Triplet-Center Loss Based Deep Embedding Learning Method for Speaker Verification

Yiheng Jiang*, Yan Song*, Jie Yan*, Lirong Dai*, Ian McLoughlin†
* National Engineering Laboratory of Speech and Language Information Processing,
University of Science and Technology of China, Hefei, China
E-mail: {jiangyh, yanjie17}@mail.ustc.edu.cn, {songy, lrdai}@ustc.edu.cn
† School of Computing, University of Kent, Medway, UK
E-mail: ivm@kent.ac.uk

***Abstract*—In this work, we introduce an effective loss function, *i.e.*, triplet-center loss, to improve the performance of deep embedding learning methods for speaker verification (SV). The triplet-center loss is combination of triplet loss and center loss so that it shares superiorities of these two loss functions. Comparing with the widely used softmax loss, the main advantage of triplet-center loss is that it learns a center for each class, and it requires distances between samples and centers from the same class are closer than those from different classes. To evaluate the performance of triplet-center loss, we conduct extensive experiments on noisy and unconstrained dataset, *i.e.*, Voxceleb. The results show that triplet-center loss significantly improves the performance of SV. Specifically, it reduces equal error rate (EER) from softmax loss by 11.6%, 10.4% in cosine scoring and PLDA backend, respectively.**

I. Introduction

Speaker Verification (SV) is the task of recognizing the identity of a speaker based on one or several given segments of speech from this speaker. A common SV system mainly consists of two steps: **a)** front-end feature learning to extract low-dimensional speaker embeddings, and **b)** back-end embeddings' similarity calculating to confirm the identity of the speaker.

For decades, the most popular method for SV was i-vector system [1] with Probabilistic Linear Discriminative Analysis (PLDA) [2], [3] modeling. The i-vector based method is trained on unsupervised fashion, and the PLDA is employed to model features' channel variability [4].

Recently, more attention of SV has been moved to deep embedding learning methods. Thanks to supervised training process, the deep learning systems such as d-vector [5] and x-vector [6] have shown great potential, especially in the short duration case. Some kinds of neural network architectures have also emerged in SV field, including time-delay neural network (TDNN) [6], [7], convolutional neural network (CNN) [8], and long short-term memory network (LSTM) [9]. All these architectures commonly used softmax loss to learn the parameters.

Considering that SV task is an open-set recognition problem, which means there is no overlap of speakers between the training and test set. Therefore, SV is closely related to the metric learning problem, where ideal speaker embeddings should be compact in the same class and be discriminative in different classes. Although softmax loss is very suitable for classification in close-set problem, it can not explicitly encourage the discriminative learning of features in open-set problem.

To overcome the weakness of softmax loss, some other loss functions were introduced. J. S. Chung *et al* [10] pre-trained the model using softmax loss at first, and then fine-tuned it using contrastive loss. In [11], [12], angular softmax (A-softmax) loss was introduced, which required the angles between each sample and its ground truth class center to be m (m is margin) times smaller than that of wrong classes. Furthermore, M. Hajibabaei *et al* [13] applied additive margin softmax loss (AM-softmax) to SV task, in which the cosine similarity was incorporated with an additive margin to force samples from the same speaker to be closer than those from different speakers.

In the last few years, triplet loss for SV was first presented in [14], [15], it encourages to find an embedding space where the distances between embeddings from the same class (*i.e.*, *anchor* and *positive* samples) are smaller than those from different classes (*i.e.*, *anchor* and *negative* samples) by at least a margin m. However, triplet loss may cause problems of much time-consuming and dramatic data expansion when constructing triplets [16]. Center loss for SV was introduced in [11], [17], it learns a center for each class and focuses on reducing the distances between samples and centers from the same class, thus it compacts intra-class variability. However, center loss has no effect on enlarging inter-class distances.

In this paper, we introduce a novel triplet-center loss [16], which can be regarded as union of triplet loss and center loss, to improve SV performance. The triplet-center loss learns a center for each class, such that samples are pulled close to the corresponding center of the same class and meanwhile pushed away from the other nearest center of different classes. Different from triplet loss, triplet-center loss avoids to construct and select triplets and it has no data expansion. In addition, comparing with center loss which only focuses on compacting the intra-class variability, triplet-center loss also considers to enlarge the distances between different classes. Extensive experiments on Voxceleb [18] prove that triplet-center loss is an effective criterion for SV. Particularly, results show that it obtains 11.6%, 10.4% relative improvements in terms of equal error rate (EER) in cosine scoring and PLDA

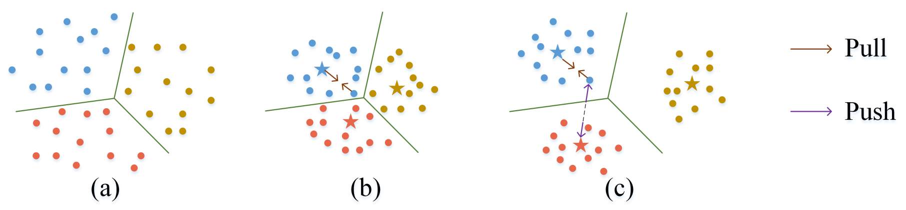


Fig. 1. A simple illustration of results caused by (a) softmax loss, (b) center loss + softmax loss, (c) triplet-center loss + softmax loss. Ideally, the softmax loss aims to find a decision boundary of different classes. The center loss pulls samples close to their corresponding center which belongs to the same class. The triplet-center loss not only pulls samples close to the center of the same class, but also pushes them away from the other nearest center of different classes.

backend, respectively.

II. Proposed Method

Softmax loss is commonly exploited to achieve the classification in deep neural network. However, as illustrated in Fig. 1 (a), it might not be suitable for open-set problem like SV, since it only focuses on finding a decision boundary of different classes without considering the intra-class compactness of features. Here we first introduce triplet loss and center loss, and then derive our triplet-center loss to address this problem.

A. Triplet Loss

Triplet loss was proposed in FaceNet [19], and in [14], [15], it was first applied to SV. Assuming that speaker embedding is represented by $f(x_i) \in \mathbf{R}^d$, which means the i-th utterance x_i in dataset is mapped to d-dimensional vector $f(x_i)$ by the neural network. Here, triplet loss makes that an embedding $f(x_i^a)$ (*anchor*) is closer to other embedding $f(x_i^p)$ (*positive*) of the same speaker, meanwhile keeps $f(x_i^a)$ away from embedding $f(x_i^n)$ (*negative*) of different speaker. Specifically, triplet loss can be formulated as:

$$L_{trip} = \sum_{i=1}^{N} \max(0, m + \| f(x_i^a) - f(x_i^p) \|_2^2 - \| f(x_i^a) - f(x_i^n) \|_2^2) \tag{1}$$

where N is number of triplets in the training set, $\| \cdot \|_2^2$ is euclidean distance, m is a fixed element which requires distance between $f(x_i^a)$ and $f(x_i^p)$ smaller than that between $f(x_i^a)$ and $f(x_i^n)$ by at least a margin m. In other words, the margin m enforce a distance between different classes, leading to more discriminative deep embeddings. Note that although we require euclidean distance, it could also be defined as other forms like cosine distance.

In general, there exists too many triplets to computing in the whole training set, thus it is essential to design a strategy for triplets selection so that the model could converge without consuming too much time. Existing way of selection strategy is hard mining [19], [20]. For each anchor, the hard mining strategy chooses the "hardest" positive and negative samples from within the mini-batch or the subset of data to make up a triplet. Therefore, the performance of triplet loss highly dependents on the mining strategy. Nevertheless, how to define better "hard triplets" is still an open problem. All these factors make triplet loss hard to train. To avoid this limitation, we will combine it with center loss (see Section II-B) to propose a novel triplet-center loss.

B. Center Loss

Center loss was proposed in [21]. It learns a center for each class. As illustrated in Fig. 1 (b), it pulls samples close to their corresponding center which belongs to the same class. Thus the learning goal is to minimize the distances between samples and centers from the same class. The center loss can be defined as:

$$L_{cen} = \frac{1}{2} \sum_{i=1}^{M} \| f(x_i) - c_{y_i} \|_2^2 \tag{2}$$

where $c_{y_i} \in \mathbf{R}^d$ represents the d-dimensional center of class y_i, and $\| \cdot \|_2^2$ denotes the euclidean distance. Particularly, the centers are updated based on mini-batch with batch-size $= M$. Note that the center loss can not be used independently, otherwise the deeply learned features and centers will degrade to zeros (at this point, the center loss is very small) [21]. The reason is that center loss only compacts the intra-class variability without considering inter-class separability. Hence the model must be trained based on the combination of center loss and softmax loss.

C. Proposed Triplet-center Loss

In order to overcome the limitations of triplet loss (it is complex to construct "good" triplets) and center loss (it dose not consider inter-class separability), meanwhile, to exploit the advantages of these two loss functions, we derive a novel triplet-center loss for SV task.

Assuming that training set is $\{x_i, y_i\}_{i=1}^N$, where N is total number of samples, x_i is the i-th sample and $y_i \in \{1, 2, \cdots, Y\}$ is corresponding label of x_i, Y is number of classes. Let $f(x_i) \in \mathbf{R}^d$ represents d-dimensional embedding of x_i, and let $C = \{c_1, c_2, \cdots, c_Y\}$ denotes the learnable centers where $c_y \in \mathbf{R}^d$ is center vector of y-th class. Then the triplet-center loss can be represented as:

$$L_{tc} = \sum_{i=1}^{M} \max(0, m + \| f(x_i) - c_{y_i} \|_2^2 - \min_{j \neq y_i} \| f(x_i) - c_j \|_2^2) \tag{3}$$

where M is batch-size, $\| \cdot \|_2^2$ denotes the euclidean distance, and m is margin. As illustrated in Fig. 1 (c), triplet-center loss wants to pull embedding $f(x_i)$ to be closer to its corresponding center c_{y_i} and push $f(x_i)$ away from the other nearest *negative center* c_j ($j \neq y_i$). The distances between embeddings and corresponding centers are smaller than those from different classes by at least a margin m. Here, the triplet can be expressed as $(f(x_i), c_{y_i}, c_j)$, thus there are only N triplets in total. Naturally, compared with triplet loss which has N^3 triplets, triplet-center loss avoids the complexity of constructing triplets.

On the other hand, softmax loss aims to map samples to a class-separable space, while triplet-center loss focuses on applying metric learning to embeddings directly. Considering that the centers are randomly initialized, and they are updated once in each mini-batch instead of in the whole training data. The training process is not stable enough if using triplet-center loss independently. Thus we also utilize softmax loss to be a guider for better converging of triplet-center loss. The joint version of softmax loss and triplet-center loss can be presented as:

$$L_{stc} = L_s + \lambda L_{tc} \tag{4}$$

where L_s is softmax loss, and weight λ is used for balancing these two loss functions. It is worth noting that the joint loss L_{stc} degenerates softmax loss when λ is set to be 0.

III. Experimental Setup

A. Datasets and Acoustic Features

Dataset: We use Voxceleb to train model and evaluate performance. There are 1,251 speakers in total. We use development set of 1,211 speakers for training and test set of 40 speakers for testing. Due to sampled from the real world, Voxceleb includes background noise such as laughter and music. It is therefore challenging to design a robust model and an appropriate loss function to handle this complex environment. To compare with other results on Voxceleb, no data augmentation strategy is adopted to the dataset.

Feature: The feature extraction process follows Kaldi toolkit [22]. The dimension of feature is 41, which includes log mel filterbank coefficients of 40 dimensions and energy of 1 dimension. All features are obtained from 25ms windows with 10ms shift between frames. Furthermore, we apply mean-normalization over a sliding window of 3s, and use voice activity detection (VAD) to remove silent segments.

B. Model Configuration

The features from the training dataset are randomly cropped to lengths of 2-4s. Then we group features which have the same length into a mini-batch with batch-size = 128. All neural networks are implemented by PyTorch [23]. The model is optimized using stochastic gradient descent (SGD) [24] with momentum of 0.95 and weight decay of 5e-4. Our model will be trained for 192 epochs in total. The learning rate gradually decreases from 1e-2 to 1e-5 in the whole training process.

The backbone of our network is ResNet-34 [25]. Based on this network, Cai *et al* [26] proved that applying length normalization to deep embeddings could improve SV performance. The implementation of length normalization is to compute L2-norm of embeddings and then multiply a scale parameter α. Hence we follow them that we also adopt length normalization (α = 12 in all our experiments) to embeddings. In addition, we apply statistics pooling [7] to transform variable-length representations into the fixed-length vectors. The statistics pooling layer receives the output of the final convolutional layer and calculate mean and standard deviation to be the statistics information. It then concatenates these statistics and feed them into next embedding-extracting layer. The detail of our model's architecture is reported in Table I, where L is variable-length data frames.

TABLE I
Architecture of ResNet-34 in our experiments.

Layer	Channels	Blocks	Down-sample	Output-size
Conv1	16	-	$\times$	$64\times L$
Stage1	16	3	$\times$	$64\times L$
Stage2	32	4	✓	$32\times \frac{L}{2}$
Stage3	64	6	✓	$16\times \frac{L}{4}$
Stage4	128	3	✓	$8\times \frac{L}{8}$
Statistics pooling				256
Embedding (FC)				128
Length normalization (L2-norm + scale)				128
Classifier (FC)				speaker categories

After model training finished, the 128-dimensional speaker embeddings are extracted from an FC layer. At last, we compute the similarity between embeddings using cosine distance and PLDA.

C. Loss Function Configuration

Our triplet-center loss is trained combining with softmax loss, where weight λ = 0.01, margin m = 5 and the learning rate of center in triplet-center loss is 0.1. In order to compare the advantage of our proposed method, We also conduct experiments on triplet loss and center loss. In triplet loss experiment, we first pretrain the model using softmax loss and then finetune it by triplet loss. More specifically, we apply hard mining strategy [20] to construct triplets, *i.e.*, each mini-batch (batch-size = 128) includes 32 speakers and each speaker has 4 speech segments. In this case, each mini-batch could construct 128 "hard triplets". In center loss experiment, it is very essential to combine center loss with softmax loss for model training, the form of this combination is like Eq. (4) where λ (*i.e.*, the weight of center loss) is 0.01. The learning rate of center in center loss is 0.1.

D. Ramp-up of Triplet-center Loss

Our experiments show that the ramp-up of the triplet-center loss component could slightly improve the system performance. Hence in all experiments, we ramp up the triplet-center loss weight λ during the first 31 epochs using a Gaussian ramp-up curve:

$$\lambda(t) = a * e^{-5(1-\frac{t}{T})^2} \tag{5}$$

TABLE II
PERFORMANCE OF DIFFERENT LOSS FUNCTIONS BASED ON RESNET-34 (LOWER IS BETTER).

ID	Loss function	Length normalization	Pooling	Cosine distance		PLDA backend	
				EER (%)	$\mathrm{minDCF}_{0.01}$	EER (%)	$\mathrm{minDCF}_{0.01}$
1	i-vector [26]	-	-	13.80	0.681	5.48	0.488
2	Center loss [11]	×	LDE	4.98	0.496	4.87	0.632
3	Center loss [11]	×	Average pooling	4.75	0.522	4.59	0.516
4	A-softmax loss [11]	×	LDE	4.56	0.441	4.48	0.576
5	A-softmax loss [11]	×	Average pooling	5.27	0.439	4.46	0.577
6	Softmax loss [11]	×	LDE	5.21	0.516	5.07	0.519
7	Softmax loss [11]	×	Average pooling	5.48	0.553	5.21	0.545
8	Softmax loss	✓	Average pooling	5.18	0.452	4.71	0.512
9	Triplet-center loss	✓	Average pooling	**4.34**	**0.429**	**4.16**	**0.484**
10	Softmax loss	✓	Statistics pooling	4.83	0.449	4.42	0.463
11	Triplet loss	✓	Statistics pooling	4.61	0.495	4.21	0.477
12	Center loss	✓	Statistics pooling	4.52	0.436	4.30	0.454
13	Triplet-center loss	✓	Statistics pooling	**4.27**	**0.384**	**3.96**	**0.454**

where t is epoch from 0 to 30, $\lambda(t)$ is the value of λ in t-th epoch. T = 30 is the maximum epoch during ramp-up period, and a is the maximum value of λ.

IV. RESULTS

A. Main Results

The performance is evaluated in terms of equal error rate (EER) and minimum of detection cost function ($\mathrm{minDCF}_{0.01}$). The main results are reported in Table II. The i-vector system is a baseline, where 2048-components Gaussian Mixture Model-Universal Background Model (UBM-GMM) is trained, and 400-dimensional i-vector is extracted.

The experiments of ID from 2 to 9 are comparison of our triplet-center loss with other state-of-the-art loss functions, where LDE is a novel pooling method proposed in [11]. Comparing result of ID 9 with ID 8, our triplet-center loss achieves a notable progress over softmax loss. And comparing result of ID 9 with other loss functions reported in [11], triplet-center loss also keeps obvious advantage. On the other hand, comparison between result of ID 8 and ID 7, we could discover that length normalization is an effective operation in our experiments, this is consistent with conclusion in [26].

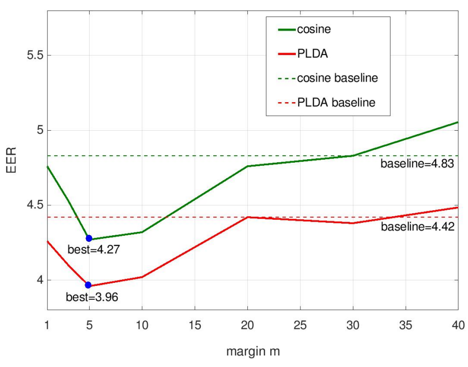
Comparison between result of ID 8 and ID 10 shows that statistics pooling is more effective than average pooling. Hence, in order to obtain better performance and keep consistency, we adopt statistics pooling and length normalization in experiments from ID 10 to 13. In these 4 results, the performances of EER in triplet loss are slightly improved over softmax loss, but the performances of $\mathrm{minDCF}_{0.01}$ in triplet loss are worse than those in softmax loss. The reason is that we use the hard mining strategy to select triplets in each mini-batch for triplet loss training, but it may not be a good enough way to construct triplets in this experiment. It proves that designing an appropriate mining strategy to fit the triplet loss is very difficult. However, in the same configuration, the triplet-center loss acquires better performance than triplet loss. Now, look at center loss (ID 12), it gets improvement compared with softmax loss. But due to no consideration of enlarging the inter-class distance, the result is worse than that of triplet-center loss. The comparison between these 4 results proves that our triplet-center loss not only overcomes the limitations of triplet loss and center loss, but also shares the superiorities of these two loss functions. Thus it is reasonable that triplet-center loss obtains the best performance in all our experiments. Specifically, comparing result of ID 13 with ID 10, our triplet-center loss reduces EER from softmax loss by 11.6%, 10.4% in cosine scoring and PLDA backend, respectively.

B. Study on Parameter Influence

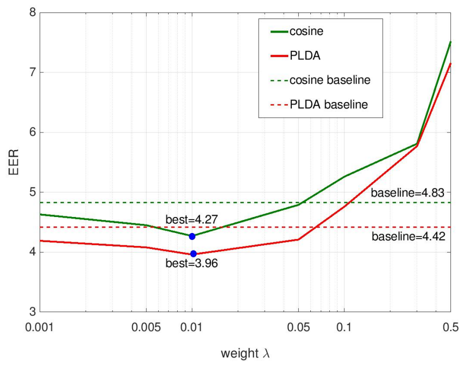
As shown in Eq. (3) and (4), both the margin m and the weight λ have influence on performance. To investigate the impact of these two hyper-parameters, we carry out more experimental analysis in this section.

The analysis is presented in Fig. 2, where the baseline is the result of softmax loss (ID 10 in Table II). Since EER and $\mathrm{minDCF}_{0.01}$ curves have almost the same tendency, here we only show EER curves. Extensive experiments indicate that the best choices are m = 5 and λ = 0.01. Hence, when studying on margin m in Fig. 2 (a), we fix λ to be 0.01, and when studying on weight λ in Fig. 2 (b), we fix m to be 5.

In Fig. 2 (a), we could find when margin m = 1, the result is close to the baseline, this is because m is so small that triplet-center loss could not produce a marked effect. The best result is m = 5, and as m increases from 5, the performance becomes worse and worse, it illustrates that too large value of m may lead to over-fitting. Now, look at Fig. 2 (b), although the best performance appears at λ = 0.01, it shows better results than baseline when λ is in range from 0.001 to 0.05. One possible explanation is that triplet-center loss and softmax loss work in different aspects, namely, they are complementary. Triplet-center loss aims to construct a metric learning space where deep embeddings are not only discriminative between different classes but also compact within the same class. However, achieving such a perfect space to set all embeddings to appropriate position is very difficult. In this case, softmax loss is employed to find a decision boundary of different classes in a separable label space. Once the model with softmax loss has converged, the clear decision boundary could help triplet-center loss further compact intra-class distance. Meanwhile, based on the



(a) Performance in terms of margin m



(b) Performance in terms of weight λ (λ is logarithmic scale)

Fig. 2. Performances in terms of (a) varying m when λ is fixed to 0.01 and (b) varying λ when m is fixed to 5.

appropriate configuration of hyper-parameters, triplet-center loss will not damage the classification performance of softmax loss. Thus good result can be achieved.

V. Conclusions

This paper applies an effective loss function, *i.e.*, triplet-center loss, to improve the performance of deep embedding learning method for SV task. It learns a center for each class and requires distances between samples and centers from the same class are closer than those from different classes. And it further introduces a margin to enforce a distance between different classes, leading to more discriminative deep embeddings. It not only overcomes the limitations of triplet loss and center loss, but also shares the superiorities of these two loss functions.

Extensive experiments are conducted on Voxceleb to evaluate the effectiveness of our proposed triplet-center loss. It obtains significant performance gains when comparing with baseline loss function. Specifically, our triplet-center loss reduces EER from softmax loss by 11.6%, 10.4% in cosine scoring and PLDA backend, respectively. In addition, further experiments show that appropriate values of margin m and weight λ are both crucial to system performance.

VI. Acknowledgement

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

References

[1] S. Ramoji and S. Ganapathy, "Supervised i-vector Modeling-Theory and Applications,"
[2] N. Dehak, P. 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, pp. 788-798, 2011.
[3] P. Kenny, "Bayesian speaker verification with heavy-tailed priors," in *Proc. Odyssey,* pp. 14, 2010.
[4] L. Burget, O. Plchot, S. Cumani, O. Glembek, P. Matejka, and N. Brummer, "Discriminatively trained probabilistic linear discriminant analysis for speaker verification," in *Proc. ICASSP,* pp. 4832–4835, 2011. in *Proc. Interspeech,* pp. 1091-1095, 2018.
[5] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *Proc. ICASSP,* pp. 4052–4056, 2014.
[6] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *Proc. ICASSP,* pp. 5329–5333, 2018.
[7] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep neural network embeddings for text-independent speaker verification," in *Proc. Interspeech,* pp. 999-1003, 2017.
[8] H. Muckenhirn, M. Magimai-Doss, and S. Marcel, "Towards directly modeling raw speech signal for speaker verification using CNNs," in *Proc. ICASSP,* pp. 4884-4888, 2018.
[9] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer, "End-to-end text-dependent speaker verification," in *Proc. ICASSP,* pp. 5115-5119, 2016.
[10] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," in *Proc. Interspeech,* pp. 1086–1090, 2018.
[11] W. Cai, J. Chen, and M. Li, "Exploring the encoding layer and loss function in end-to-end speaker and language recognition system," *arXiv preprint arXiv:1804.05160,* 2018.
[12] Z. Huang, S. Wang, and K. Yu, "Angular softmax for short-duration text-independent speaker verification," in *Proc. Interspeech,* pp. 3623–3627, 2018.
[13] M. Hajibabaei and D. Dai, "Unified hypersphere embedding for speaker recognition," *arXiv preprint arXiv:1807.08312,* 2018.
[14] C. Li, X. Ma, B. Jiang, X. Li, X. Zhang, X. Liu, *et al*, "Deep speaker: an end-to-end neural speaker embedding system," *arXiv preprint arXiv:1705.02304,* 2017.
[15] S. Novoselov, V. Shchemelinin, A. Shulipa, A. Kozlov, and I. Kremnev, "Triplet loss based cosine similarity metric learning for text-independent speaker recognition," in *Proc. Interspeech,* pp. 2242–2246, 2018.
[16] X. He, Y. Zhou, Z. Zhou, S. Bai, and X. Bai, "Triplet-Center Loss for Multi-View 3D Object Retrieval," in *Proc. CVPR,* pp. 1945-1954, 2018.
[17] N. Li, D. Tuo, D. Su, Z. Li, and D. Yu, "Deep discriminative embeddings for duration robust speaker verification," in *Proc. Interspeech,* pp. 2262–2266, 2018.
[18] A. Nagrani, J. S. Chung, and A. Zisserman, "VoxCeleb: a large-scale speaker identification dataset," in *Proc. Interspeech,* 2017.
[19] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A Unified Embedding for Face Recognition and Clustering," in *Proc. CVPR,* 2015.
[20] A. Hermans, L. Beyer, and B. Leibe, "In Defense of the Triplet Loss for Person Re-Identification," *arXiv preprint arXiv:1703.07737,* 2017.
[21] Y. Dong, K. Zhang, Z. Li, and Y. Qiao, "A Discriminative Feature Learning Approach for Deep Face Recognition," in *Proc. ECCV,* 2016.
[22] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel *et al*, "The kaldi speech recognition toolkit," in *Proc. ASRU,* 2011.
[23] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, *et al*, "Automatic differentiation in pytorch," 2017.
[24] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE,* vol. 86, pp. 2278-2324, 1998.
[25] H. Kai, X. Zhang, S. Ren, and J. Sun "Deep residual learning for image recognition," in *Proc. CVPR,* pp. 770-778, 2016.
[26] W. Cai, J. Chen, and M. Li, "Analysis of Length Normalization in End-to-End Speaker Verification System," in *Proc. Interspeech,* pp. 3618-3622, 2018.