cost. The model only has 1.4 million parameters compared to 22 million of the standard ResNet-34. Self-attentive pooling (SAP) [34] is used to aggregate frame-level features into utterance-level representation while paying attention to the frames that are more informative for utterance-level speaker recognition. Thin ResNets of [34] and [39] differ slightly in their implementation details, but in our experiments we use that of [39].

Fast ResNet-34. The number and size of filters are identical to the Thin ResNets of [34, 39], but the input dimensions are smaller than [39] and the strides are earlier than [34] in order to reduce computational requirements. Due to space constraints, the exact specification can be found in the accompanying code. The performance is on par with both Thin ResNet models, while the computation cost is less than half of those models.

Network	Params	MACs
VGG-M-40 [39]	4.0M	0.53G
Thin ResNet-34 [39]	1.4M	0.99G
Thin ResNet-34 [34]	1.4M	0.93G
Fast ResNet-34	1.4M	0.45G

Table 1: *Network statistics. Multiplyaccumulate operations (MACs) are measured for a 2-second input.*

3.3. Implementation details

Datasets. The network is trained on the development set of VoxCeleb2 [2] and evaluated on test set of VoxCeleb1 [1]. Note that the development set of VoxCeleb2 is completely disjoint from the VoxCeleb1 dataset (*i.e.* no speakers in common).

Training. Our implementation is based on the PyTorch framework [41] and trained on the NAVER Smart Machine Learning (NSML) platform [42]. The models are trained using a NVIDIA V100 GPU with 32GB memory for 500 epochs. For each epoch, we randomly sample a maximum of 100 utterances from each of the 5,994 identities to reduce class imbalance. We use the Adam optimizer with an initial learning rate of 0.001 decreasing by 5% every 10 epochs. For metric learning objectives, we use the largest batch size that fits on a GPU. For classification objectives, we use a fixed batch size of 200. The training takes approximately one day for the VGG-M-40 model, two days for the Fast ResNet model and five days for the Thin ResNet model.

All experiments were repeated independently three times in order to minimise the effect of random initialisation, and we report mean and standard deviation of the experiments.

Data augmentation. No data augmentation is performed during training, apart from the random sampling.

Curriculum learning. The AAM-Softmax loss function demonstrates unstable convergence from random initialisation with larger values of m such as 0.3. Therefore, we start training the model with $m = 0.1$ and increase it to $m = 0.3$ after 100 epochs. This strategy is labelled *Curriculum* in Table 2.

Similarly, the triplet loss can cause models to diverge if the triplets are too difficult early in the training. We only enable hard negative mining after 100 epochs, at which point the network only sees the most difficult 1% of the negatives.

3.4. Evaluation

Evaluation protocol. The trained networks are evaluated on the VoxCeleb1 test set. We sample ten 4-second temporal crops at regular intervals from each test segment, and compute the similarities between all possible combinations ($10 \times 10 = 100$) from every pair of segments. The mean of the 100 similarities is used as the score. This protocol is in line with that used by [2, 39].

Results. The results are given in Table 2. It can be seen that the performance of networks trained with AM-Softmax and AAM-Softmax loss functions can be very sensitive to the value of margin and scale set during training. We iterate over many combinations of m and s to find the optimal value. The model trained with the most common setting (AM-Softmax with $m = 0.3$ and $s = 30$) is outperformed by the vanilla triplet loss.

Generalised end-to-end and prototypical losses show improvements over the triplet loss by using multiple negatives in training. The prototypical networks perform best when the value of M matches the test scenario, removing the necessity for hyperparameter optimisation. The performance of the model trained with the proposed angular objective exceeds that of all classification-based and metric learning methods.

There are a substantial number of recent works on the VoxCeleb2 dataset, but we do not compare to these in the table, since the goal of this work is to compare the performance of different loss functions under identical conditions. However,

Objective	Hyperparameters	VGG-M-40	Thin ResNet-34	Fast ResNet-34
Softmax	-	10.14 ± 0.20	5.82 ± 0.47	6.46 ± 0.06
AM-Softmax [14]	$m = 0.1, s = 15$	4.86 ± 0.14	2.81 ± 0.08	2.77 ± 0.03
	$m = 0.2, s = 15$	5.14 ± 0.13	2.85 ± 0.07	3.05 ± 0.03
	$m = 0.3, s = 15$	5.24 ± 0.08	3.08 ± 0.05	3.08 ± 0.08
	$m = 0.4, s = 15$	5.22 ± 0.15	3.09 ± 0.06	3.25 ± 0.09
	$m = 0.1, s = 30$	$\mathbf{4.76 \pm 0.10}$	2.59 ± 0.09	$\mathbf{2.41 \pm 0.01}$
	$m = 0.2, s = 30$	4.88 ± 0.03	$\mathbf{2.40 \pm 0.07}$	2.43 ± 0.05
	$m = 0.3, s = 30$	5.19 ± 0.08	2.71 ± 0.10	2.52 ± 0.04
	$m = 0.4, s = 30$	5.35 ± 0.06	2.81 ± 0.10	2.67 ± 0.05
	$m = 0.1, s = 50$	5.45 ± 0.06	2.99 ± 0.04	2.73 ± 0.07
	$m = 0.2, s = 50$	5.28 ± 0.07	2.60 ± 0.10	2.51 ± 0.01
	$m = 0.3, s = 50$	5.62 ± 0.09	2.80 ± 0.09	2.53 ± 0.06
	$m = 0.4, s = 50$	5.91 ± 0.12	2.96 ± 0.08	2.69 ± 0.07
AAM-Softmax [16]	$m = 0.1, s = 15$	4.81 ± 0.03	2.78 ± 0.04	2.80 ± 0.11
	$m = 0.2, s = 15$	4.88 ± 0.08	2.88 ± 0.09	2.98 ± 0.05
	$m = 0.3, s = 15$	14.90 ± 0.16	3.16 ± 0.05	14.98 ± 0.20
	$\rightarrow$ Curriculum	5.00 ± 0.05	2.91 ± 0.08	3.04 ± 0.06
	$m = 0.1, s = 30$	4.67 ± 0.06	2.60 ± 0.07	2.48 ± 0.02
	$m = 0.2, s = 30$	$\mathbf{4.64 \pm 0.04}$	$\mathbf{2.36 \pm 0.04}$	2.38 ± 0.01
	$m = 0.3, s = 30$	13.25 ± 0.07	10.55 ± 0.33	11.35 ± 0.18
	$\rightarrow$ Curriculum	4.69 ± 0.02	2.39 ± 0.05	$\mathbf{2.37 \pm 0.02}$
	$m = 0.1, s = 50$	5.27 ± 0.03	2.88 ± 0.05	2.71 ± 0.07
	$m = 0.2, s = 50$	4.96 ± 0.03	2.50 ± 0.05	2.49 ± 0.04
	$m = 0.3, s = 50$	10.42 ± 0.12	8.79 ± 0.21	9.49 ± 0.25
	$\rightarrow$ Curriculum	4.86 ± 0.11	2.41 ± 0.08	2.42 ± 0.06
Triplet [26]	$m = 0.1$, CHNM	4.86 ± 0.15	$\mathbf{2.53 \pm 0.10}$	2.73 ± 0.03
	$m = 0.2$, CHNM	$\mathbf{4.67 \pm 0.06}$	2.60 ± 0.02	$\mathbf{2.71 \pm 0.06}$
	$m = 0.3$, CHNM	4.84 ± 0.13	2.66 ± 0.03	2.85 ± 0.04
	$m = 0.4$, CHNM	4.84 ± 0.08	2.76 ± 0.10	2.96 ± 0.07
GE2E [32]	$M = 2$	4.60 ± 0.04	2.56 ± 0.08	2.51 ± 0.07
	$M = 3$	$\mathbf{4.40 \pm 0.08}$	$\mathbf{2.52 \pm 0.07}$	$\mathbf{2.37 \pm 0.10}$
	$M = 4$	4.49 ± 0.05	2.59 ± 0.12	2.59 ± 0.08
	$M = 5$	4.69 ± 0.09	2.78 ± 0.09	2.66 ± 0.02
	$M = 10$	5.53 ± 0.04	3.68 ± 0.08	3.55 ± 0.05
Prototypical [29]	$M = 2$	$\mathbf{4.59 \pm 0.02}$	$\mathbf{2.34 \pm 0.08}$	$\mathbf{2.32 \pm 0.02}$
	$M = 3$	4.73 ± 0.11	2.54 ± 0.07	2.39 ± 0.05
	$M = 4$	4.99 ± 0.19	2.83 ± 0.04	2.89 ± 0.04
	$M = 5$	5.34 ± 0.03	3.33 ± 0.11	3.21 ± 0.01
Angular Prototypical	$M = 2$	$\mathbf{4.29 \pm 0.07}$	$\mathbf{2.21 \pm 0.03}$	$\mathbf{2.22 \pm 0.05}$
	$M = 3$	4.30 ± 0.05	2.45 ± 0.07	2.40 ± 0.04
	$M = 4$	4.53 ± 0.03	2.75 ± 0.06	2.60 ± 0.02
	$M = 5$	4.73 ± 0.01	3.00 ± 0.11	2.90 ± 0.11

Table 2: *Equal Error Rates (EER, %) on the VoxCeleb1 test set. We report the mean and standard deviation of the repeated experiments.* **CHNM***: Curriculum Hard Negative Mining.*

Objective	Hyperparameters	200	400	600	800
AM-Softmax	$m = 0.2, s = 30$	2.40 ± 0.07	2.53 ± 0.08	2.49 ± 0.11	2.57 ± 0.07
Prototypical	$M = 2$	2.42 ± 0.04	2.40 ± 0.07	2.34 ± 0.05	2.34 ± 0.08
Angular Prototypical	$M = 2$	2.37 ± 0.07	2.31 ± 0.05	2.32 ± 0.09	2.21 ± 0.03

Table 3: *Effect of training batch size on test performance. Equal Error Rates (EER, %) using the Thin ResNet-34 architecture on the VoxCeleb1 test set. We report the mean and standard deviation of the repeated experiments.*

we are unaware of any work that outperforms our method with a similar number of network parameters.

Batch size. The effect of batch size on various loss functions is shown in Table 3. We observe that a bigger batch size has a positive effect on performance for metric learning methods, which can be explained by the ability to sample harder negatives within the batch. We make no such observation for the network trained with classification loss.

4. Conclusions

In this paper, we have presented a case for metric learning in speaker recognition. Our extensive experiments indicate that the GE2E and prototypical networks show superior performance to the popular classification-based methods. We also propose an angular variant of the prototypical networks that outperforms all existing training functions. Finally, we release a flexible PyTorch trainer for large-scale speaker recognition that can be used to facilitate further research in the field.

5. References

[1] A. Nagrani, J. S. Chung, and A. Zisserman, "VoxCeleb: a large-scale speaker identification dataset," in *INTERSPEECH*, 2017.

[2] J. S. Chung, A. Nagrani, and A. Zisserman, "VoxCeleb2: Deep speaker recognition," in *INTERSPEECH*, 2018.

[3] M. McLaren, L. Ferrer, D. Castan, and A. Lawson, "The speakers in the wild (SITW) speaker recognition database," in *INTERSPEECH*, 2016.

[4] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep neural network embeddings for text-independent speaker verification." in *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 *Proc. ICASSP*. IEEE, 2018, pp. 5329–5333.

[6] M. Ravanelli and Y. Bengio, "Speaker recognition from raw waveform with sincnet," in *IEEE Spoken Language Technology Workshop*. IEEE, 2018, pp. 1021–1028.

[7] K. Okabe, T. Koshinaka, and K. Shinoda, "Attentive statistics pooling for deep speaker embedding," in *INTERSPEECH*, 2018.

[8] D. Snyder, D. Garcia-Romero, G. Sell, A. McCree, D. Povey, and S. Khudanpur, "Speaker recognition for multi-speaker conversations using x-vectors," in *Proc. ICASSP*. IEEE, 2019, pp. 5796–5800.

[9] S. Ioffe, "Probabilistic linear discriminant analysis," in *Proc. ECCV*. Springer, 2006, pp. 531–542.

[10] S. Ramoji, V. Krishnan, P. Singh, S. Ganapathy *et al.*, "Pairwise discriminative neural plda for speaker verification," *arXiv preprint arXiv:2001.07034*, 2020.

[11] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," in *Proc. CVPR*, 2017, pp. 212–220.

[12] J. Villalba, N. Chen, D. Snyder, D. Garcia-Romero, A. McCree, G. Sell, J. Borgstrom, F. Richardson, S. Shon, F. Grondin *et al.*, "State-of-the-art speaker recognition for telephone and video speech: the jhu-mit submission for nist sre18," *Interspeech*, pp. 1488–1492, 2019.

[13] D. Snyder, J. Villalba, N. Chen, D. Povey, G. Sell, N. Dehak, and S. Khudanpur, "The jhu speaker recognition system for the voices 2019 challenge," in *Interspeech*, 2019, pp. 2468–2472.

[14] 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.

[15] 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 *Proc. CVPR*, 2018, pp. 5265–5274.

[16] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proc. CVPR*, 2019, pp. 4690–4699.

[17] W. Xie, A. Nagrani, J. S. Chung, and A. Zisserman, "Utterance-level aggregation for speaker recognition in the wild," in *Proc. ICASSP*, 2019.

[18] M. Hajibabaei and D. Dai, "Unified hypersphere embedding for speaker recognition," *arXiv preprint arXiv:1807.08312*, 2018.

[19] Y. Liu, L. He, and J. Liu, "Large margin softmax loss for speaker verification," in *INTERSPEECH*, 2019.

[20] D. Garcia-Romero, D. Snyder, G. Sell, A. McCree, D. Povey, and S. Khudanpur, "X-vector dnn refinement with full-length recordings for speaker recognition," in *Interspeech*, 2019, pp. 1493–1496.

[21] H. Zeinali, S. Wang, A. Silnova, P. Matějka, and O. Plchot, "BUT system description to VoxCeleb Speaker Recognition Challenge 2019," *arXiv preprint arXiv:1910.12592*, 2019.

[22] C. Luu, P. Bell, and S. Renals, "Channel adversarial training for speaker verification and diarization," *arXiv preprint arXiv:1910.11643*, 2019.

[23] ——, "Dropclass and dropadapt: Dropping classes for deep speaker representation learning," *arXiv preprint arXiv:2002.00453*, 2020.

[24] X. Xiang, S. Wang, H. Huang, Y. Qian, and K. Yu, "Margin matters: Towards more discriminative deep neural network embeddings for speaker recognition," *arXiv preprint arXiv:1906.07317*, 2019.

[25] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Proc. CVPR*, vol. 1. IEEE, 2005, pp. 539–546.

[26] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proc. CVPR*, 2015.

[27] C. Zhang, K. Koishida, and J. H. 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.

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

[29] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *NIPS*, 2017, pp. 4077–4087.

[30] J. Wang, K.-C. Wang, M. T. Law, F. Rudzicz, and M. Brudno, "Centroid-based deep metric learning for speaker recognition," in *Proc. ICASSP*. IEEE, 2019, pp. 3652–3656.

[31] P. Anand, A. K. Singh, S. Srivastava, and B. Lall, "Few shot speaker recognition using deep neural networks," *arXiv preprint arXiv:1904.08775*, 2019.

[32] L. Wan, Q. Wang, A. Papir, and I. L. Moreno, "Generalized end-to-end loss for speaker verification," in *Proc. ICASSP*. IEEE, 2018, pp. 4879–4883.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016.

[34] W. Cai, J. Chen, and M. Li, "Exploring the encoding layer and loss function in end-to-end speaker and language recognition system," in *Speaker Odyssey*, 2018.

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

[36] J. B. Allen and D. A. Berkley, "Image method for efficiently simulating small-room acoustics," *The Journal of the Acoustical Society of America*, vol. 65, no. 4, pp. 943–950, 1979.

[37] J. Wang, F. Zhou, S. Wen, X. Liu, and Y. Lin, "Deep metric learning with angular loss," in *Proc. ICCV*, 2017, pp. 2593–2601.

[38] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022*, 2016.

[39] J. S. Chung, J. Huh, and S. Mun, "Delving into VoxCeleb: environment invariant speaker recognition," in *Speaker Odyssey*, 2020.

[40] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," in *Proc. BMVC.*, 2014.

[41] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *NIPS*, 2019, pp. 8024–8035.

[42] N. Sung, M. Kim, H. Jo, Y. Yang, J. Kim, L. Lausen, Y. Kim, G. Lee, D. Kwak, J.-W. Ha *et al.*, "Nsml: A machine learning platform that enables you to focus on your models," *arXiv preprint arXiv:1712.05902*, 2017.