INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

Masked Proxy Loss For Text-Independent Speaker Verification

Jiachen Lian, Aiswarya Vinod Kumar*, Hira Dhamyal*, Bhiksha Raj+, Rita Singh+*

*Electrical and Computer Engineering, Carnegie Mellon University, USA
+Language Technologies Institute, Carnegie Mellon University, USA
{jlian2, avinodku}@andrew.cmu.edu, {hyd, bhiksha, rsingh}@cs.cmu.edu

Abstract

Open-set speaker recognition can be regarded as a metric learning problem, which is to maximize inter-class variance and minimize intra-class variance. Supervised metric learning can be categorized into pair-based learning and proxy-based learning [1]. Most of the existing metric learning objectives belong to the former division, the performance of which is either highly dependent on sample mining strategy or restricted by insufficient label information in the mini-batch. Proxy-based losses mitigate both shortcomings, however, fine-grained connections among entities are either not or indirectly leveraged. This paper proposes a Masked Proxy (MP) loss which directly incorporates both proxy-based relationship and pair-based relationship. We further propose Multinomial Masked Proxy (MMP) loss to leverage the hardness of speaker pairs. These methods have been applied to evaluate on VoxCeleb test set and reach state-of-the-art Equal Error Rate(EER).

Index Terms: speaker recognition, deep metric learning, masked proxy, fine-grained

1. Introduction

Text-independent speaker verification [2] is the task of verifying speakers' claimed identities from recordings of their voice with no expectation that the words spoken in two recordings are the same. The most successful approaches are to use the deep neural network to derive discriminative speaker embeddings. These approaches typically take one of two forms.

In the first approach, the network is trained as a multi-class classification network that *classifies* a large number of speakers using softmax crossentropy loss and its enhanced variants [3–8]. The expectation is that this behavior will generalize to data outside the training set.

The second approach, which relates directly to the topic of this paper, is based on *metric learning*. In metric learning, instead of explicitly modelling the classes, the training losses are computed directly from *comparison of distances* between instances. A number of metric learning objectives have been proposed for this purpose, such as the Contrastive [3, 9], Triplet [3, 10], GE2E [3, 11], Prototypical [3, 12], and Angular Prototypical [3] losses.

While this approach has proven to be more effective than the classification-based approach [3], it faces considerable logistic challenges. Firstly, in order to maximally ensure that the distance between embeddings of same-class instances is consistently smaller than that between instances of different classes, ideally the distances (or similarities) between *every* pair of same-class instances must be compared to every pair of either of those instances with every recording from the *other* classes in the training set. For a training set of size N (recordings), raw computation of such comparisons scales by $O(N^3)$, an infeasible computation for any reasonably-sized training corpus.

Secondly, model updates are usually performed over *minibatches* of data rather than the entire training set, and most metric learning approaches restrict themselves to comparisons of classes that are present within the minibatches [3, 11, 12]. While the computation here is less daunting – scaling only by $O(NB^2)$ for minibatches of size B for a single pass through the data – a different suboptimality results. The size of minibatches is typically much smaller than the total number of classes in the training data. Even with exhaustive comparison of all triplets within each minibatch, updates made to the model will not have considered the complete possible set of comparisons across classes.

Proxy NCA [13], attempts to address these deficiencies by introducing learnable *proxy* embeddings, one per class in the training set, and replacing instance-instance comparisons with instance-proxy comparisons, thereby reducing the time complexity to visit all comparisons to $O(NP)$, where P is number of proxies. Since each training instance is compared to *all* proxies, including those for classes not currently in the minibatch, Proxy-NCA leads not only to faster convergence, but potentially also better models. However, by concentrating on entity-to-proxy distances (by "entity", we mean an actual data instance, as opposed to the proxies which are purely synthetic), it loses the more fine-grained entity-to-entity relationships. Proxy Anchor [1] attempts to address this by assigning dynamic weights to the entity-to-proxy comparisons, however, the real entity-to-entity relationships remain unexploited, since the actual comparisons are still between entities and proxies.

This paper proposes a Masked Proxy(MP) loss which takes both entity-to-entity distance and entity-to-proxy distance into consideration. Within each minibatch we utilize entity-to-entity comparisons for all classes represented in the minibatch, computed through the comparison of a randomly reserved query sample to batch centroids for the within-minibatch classes, and utilize entity-proxy comparisons only for classes not represented in the minibatch (i.e. by masking out the proxies for the within-minibatch classes). To leverage multi-similarity [14] and the hardness of positive pairs [1], we also propose a Multinomial Masked Proxy (MMP) loss. To the best of our knowledge, this is the first work to apply proxy-based loss in speaker recognition. Using our proposed approach, we achieve state-of-art Equal Error Rates(EER) on the VoxCeleb test set training on the VoxCeleb2 dev set.

2. Related Works

2.1. Proxy-Based Loss

In contrast to pair-based metric learning which computes losses entirely from authentic data instances, proxy-based losses utilize synthetic, learnable proxies to compose triplets.

2.1.1. Proxy NCA

Proxy NCA [13] assigns a proxy to each data point according to its class label. The objective is to make each embedding closer to its proxy than other proxies, as shown below:

$$L_{NCA} = -\frac{1}{N}\sum_{i=1}^{N}\log(\frac{e^{-d(x_i,p_i)}}{\sum_{j=1,j\neq i}^{N} e^{-d(x_i,p_j)}}) \tag{1}$$

where d denotes Euclidean distance.

2.1.2. Proxy Anchor

In contrast to Proxy NCA, Proxy Anchor [1] regards the proxy as an *anchor* that instances are drawn to. Both positive and negative pairs contribute to the loss objective by their hardness, which is illustrated in Equation 6 in [1].

$$\begin{aligned} L_{Anchor} = \frac{1}{|P_+|}\sum_{p\in P_+}\log(1+\sum_{x\in X_p^+} e^{-\alpha(s(x,p)-\delta)}) \\ + \frac{1}{|P|}\sum_{p\in P}\log(1+\sum_{x\in X_p^-} e^{\alpha(s(x,p)+\delta)}) \end{aligned} \tag{2}$$

Here P represents the proxy set, P_+ represents the set of proxies for all classes represented in the minibatch, X_p^+ denotes the set of all instances (from the minibatch) belonging to the same class as proxy p, and X_p^- denotes set of instances *not* from the same class as p. α is a scaling factor, δ is a margin and s refers to cosine similarity.

Note that neither Proxy NCA nor Proxy Anchor invoke actual entity-to-entity distances, thus not leveraging the fine-grained distance/similarity relations within a minibatch.

3. Proposed Masked Proxy Methods

3.1. Masked Proxy (MP)

As seen in Equations 1 and 2, both Proxy NCA and Proxy Anchor only deal with entity-proxy pairs. In our proposed loss, both entity-proxy pairs and entity-entity pairs are taken into consideration. Specifically, for all classes represented within a minibatch we will compute entity-based distances, and compute entity-proxy distances for the remaining classes. In the following discussion, all computations are over a minibatch.

Let L_i denote the class label of x_i and L_{p_i} denote the corresponding class label of proxy p_i. L_M is the label set of classes represented in the minibatch. c_{L_i} is the centroid [12] of embeddings whose labels are L_i. N_{L_i} is the number of instances in the minibatch with labels are L_i. Similar to [3, 11, 12], we reserve one instance from each class in the minibatch as a "query" for the class embedding as query for each class, and we define X_Q as the query set for the minibatch, without duplicates. The centroids can be represented as follows:

$$c_{L_i} = \frac{1}{N_{L_i}-1}\sum_{j\neq i, L_j = L_i} x_j \tag{3}$$

For each query instance in X_Q, we compose the following loss:

$$l(x_i) = -\log(\frac{e^{s(x_i,c_{L_i})}}{\sum\limits_{\substack{L_j\neq L_i\\ L_j\in L_M}} e^{s(x_i,c_{L_j})} + \sum\limits_{\substack{p_k\\ L_{p_k}\notin L_M}} e^{s(x_i,p_k)}}) \tag{4}$$

where

$$s(u,v) = \alpha((u^T v) - \beta) \tag{5}$$

Both embeddings and proxies are normalized by length. α and β are learnable smoothing factor and bias.

The numerator in Equation 4 considers the similarity of the instance to its own proxy. The first term in the denominator invokes the similarity of the instance to the *mini-batch centroids* for all classes in the minibatch, and is effectively entirely entity-based. The second term refers to the similarity of the instance to the proxies of all classes that are not in the minibatch and is proxy-based. The loss objective over the entire minibatch is formulated as follows:

$$l_1 = \frac{1}{|X_Q|}\sum_{x_i\in X_Q} l(x_i) \tag{6}$$

Equation 6 does not refer to the proxies of classes represented in the minibatch. In order to be able to also update these proxies classes, we include a *Mask Proxy Regulator* (MPR), which minimizes the distance between these proxies and the centroid for their class, while maximizing their separation to other centroids:

$$l_2 = -\frac{1}{|L_M|}\sum_{\substack{c_{L_i}\\ L_i\in L_M, L_{p_i}=L_i}}\log(\frac{e^{s(c_{L_i},p_i)}}{\sum\limits_{L_j\neq L_i} e^{s(c_{L_j},p_i)}}) \tag{7}$$

Mask Proxy loss can be defined as the weighted summation of Equations 6 and 7:

$$l_{mp} = l_1 + \lambda_{mp} l_2 \tag{8}$$

where λ_{mp} is balancing factor, which is set to 0.5 in this paper.

3.2. Multinomial Masked Proxy (MMP)

Multinomial-based loss functions can leverage both *multi-similarity* among pairs [14] and hardness of positive pairs [1] as compared to softmax-based functions. The multinomial form of Equation 6 can be formulated as follows:

$$\begin{aligned} l_{1m} = \log(1+\sum_{x_i\in X_Q} e^{-s(x_i,c_{L_i})}) \\ + \frac{1}{|X_Q|}\sum_{x_i\in X_Q}\log(1+\sum_{\substack{L_j\neq L_i\\ L_j\in L_M}} e^{s(x_i,c_{L_j})}) \\ + \frac{1}{|X_Q|}\sum_{x_i\in X_Q}\log(1+\sum_{\substack{p_k\\ L_{p_k}\notin L_M}} e^{s(x_i,p_k)}) \end{aligned} \tag{9}$$

In Mask Proxy loss, every positive entity-centroid pair contributes equally to the loss objective, which can be illustrated through Eq. 10.

$$\frac{\partial l_{MP}}{\partial s(x_i,c_{L_i})} = \frac{\partial l_1}{\partial s(x_i,c_{L_i})} = -\frac{1}{|X_Q|} \tag{10}$$

However, multinomial loss will assign a dynamic weight to each positive pair. As shown in Equation 11, this dynamic weight is dependent on the pair itself, meaning the harder positive pairs with lower similarity would contribute more to the loss objective.

$$\frac{\partial l_{1m}}{\partial s(x_i,c_{L_i})} = -\frac{e^{-s(x_i,c_{L_i})}}{1+\sum\limits_{x_i\in X_Q} e^{-s(x_i,c_{L_i})}} \tag{11}$$

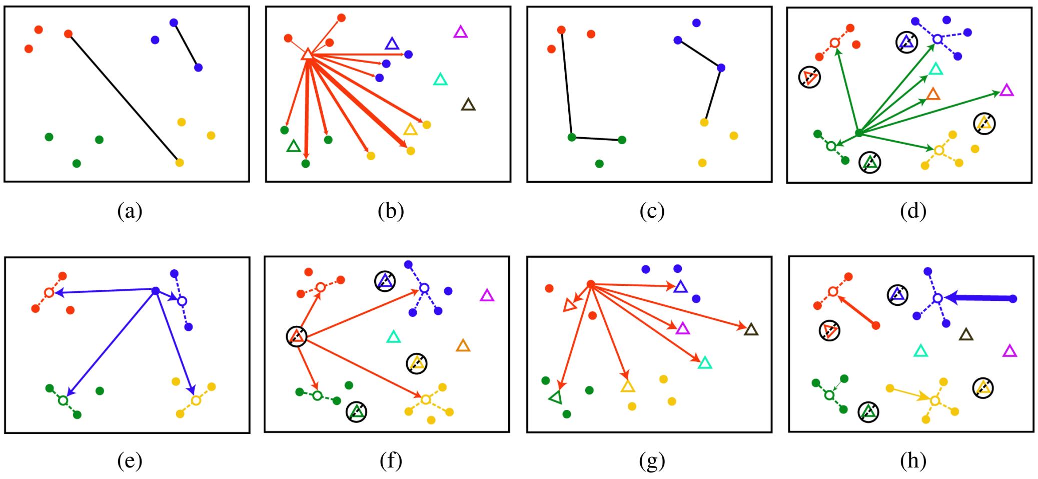

Figure 1: *Comparison of metric learning objectives such as: (a) Contrastive, (b) ProxyAnchor, (c) Triplet, (d) MP l1 loss, (e) Angular Prototypical, (f) MP Regulator (Eq.7), (g) ProxyNCA, (h) MMP Positive Pairs. Each color represents a unique class. Solid circle point is the entity while the triangle with a specific color is the proxy of the corresponding class. The hollow circle connected to solid circles by dash line represents the centroid computed by averaging those entities. For (Angular) Prototypical(e) or MP/MMP(h), one embedding is reserved as a query and other embeddings are applied to compute the centroid regarding each class. No additional identities are incorporated for Contrastive(a), Triplet(c) and (Angular) Prototypical(e). For proxy-based losses, all classes are included. The masked proxy is the triangle surrounded by a black circle and masked by a dash line, shown in (d) (f) (h). For MP l1 loss, we compute the distance between the query and all centroids, and the distance between the query and unmasked proxies, as shown in (d). For MP regulator, we compute the distance between the proxy and all centroids for all masked proxies, as shown in (f). The illustration of MMP l1 loss is almost the same as that of MP l1 loss, except that each positive pair is assigned a weight which is correlated to the distance of this positive pair, which is shown by the width of lines in (h).*

The expression of MMP can be formulated as follows:

$$l_{mmp} = l_{1m} + \lambda_{mmp} l_2 \qquad (12)$$

Where the regulator l_2 is kept as its original form and λ_{mmp} is a balancing factor. λ_{mmp} is set to 0.5 in this paper.

A comparison of the different metric learning objectives described in this section and other standard metric learning objectives is shown in Fig. 1.

3.3. Training Complexity

We denote N, B, P as training size, batch size and proxy size respectively. For each data point in Eq. 6, we compute distances over C pairs, which gives rise to a training complexity of $O(NP)$. In Eq. 7, complexity should be $O(P^2)$ to visit all possible proxy-centroid pairs. Hence, the final training complexity should be within $O(NP + P^2) \sim O(NP)$, The same result can be derived for MMP. Table 1 is a comparison of training complexity over different loss functions.

Table 1: *Training Complexity Comparison*

Loss Criterion	Training Complexity
Triplet	$O(N^3)$
Semi-hard Triplet [15]	$O(\frac{N^3}{B^2})$
Smart Triplet [16]	$O(N^2)$
(Angular) Prototypical [3]	$O(N^2)^*$
Proxy NCA [13]	$O(NP)$
Proxy Anchor [1]	$O(NP)$
MP/MMP	$O(NP)$

* Here we discuss the case where number of data points is 2 for each class within the mini-batch.

4. Experiments

4.1. Dataset

We use the VoxCeleb dataset [9, 17]. We train on VoxCeleb2 dev set which contains 5994 identities and test on VoxCeleb1 test set which contains 40 identities.

4.2. Model and Evaluation Scoring Method

We apply the backbone (Thin-ResNet34+SAP) proposed in [3] as baseline model, where SAP is self-attentive pooling [18]. We adopt Equal Error Rate(EER) as evaluation metric, where the similarity score is computed using the same method with [3,9]: 10 features are sampled using a specific window size for each utterance in a pair, then the mean of 10×10 Euclidean distances are computed as the distance measurement. Note that these 10 features cover the full utterance. That is to say, during testing, the full utterance rather than a fixed-length sample from the raw audio is evaluated. Details for evaluation method can be found in [3, 9].

4.3. Implementation Details

The experiments are performed on Nvidia Tesla V100 platform using PyTorch. We denote τ as audio duration. We apply fixed $\tau = 2s$ and $\tau = 4s$ random segments respectively during training while using the full utterance during testing [3, 9]. We employ the Mel-Spectrogram feature with 40 frequency channels. In accordance with [3], we adopt the window size of 25ms and step size of 10ms. Sampling rate is set as 16k.

We experiment with Proxy NCA, Proxy Anchor, MP and MMP on Thin-ResNet34. When experimenting with Proxy Anchor, different sets of hyper-parameters are explored. To be specific, margin is varied from 0.1 to 0.5 with an increment of 0.1 and smoothing factor is varied from 10 to 70 with an increment of 10. The balancing factor λ is varied from 0.1 to 1 with

Table 2: *Evaluation on VoxCeleb1 test set*

Loss Function	Hyper-parameters	EER%(E1, τ=2s,B=400)	EER%(E2,τ=2s,B=800)	EER%(E3,τ=4s,B=400)
Triplet [3]	m=0.1	2.50±0.06	2.49±0.08	2.50±0.07
ProtoTypical [3]	M=2	2.37±0.10	2.34±0.08	2.32±0.02
GE2E [3]	M=3	2.53±0.03	2.51±0.03	2.50±0.01
Angular Prototypical [3]	M=2	2.31±0.05	2.21±0.03	2.26±0.05
Proxy NCA	\	2.42±0.07	2.42±0.05	2.40±0.04
Proxy Anchor	m=0.15,s=50	2.39±0.03	2.40±0.02	2.39±0.01
MP(ours)	λ=0.3	**2.08±0.03**	**2.04±0.05**	**2.05±0.03**
MMP(ours)	λ=0.3	**2.06±0.03**	**2.02±0.04**	**2.02±0.02**
MP-Balance(ours)	M=2,λ=0.3	**2.03±0.01**	**1.97±0.03**	**1.99±0.01**
MMP-Balance(ours)	M=2,λ=0.3	**1.99±0.02**	**1.95±0.03**	**1.93±0.01**

an increment of 0.1. While training with MP and MMP, the initial margin β is set to 0.1 and smoothing factor α is set to 10. Inspired by the idea that balanced training samples matter [3, 11, 12], we also adopt this manner in MP and MMP. Concretely, the number of samples for each class is a fixed value M which is a hyper-parameter. In this experiment, we set M as 2 based on the fact that *fewer shot* learning matters in Angular Prototypical loss [3]. We call this manner **MP-Balance** for Mask Proxy and **MMP-Balance** for Multinomial Mask Proxy. We adopt the expected batch size B of 800 and 400[1] respectively. We apply SGD as optimizer with a starting learning rate of 0.2 and ReduceLROnPlateau as learning rate scheduler with a factor of 0.8, patience of 3 and $EER\%$ as metric.

Based on the aforementioned statement, three experiments are performed and we denote them as **E1**($\tau = 2s, B = 400$), **E2**($\tau = 2s, B = 800$), **E3**($\tau = 4s, B = 400$) respectively. We also duplicate the experiments in [3] using Triplet, Prototypical, GE2E and Angular Prototypical respectively. To analyze the training speed over different loss objectives, in each epoch, we compute the $EER\%$ on test set and record the results in Fig. 2. Based on our experiments, there is no significant difference on training convergence over $E1$, $E2$ and $E3$. Thus we just take $E1$ into consideration in Fig. 2. The final results are displayed in Table 2. The hyper-parameters in both Table.2 and Fig. 2 are the best ones that we have explored.

4.4. Discussion

Results Analysis. Table 2 presents the EER achieved using different loss functions under certain settings. The performance for Proxy NCA and Proxy Anchor are similar to that of Prototypical. Based on the current best smoothing factor and margin, Proxy Anchor reaches slightly lower EER than Proxy NCA. The potential reason is that Proxy Anchor indirectly leverages the fine-grained data relationship.

The proposed loss functions including Mask Proxy and Multinomial Mask Proxy outperform the existing state-of-the-art Angular Prototypical loss objective. Based on these results, we speculate that both fine-grained data-to-data connection and number of classes incorporated in the mini-batch are the major contributing factors for a lower EER achieved by the proposed loss functions.

[1]For MP/MMP-Balance, input size is 2 × [$batchsize$, feature size] and *expected batch size* is $batchsize$ × 2, which is consistent with [3]. For MP and MMP, input size is [$\sum_{i=1}^{batchsize} M_i$, feature size], where M_i is number of features for a certain class i. M_i is randomly determined to be 2 or 3, and $\sum_{i=1}^{batchsize} M_i = batchsize \times 2.5$ is the *expected batch size*. E.g. for *expected batch size* 800, the $batchsize$ is set to 320.

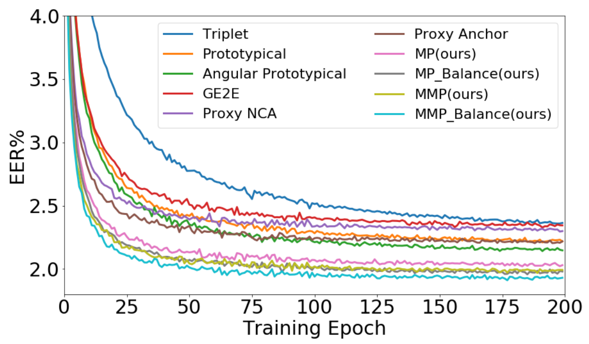

Figure 2: *EER on the test set - VoxCeleb1 dataset using different loss objectives over epochs. Details found in Sec.4.3*

Ablation Study Based on the results in Table. 2, balanced data inputs always result in lower EER for both MP and MMP. Both larger batch size B and larger training audio duration τ give lower EER for MP, MMP, MP-Balance and MMP-Balance. These conclusions hold for all three experiments $E1$, $E2$ and $E3$.

Training Complexity and Speed. Based on our experiments, both training audio duration and batch size make no difference in the training speed. To be concise, we just choose the training process of $E1$ in Fig. 2 for the illustration of training speed over different loss functions. It is observed that Triplet converges the most slowly due to its largest training complexity as shown in Table. 1. Other non-proxy-based losses converge faster than Triplet due to its smaller training complexity. Proxy-based losses converge fastest. Note that though Proxy NCA and Proxy Anchor give a larger final EER in comparison to that of Angular Prototypical, they still converge faster than Angular Prototypical. Our proposed loss functions give the best EER as well as training speed.

5. Conclusion

The proposed loss objectives in this paper achieve state-of-the-art speaker verification performance on the VoxCeleb dataset. To the best of our knowledge, it is the first work that applies proxy-based losses on the task of speaker verification. The proposed Masked Proxy losses leverage real data-to-data relationships more than other proxy-based objectives in a novel manner. Our future work will focus on self-supervised and low-resource speaker recognition.

6. References

[1] S. Kim, D. Kim, M. Cho, and S. Kwak, "Proxy anchor loss for deep metric learning," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 3235–3244.

[2] T. Kinnunen and H. Li, "An overview of text-independent speaker recognition: From features to supervectors," *Speech Communication*, vol. 52, no. 1, pp. 12 – 40, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167639309001289

[3] J. S. Chung, J. Huh, S. Mun, M. Lee, H.-S. Heo, S. Choe, C. Ham, S. Jung, B.-J. Lee, and I. Han, "In defence of metric learning for speaker recognition," *Interspeech 2020*, Oct 2020. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2020-1064

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

[5] X. Shi, X. Du, and M. Zhu, "End-to-end residual cnn with l-gm loss speaker verification system," in *2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)*. IEEE, 2018, pp. 1–5.

[6] Z. Wang, K. Yao, S. Fang, and X. Li, "Joint optimization of classification and clustering for deep speaker embedding," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2019, pp. 284–290.

[7] N. Li, D. Tuo, D. Su, Z. Li, D. Yu, and A. Tencent, "Deep discriminative embeddings for duration robust speaker verification." in *Interspeech*, 2018, pp. 2262–2266.

[8] Y. Li, F. Gao, Z. Ou, and J. Sun, "Angular softmax loss for end-to-end speaker verification," in *2018 11th International Symposium on Chinese Spoken Language Processing (ISCSLP)*. IEEE, 2018, pp. 190–194.

[9] A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman, "Voxceleb: Large-scale speaker verification in the wild," *Computer Speech Language*, vol. 60, p. 101027, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0885230819302712

[10] C. Zhang, K. Koishida, and J. H. L. 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.

[11] L. Wan, Q. Wang, A. Papir, and I. L. Moreno, "Generalized end-to-end loss for speaker verification," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 4879–4883.

[12] J. Wang, K. Wang, M. T. Law, F. Rudzicz, and M. Brudno, "Centroid-based deep metric learning for speaker recognition," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 3652–3656.

[13] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh, "No fuss distance metric learning using proxies," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 360–368, 2017.

[14] X. Wang, X. Han, W. Huang, D. Dong, and M. R. Scott, "Multi-similarity loss with general pair weighting for deep metric learning," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5017–5025, 2019.

[15] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 815–823, 2015.

[16] B. Harwood, G. VijayKumarB., G. Carneiro, I. Reid, and T. Drummond, "Smart mining for deep metric learning," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2840–2848, 2017.

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

[18] Y. Zhu, T. Ko, D. Snyder, B. K.-W. Mak, and D. Povey, "Self-attentive speaker embeddings for text-independent speaker verification," in *INTERSPEECH*, 2018.