Ju *et al. EURASIP Journal on Audio, Speech, and Music Processing* (2022) 2022:17
https://doi.org/10.1186/s13636-022-00249-4

EURASIP Journal on Audio, Speech, and Music Processing

METHODOLOGY Open Access

Masked multi-center angular margin loss for language recognition

Minghang Ju[1,2], Yanyan Xu[1,2*], Dengfeng Ke[3*] and Kaile Su[4]

Abstract

Language recognition based on embedding aims to maximize inter-class variance and minimize intra-class variance. Previous researches are limited to the training constraint of a single centroid, which cannot accurately describe the overall geometric characteristics of the embedding space. In this paper, we propose a novel masked multi-center angular margin (MMAM) loss method from the perspective of multiple centroids, resulting in a better overall performance. Specifically, numerous global centers are used to jointly approximate entities of each class. To capture the local neighbor relationship effectively, a small number of centers are adapted to construct the similarity relationship between these centers and each entity. Furthermore, we use a new reverse label propagation algorithm to adjust neighbor relations according to the ground truth labels to learn a discriminative metric space in the classification process. Finally, an additive angular margin is added, which understands more discriminative language embeddings by simultaneously enhancing intra-class compactness and inter-class discrepancy. Experiments are conducted on the APSIPA 2017 Oriental Language Recognition (AP17-OLR) corpus. We compare the proposed MMAM method with seven state-of-the-art baselines and verify that our method has 26.2% and 31.3% relative improvements in the equal error rate (EER) and C_{avg} respectively in the full-length test ("full-length" means the average duration of the utterances is longer than 5 s). Also, there are 31.2% and 29.3% relative improvements in the 3-s test and 14% and 14.8% relative improvements in the 1-s test.

Keywords: Spoken language recognition, Masked multi-center angular margin, Multi-center loss, Single-center loss, ECAPA-TDNN neural network

1 Introduction

Language recognition (LR) is the task of automatically identifying or verifying a language or languages being spoken in a given speech utterance [1]. It plays an essential role in multilingual speech pre-processing, which is typically followed by speech recognition systems and automatic translation systems [2].

Generally speaking, there are two types of LR tasks: close-set LR and open-set LR. Most current research focuses on close-set LR, meaning that all test utterances correspond to a target language. In other words, the language of the training set and the test set are the same. However, the open-set LR means that the test utterances are unlikely to be strictly restricted to a target language but may also correspond to some unknown languages [3]. This paper mainly improves the performance of the former category.

Due to the similarity in research fields, recent advances in automatic speech recognition and speaker recognition based on single-center loss (SCL) have improved language recognition applications. Single-center loss can be divided into two types, that is, classification loss and metric loss.

The pioneering work of using the classification loss is to learn the speaker embedding for speaker recognition [4–6]. Since then, popular methods train embeddings using softmax classifiers [7–11]. Although the softmax loss can learn separable embeddings, since it is not explicitly designed to optimize embedding similarity, it is not

*Correspondence: xuyanyan@bjfu.edu.cn; dengfeng.ke@blcu.edu.cn
[1]School of Information Science and Technology, Beijing Forestry University, 35 Qing-Hua East Road, Beijing 100083, China
[2]Engineering Research Center for Forestry-oriented Intelligent Information Processing of National Forestry and Grassland Administration, Beijing Forestry University, 35 Qing-Hua East Road, Beijing 100083, China
Full list of author information is available at the end of the article

distinguishable enough. Therefore, the model trained by softmax is usually combined with the back end of PLDA [6, 12] to generate a scoring function [13, 14]. Wang et al. [15] proposed angular softmax (A-softmax), using cosine similarity as the logit input of the softmax layer to solve this problem. Many studies have proven that A-softmax is superior to softmax in speaker recognition [16–19]. Additive margin variables AM-Softmax [15, 20] and AAM-Softmax [21] have been proposed to increase the variance between classes by introducing a cosine margin penalty on the target logit, which has been well applied due to their good performances [16–18]. However, training AM-Softmax and AAM-Softmax have proven to be challenging because they are sensitive to the scale and the marginal value of the loss function. To improve the performance of AM-Softmax loss, Zhou et al. [22] proposed to dynamically set the margin of each training sample different from the cosine angle of that sample. Specifically, the smaller the cosine angle, the greater the distance between the training sample and the corresponding class in the feature space, and the better the intra-class compactness.

The embedding learned from the classification loss is only optimized for the separation between classes. Differently, the metric loss is used to embed the speaker, which not only expands the inter-class variance but also reduces the intra-class variance [22]. Triplet loss [23, 24] and contrast loss [25] optimize the embedding space by minimizing the distance between feature pairs and the same speaker and maximizing the distance between feature pairs and different speakers. However, these methods require careful attention to the choice of the couple and triplet, which is time-consuming and performance-sensitive. The generalized end-to-end (GE2E) loss [26] is an enhanced contrast loss, which directly optimizes the cosine distance between the speaker embedding and the centroid, without the need for complicated sample selection such as triple loss [23, 24] and contrast loss [25]. This metric loss also has the final classification layer, and the extraction embedding also needs to be removed.

To sum up, the classification loss only optimizes the distance between a sample and the center without considering the relationship between any two samples. On the contrary, the metric loss only optimizes the distance between two samples without considering the sample and center relationship. In this paper, we employ the advantages of both the classification loss and the metric loss and propose to use the multi-center loss (MCL). More specifically, given C classes, MCL designs K centers for each class, so there are $K \cdot C$ centers. For a training sample, we will get K positive centers and $K \cdot (C-1)$ negative centers, where "positive center" means a sample belongs to a class, and correspondingly, "negative center" represents a sample does not belongs to a class. A similar method was also studied in [27–29]. Deng et al. [27] is to optimize the distance between the sample and one of the pre-defined multi-centers without considering the other centers. Although [28] optimizes the distance between the sample and all the pre-defined centers, it only optimizes the distance-weighted of all centers. Zhu et al. [29] proposes a new proxy-based deep graph metric learning (ProxyGML) method for graph classification, which uses fewer proxies but has better overall performances. As [29] provides a good example of the optimization method, we also use this method for our multi-center loss in this paper. Moreover, Wang et al. [15], Wang et al. [20] and Deng et al. [21] introduce an additional corner penalty between the speaker embeddings and the centroid, which reduces the distance of the class inner corners so that the speaker embeddings belonging to the same speaker are gathered closely around the centroid. Inspired by this, we also penalize the distance between the sample and the center cosine by increasing the margin. In summary, based on MCL, the optimization method provided by ProxyGML and additional corner penalties, we propose masked multi-center angular margin (MMAM) loss in this paper. Our contributions are summarized as follows:

1. We propose to use multi-center loss to optimize the cosine distance between the language embedding and the corresponding multi-centers and optimize the cosine distance between the multi-centers simultaneously while taking advantage of both the classification loss and the metric loss.
2. In addition, we added an additional angular margin to the multi-center loss, which learns more discriminative language embeddings by simultaneously explicitly enhancing intra-class compactness and inter-class differences.
3. Thirdly, we add a masking operation to the multi-center loss, so that the network itself adaptively selects the optimization method of samples and multiple centroids.
4. The proposed masked multi-center angular margin loss is evaluated by comparing it with seven state-of-the-art baselines. Both its performance and convergence speed far exceed those of the baselines.
5. The proposed MMAM loss can be readily applied to various similar tasks, such as speaker verification and speaker recognition. To the best of our knowledge, we introduce the multi-center loss into language recognition for the first time.

This paper is arranged as follows. In Section 2, we review the GE2E loss [26], AM-Centroid loss [30], Softmax loss [7], AAM-Softmax loss [21], and DAM-Softmax loss [22], Sub-center loss [27], and Softtriple loss [28], which are the start-of-the-art types of loss used in LR methods. In Section 3, we describe our MMAM loss. We give experimental setup and experimental results in

Sections 4 and 5, respectively. Finally, in Section 6, we conclude this paper.

2 Baselines

Our MMAM loss is inspired by metric loss (e.g., GE2E [26] and AM-Centroid [30]) and classification loss (e.g., Softmax [7], AAM-Softmax [21], and DAM-Softmax [22]), as well as the newly popular MCL (e.g., Sub-center [27] and Softtriple [28]).

2.1 Metric Loss

GE2E and AM-Centroid are two types of metric loss, which serve as two baselines for our experiments.

2.1.1 GE2E

Let a batch consist of N languages and M utterances per language. We use $x_{ij}(1 \le i \le N, 1 \le j \le M)$ to denote the language embedding extracted from language i utterance j. In GE2E training, every utterance in the batch except the query itself is used to form centroids. As a result, the embedding centroids of sample k that belong to different classes and the same class from the query are defined as follows:

$$c_k = \frac{1}{M}\sum_{m=1}^{M} x_{k,m}, \tag{1}$$

$$c_k^{(-j)} = \frac{1}{M-1}\sum_{m=1,m\neq j}^{M} x_{k,m}. \tag{2}$$

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

$$S_{ij,k} = \begin{cases} w \cdot cos\left(\theta_{x_{ij},c_k^{(-j)}}\right) + b & \text{if} \quad i = k, \\ w \cdot cos\left(\theta_{x_{ij},c_k}\right) + b & else, \end{cases} \tag{3}$$

where w, b are learnable parameters and θ_{x_{ij},c_k} refers to the angle between x_{ij} and c_k. The final GE2E loss [26] is defined as:

$$\ell_G = -\frac{1}{N}\sum_{i,j} log\frac{e^{S_{ij,i}}}{\sum_{k=1}^{N} e^{S_{ij,k}}}. \tag{4}$$

2.1.2 AM-Centroid

Although GE2E loss promotes the embedding of language k to be closer to its centroid c_k than other centroids, there is still a sizeable intra-class distance. If the included margin between each embedded language and its center of mass is large, it will be penalized. After setting b = 0, replacing w with a scalar value s, and adding the angle margin m to the target angle, we get AM-Centroid [30] from Eq. 3 as follows:

$$S_{ij,k} = \begin{cases} s \cdot cos\left(\theta_{x_{i,j},c_k^{(-j)}} + m\right) & \text{if} \quad i = k, \\ s \cdot cos\left(\theta_{x_{i,j},c_k}\right) & else. \end{cases} \tag{5}$$

2.2 From softmax to angular softmax

This section mainly introduces the development process of classification loss, and we choose Softmax [7], AAM-Softmax [21], and DAM-Softmax [22] as three baselines for subsequent experiments.

2.2.1 Softmax

The softmax loss consists of a softmax function followed by a multi-class cross-entropy loss. Its basic form is defined as:

$$\ell_S = -\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{C} e^{W_j^T x_i + b_j}}, \tag{6}$$

where N and C are the numbers of training samples and the number of classes respectively, and x_i and y_i are the feature representation of the ith sample and the target class of the ith sample, respectively, and W and b are the weight and bias of the last layer of the backbone network respectively. This loss function only penalizes classification errors and does not explicitly enforce intra-class compactness and inter-class separation.

2.2.2 Angular softmax

By normalizing the weight and the input vector, the softmax loss can be re-expressed. The posterior probability only depends on the cosine of the angle between the weight and the input vector. The expression $W_{y_i}^T x_i + b_{y_i}$ in the numerator on the right-hand side of Eq. 6 can be rewritten as:

$$\|W_{y_i}\|\|x_i\| cos(\theta_{y_i}) + b_{y_i}. \tag{7}$$

From Eq. 7, we normalize the weight vector to unit norm, and discard the deviation term by setting $\|W_{y_i}\| = 1$ $\|x_i\| = 1$ and $b_{y_i} = 0$, which leads to the so-called angular softmax loss [15], defined as follows:

$$\ell_A = -\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{cos(\theta_{y_i,i})}}{\sum_{j=1}^{C} e^{cos(\theta_{j,i})}}. \tag{8}$$

Equation 8 is just a rewrite of Eq. 6, which has the same advantages and disadvantages as Eq. 6. To alleviate this problem, the cosine margin m is added to Eq. 8. The additive angular margin penalty is equal to the geodesic distance margin penalty in the normalized hypersphere. There are two types of additional corner penalties. One is the penalty for corners in the angle range, and the other

is for angles. The corresponding AM-Softmax [20] and AAM-Softmax [21] loss formulas are:

$$\ell_{AM} = -\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{s\cdot(cos(\theta_{y_i,i})-m)}}{\sum_{j\neq y_i}^{C} e^{s\cdot(cos(\theta_{j,i}))} + e^{s\cdot(cos(\theta_{y_i,i})-m)}}, \quad (9)$$

$$\ell_{AAM} = -\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{s\cdot(cos(\theta_{y_i,i}+m))}}{\sum_{j\neq y_i}^{C} e^{s\cdot(cos(\theta_{j,i}))} + e^{s\cdot(cos(\theta_{y_i,i}+m))}}, \quad (10)$$

where s is a fixed scale factor to prevent the gradient of the training phase from being too small. The cosine margin m is manually tuned and is usually larger than 0.

2.2.3 DAM-Softmax

In Eq. 9, the cosine margin m is a constant shared by all training samples. However, the penalty scales for different samples should be different. DAM-Softmax [22] is based on the assumption that the smaller the $cos(\theta)$, the farther the sample is from the corresponding class in the feature space, and the margin m should be set larger to force compactness within the class, so DAM-Softmax loss changes margin m to:

$$m_i = \frac{me^{(1-cos(\theta_{y_i}))}}{\lambda}, \quad (11)$$

where m_i is the cosine margin value of the ith sample, and m is the essential margin value, and λ is the control factor that controls the margin value range.

2.3 Multiple centroids

Multi-centroid loss is first proposed in the field of graphics [27, 28]. Sub-center [27] and Softtriple [28] are two types of the existing multi-center classification loss, which serve as two baselines for subsequent experiments.

2.3.1 Sub-center

Assuming that each class has K centers, then the similarity between sample x_i and class c can be defined as:

$$S_{i,c} = \max_k x_i^T w_c^k. \quad (12)$$

Sub-center loss [27] is designed to optimize the distance between the sample and one of the pre-defined multi-centers without considering the other centers. The loss is defined as follows:

$$\ell_{Sub} = -log\frac{e^{cos(\theta_{i,y_i}+m)}}{e^{s\cdot(cos(\theta_{i,y_i}+m))} + \sum_{j=1,j\neq y_i}^{N} e^{s\cdot(cos(\theta_{i,j}))}}, \quad (13)$$

where $\theta_{i,j} = arccos\left(max_k\left(W_{jk}^T x_i\right)\right), and\ k \in \{1, 2, \ldots, K-1, K\}$.

2.3.2 Softtriple

Softtriple loss [28] mainly considers the similarity distance between the sample and all centers by weighted summation. Its main formulas are as follows:

$$S_{i,c} = \sum_k \frac{e^{\frac{1}{\gamma}x_i^T w_c^k}}{\sum_k e^{\frac{1}{\gamma}x_i^T w_c^k}} e^{x_i^T w_c^k}, \quad (14)$$

$$\ell_{Softtriple}(x_i) = -log\frac{e^{\lambda(S_{i,c}-\delta)}}{e^{\lambda(S_{i,c}-\delta)} + \sum_{j\neq y_i} e^{\lambda S_{i,c}}}, \quad (15)$$

where λ is the weighted summation of the similarity between the representative sample and all centers, and δ is similar to the previous parameter $m \in (0, 1)$.

3 The proposed method

3.1 Formulation

Our goal is to design a more discriminative feature embedding by adjusting the network structure parameters. Given the training set with C classes, a small batch of B samples is randomly selected from the training set as in usual batch training. Indicating that the embedding vector of the ith data sample is x_i^s, and the corresponding label is y_i^s, then the embedding output of the small batch of samples extracted by the deep neural network can be defined as $S = \left\{\left(x_1^s, y_1^s\right), \left(x_2^s, y_2^s\right), \ldots, \left(x_B^s, y_B^s\right)\right\}$. In addition, we assign K trainable centers to each class. The value of K is preset, so it requires to find the optimal value of K. The total number of central sets to be trained is $C \cdot K$. The central set can be expressed as $C = \left\{\left(x_1^c, y_1^c\right), \left(x_2^c, y_2^c\right), \ldots, \left(x_{C\cdot K}^c, y_{C\cdot K}^c\right)\right\}$. In order to constrain the similar relationship between the sample and the center, we also represent the center label in the set C as a one-hot label matrix $Y^c \in \{0, 1\}^{(C\cdot K)\times C}$ with $Y_{ij}^c = 1$ if $y_i^c = j$ and $Y_{ij}^c = 0$ else.

3.2 Masking of similar relationships

The basic idea behind MMAM loss is to ensure that each data sample is close to its associated positive center and away from its negative center. Given the embedding vector $\left\{x_i^s\right\}_{i=1}^B$ and all centers $\left\{x_j^c\right\}_{j=1}^{C\cdot K}$ in the small batch processing, we first construct the similarity matrix S between the sample and the center, where $S \in \Re^{B\times(C\cdot K)}$ represents the similarity between the sample and the center, which is calculated by Eq. 12. Both x_i^s and x_j^c are normalized to be the unit length, and thus $S_{ij} \in [-1, 1]$.

Figure 1 illustrates the overall architecture of MMAM. By generating a similarity matrix between the sample and all the centers, the local relationship around each sample can be further constructed into a series of similarity matrices, capturing the fine-grained neighborhood structure

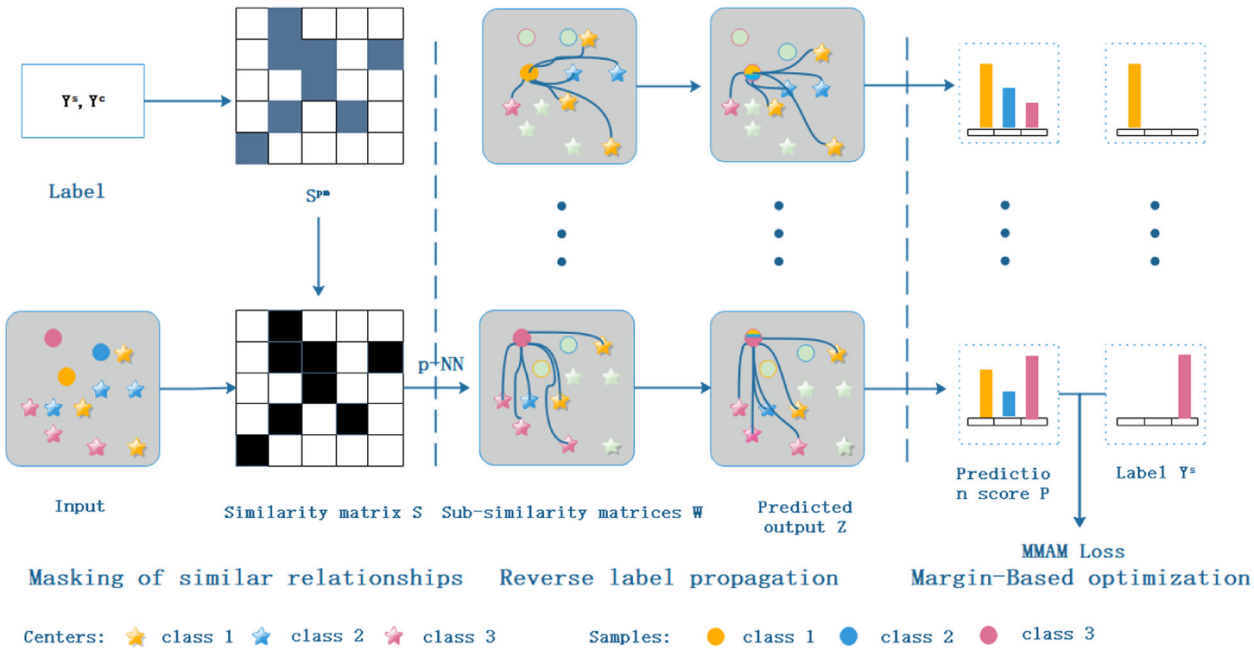


Fig. 1 The overall architecture diagram of our proposed MMAM. The input contains the samples in a mini-batch and all the centers, and the labels include the samples and the pre-defined multi-center labels. Different colors represent different category labels. Multicolor samples indicate that their neighboring centers influence their classification information

better, that is, the similarity matrix S in Fig. 1 can be transformed into a series of sub-similarity matrices W, and the optimization process changes from coarse-grained to fine-grained. Our approach keeps the maximum value of p in each row in S to construct a p-nearest neighbor (p-NN) matrix. Since all centers are initialized randomly, directly selecting the center closest to K for each sample may miss many positive centers. These centers of the same category cannot be updated at the same time in each iteration. Therefore, we introduce a positive mask S^{pm} to ensure that all the positive centers of each sample are selected, which can also be regarded as a kind of "soft" constraint on the centers, and as shown in Fig. 1, a series of sub-similarity matrices W encourages similar centers through the guidance of reverse label propagation in subsection 3.3 close to their corresponding samples while keeping similar centers close to each other.

The positive mask S^{pm} is based on the label of the sample and the center, which essentially reflects the genuine similarity between them:

$$S_{ij}^{pm} = \begin{cases} 1, & \text{if} \quad y_i^s = y_j^c, \\ 0, & else. \end{cases} \tag{16}$$

We calculate the index of the p-max value of each row of $(S + S^{pm})$ and store it in the set of p elements $\Omega = \{\ldots, (i,j), \ldots\}$. Then construct the sub-similarity matrix to be represented by a sparse neighbor matrix W:

$$W_{ij} = \begin{cases} S_{ij}, & \text{if} \quad (i,j) \in \Omega, \\ 0, & \text{else}, \end{cases} \tag{17}$$

where $W \in \Re^{B \times (C \cdot K)}$. With the help of the positive mask S^{pm}, even if p is relatively small, all the positive centers of each sample will participate in each sub-similarity matrix. In detail, p is given by $p = \lceil r \cdot C \cdot K \rceil$, where a scale factor $r \in (0, 1]$ is introduced to easily obtain sub-similarity matrices of different scales.

3.3 Reverse label propagation

In subsection 3.2, the sub-similarity matrix W between samples and centers has been constructed, and the samples and centers from the same category should be close to each other [31]. In semi-supervised learning, the idea behind traditional label propagation (LP) is to infer unknown labels through manifold structure [32]. Zhu et al. [29] utilizes the known labels to adjust the manifold structure using the proposed reverse label propagation (RLP) algorithm. Inspired by [29, 31, 32], we use the known sample labels to guide the sub-similarity matrix W for optimization. According to the LP idea, all the sub-similarity matrices W are encoded as the predicted output Z.

$$Z^s = WY^c, \tag{18}$$

where $Z \in \Re^{B\times C}$. The pre-defined multi-center label Y guides the sub-similarity matrix W to learn the distance between the positive center and the related sample, reflecting how its neighboring center points guide the sample's classification information from the same category to be close to each other. Specifically, in optimizing the target through RLP, the positive center is close to the relevant training sample, and the negative center will be far away from the training sample.

3.4 Margin-based optimization

After introducing the previous two subsections, the sub-similarity matrix W of the similarity matrix has been constructed. The relationship Z between the positive and negative centers and the relevant samples has been designed. Finally, the process of classification learning is analyzed. Like the classification loss, the prediction output Z is first converted into a prediction score P through the softmax operation and then optimized with the ground-truth label. In this way, each value in P that reflects the similarity between the sample and the positive center or the negative center will be increased or decreased. Because there are many masking terms in Z, the denominator in the softmax function will be over-calculated, so it cannot be predicted correctly. Therefore, we use a new mask softmax function to prevent the mask value from affecting the prediction score:

$$P(\tilde{y}_i^s = j \mid X_i^s) = \frac{M_{ij} e^{Z_{ij}^s}}{\sum_{j=1}^{C} M_{ij} e^{Z_{ij}^s}}, \tag{19}$$

where $\tilde{y}_i^s$ indicates the prediction label of the ith sample x_i^s in S, and Z_{ij} indicates the jth predictive element of the ith sample, and mask $M \in \{0,1\}^{B\times C}$ is defined as follows:

$$M_{ij} = \begin{cases} 1, & \text{if} \quad Z_{ij} \neq 0, \\ 0, & else. \end{cases} \tag{20}$$

Cross-entropy loss is computed between the predicted score and the ground-truth label for each sample. Its performance can be improved when introducing a cosine marginal penalty on the target logit to increase the variance between classes. Since the previous optimization method for constructing the similarity matrix is similar to ProxyGML, the difference between MMAM and ProxyGML is mainly in the optimization calculation method. Therefore, unlike the classification-based optimization calculation method in ProxyGML, we introduce an additional cosine marginal penalty to increase the between-class variance and reduce the within-class variance, which can be proved very effective by later experiments. After adding the cosine marginal penalty, the calculation is as follows:

$$\ell_{MMAM}^s = -\frac{1}{B}\sum_{i=1}^{B}\sum_{j=1}^{C} log \frac{e^{s\cdot cos(\theta_{i,y_i}+m)}}{e^{s\cdot cos(\theta_{i,y_i}+m)} + \sum_{j=1, j\neq y_i}^{N} e^{s\cdot cos(\theta_{i,j})}}, \tag{21}$$

where $\theta_{i,j} = arccos(P(\tilde{y}_i^s = j \mid x_i^s))$. Also, we impose a constraint on the center to ensure that similar centers are very close and dissimilar centers are far away. Specifically, we regard each center as the second type of sample and other similar or different centers as positive and negative centers. Repeating the above method, first construct the total similarity matrix between the centers as:

$$S_{i,j}^c = (x_i^c)^T x_j^c, \tag{22}$$

where $S^c \in \Re^{(C\cdot K)\times(C\cdot K)}$, and both x_i^c and x_j^c are normalized to unit length. Since the multi-center is initialized randomly, we no longer construct the p-NN sub-matrix for S^c. The scale factor r is set to 1. Then, according to the RLP, the predicted output of the relationship between the intermediates is as follows:

$$Z^c = S^c Y^c. \tag{23}$$

The output Z^c can also turn into a prediction score through softmax:

$$P(\tilde{y}_i^c = j \mid X_i^s) = \frac{e^{Z_{ij}^c}}{\sum_{j=1}^{C} e^{Z_{ij}^c}}, \tag{24}$$

where $\tilde{y}_i^c$ and Z_{ij} indicate the prediction label of the ith center x_i^c in C and the jth predictive element of the ith center, respectively.

Similar to Eq. 21, we also introduce an additional cosine marginal penalty and get Eq. 25.

$$\ell_{MMAM}^c = -\frac{1}{C\cdot K}\sum_{i=1}^{C\cdot K}\sum_{j=1}^{C} log \frac{e^{s\cdot cos(\theta_{i,y_i}+m)}}{e^{s\cdot (cos(\theta_{i,y_i}+m))} + \sum_{\substack{j=1 \\ j\neq y_i}}^{N} e^{s\cdot (cos(\theta_{i,j}))}} \tag{25}$$

Combining Eq. 21 and Eq. 25, respectively, representing the MMAM loss between the sample and the center, and the MMAM loss between the center and the center, our final loss function is

$$\ell_{MMAM} = \ell_{MMAM}^s + \lambda \ell_{MMAM}^c, \tag{26}$$

where λ balances the loss between the sample and the center and between the centers. Equation 26 can lead to more discriminative language embeddings.

The source code of computing all the loss functions in this paper is available at https://github.com/hangxiu/mmam_loss/.

4 Experimental setup

4.1 The dataset

The proposed MMAM loss model is evaluated on the AP17-OLR dataset, which is for the second Oriental Language Recognition Challenge [33, 34]. The dataset is composed initially of Speechocean and Multilingual Minor Language Automatic Speech Creation and Recognition (M2ASR). There are 10 languages in the dataset, including Kazakh in China (ka-cn), Tibetan in China (ti-cn), Uyghur in China (uy-id), Cantonese in China Mainland and Hong Kong (ct-cn), Mandarin in China (zh-cn), Indonesian in Indonesia (id-id), Japanese in Japan (ja-jp), Russian in Russia(ru-ru), Korean in Korea (ko-kr), and Vietnamese in Vietnam(vi-vn) [35].

The dataset is divided into a train/dev part and a test part. The number of speakers and total volume of each language is shown in Table 1. For male and female speakers, the volume of each speaker is balanced. There is no overlap of speakers in the train/dev and test subsets. All speech utterances are recorded via mobile phones with a sampling rate of 16kHz and a sampling capacity of 16 bits. In the train/dev subset, there are approximately 10 hours of recordings of each language. This dataset provides a full-length subset, including train-all, dev-all, and test-all. Besides, it also provides two short-term (short-duration audio segments) subsets, including train-1s, train-3s, dev-1s, dev-3s, test-1s, and test-3s, which are randomly selected from train-all, dev-all and test-all according to duration.

Our system is evaluated on test-1s, test-3s, and test-all, including 22,051, 19,999, and 22,051 utterances, respectively. In particular, we use the train and dev subsets jointly as the training set.

Table 1 AP17-OLR dataset

Language	Train/dev		Test	
	Speaker	All utt.	Speaker	All utt.
ka-cn	86	4200	86	1800
ti-cn	34	11,100	34	1800
uy-id	353	5163	353	1800
ct-cn	24	7676	6	2556
zh-cn	24	7198	6	2400
id-id	24	7667	6	2557
ja-jp	24	7655	6	2548
ru-ru	24	7183	6	1800
ko-kr	24	7195	6	2398
vi-vn	24	7197	6	2396

4.2 Data augmentation

It is a well-known fact that neural networks benefit from data augmentation that generates additional training samples. Therefore, we generate a total of 4 additional samples for each utterance. Specifically, our paper studies two enhancement methods commonly used in speech processing-additive noise and room impulse response (RIR) simulation [36]. For additive noise, we use the MUSAN corpus [37], which contains 60 hours of speech, 42 hours of music, and 6 hours of noise, such as dial tone or environmental sounds. For the room impulse response, we use the simulated RIR filter provided in [36]. In each training step, the noise and RIR filters are randomly selected [38]. The type of enhancement used is similar to [5, 39]. The recording is enhanced by one of the following four methods:

1. **RIR filters:** We change the gain of the RIR filter to produce a more diverse reverberation signal.
2. **Speech:** Randomly select three to seven recordings from MUSAN, and then add a random signal-to-noise ratio (SNR) from 13 to 20 decibels to the original signal. The duration of the additive noise is matched to the sampled period.
3. **Music:** A single music file is randomly selected from MUSAN and added to the original signal, with a signal-to-noise ratio ranging from 5 to 15 dB.
4. **Noise:** Background noise in MUSAN is added to the recordings from 0 to 15 dB SNR.

4.3 Implementation details

4.3.1 Input features

We extract 80-dimensional log Mel-filterbank energies for each speech frame of width 25 ms and step 10 ms. The training speech segment is set to 2 s, which generates a spectrogram with the size of 200×80. Two-second random crops of the log Mel-filterbank feature vectors are normalized through cepstral mean subtraction, and no voice activity detection is applied.

4.3.2 Training settings

Our implementation is based on the PyTorch framework [40] and uses the Adam algorithm [41] to optimize deep neural networks. λ is set to 0.3. We use the initial learning rate 1e-3 decreasing by 5% every 5 epochs to train Softmax, AAM-Softmax ($m = 0.3$), DAM-Softmax ($m = 0.3$), GE2E, AM-Centroid ($m = 0.3$), Sub-center ($m = 0.3$), Softtriple ($m = 0.3$), ProxyGML and our MMAM ($m = 0.3$). The models trained by MMAM with ($m = 0.3$) are respectively used as pre-trained models to train MMAM with ($m > 0.3$) and the initial learning

rate is changed to 1e−4. The minimum batch size of all types of classification loss is set to 64. When training GE2E loss and AM-centroid loss, each batch contains 10 languages, and each language contains 6 speech fragments, to roughly match the minimum batch size of the classification loss.

4.3.3 Back-end

In the train/dev subset, the average vectors of a language can model the language. The test utterance score in a specific language can be the cosine distance between the vector of the test utterance and the vector of the language model generated by the train/dev subset. The formula is as follows:

$$Score(E_{avg}, E_{test}) = \frac{E_{avg}^T \cdot E_{test}}{\|E_{avg}\| \|E_{test}\|}, \tag{27}$$

where E_{avg} is the enrollment utterance mean, and E_{test} is the test utterance vector.

4.4 Evaluation metrics

As in LRE15 [33, 34], C_{avg}, minimum detection cost function (*min*DCF), detection error tradeoff (DET) curve, and equal error rate (EER) [11] are used to evaluate the performance of different loss systems. These metrics evaluate the system from different perspectives, thereby providing more reliable analysis and conclusions of experimental results. The pair-wise loss that constitutes the miss and false alarm probability of a specific target/non-target language pair is defined as:

$$C(L_t, L_n) = P_{Target} P_{Miss}(L_t) + (1 - P_{Target}) P_{FA}(L_t, L_n), \tag{28}$$

where L_t and L_n are the target and non-target languages, respectively; P_{Miss} and P_{FA} are the missing and false alarm probabilities, respectively. P_{target} is the prior probability for the target language, which to 0.5 in the evaluation. C_{avg} as the average of the above pair-wise performance:

$$C_{avg} = \frac{1}{N} \left\{ P_{Target} \cdot \sum_{L_i} P_{Miss}(L_t) + \frac{1}{N-1} \sum_{L_t} \sum_{L_n} [\,(1 - P_{Target}) P_{FA}(L_t, L_n)] \right\}, \tag{29}$$

where N is the number of languages.

4.5 The neural network architecture

The DNN architecture used to extract language embedding is based on [42], with several modifications, as shown in Fig. 2. The frame-level feature extraction layer first passes through a layer of ordinary convolutional layer (CNN). It then passes through the IM-TDNN-Block module, which consists of 3 SE-Res2Block blocks, consisting of a 1-frame background containing an extended convolution of the front and back dense layers. The first dense layer can reduce the feature dimension, while the second dense layer can restore the number of features to the original size. Next is the SE-Block to scale each channel. A skip connection covers the entire unit, and these layers have parameters, including the number of filters, the filter size, and the expansion factor. Considering the hierarchical nature of TDNN, these deeper features

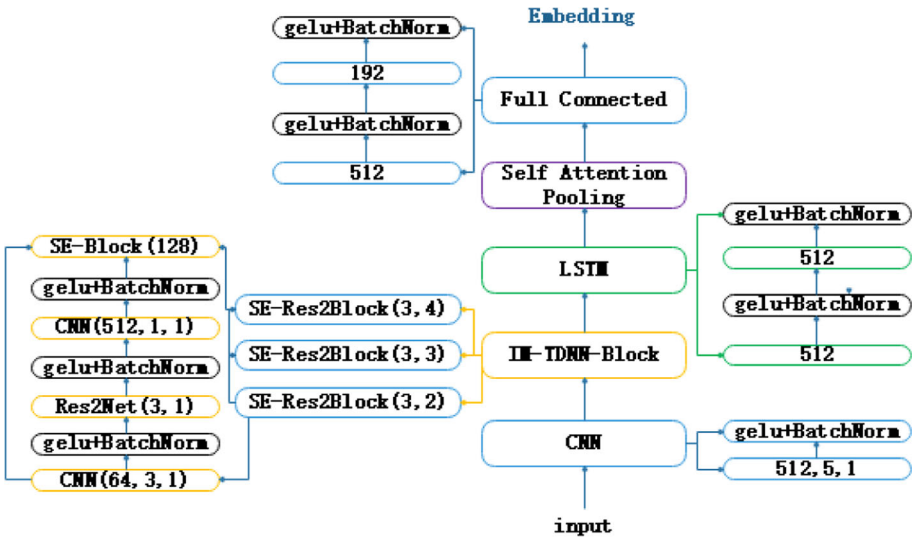


Fig. 2 The neural network architecture

are the most complex and closely related to language relationships. However, based on the evidence in [43], we believe that shallower feature mapping also contributes to more robust language embedding. For each frame, the network structure connects the output feature maps of all SE-Res2Blocks blocks. Two unidirectional LSTM layers with 512 neural units are used to capture the long-term time dependence of the frame-level feature sequence. The pooling layer based on the time attention mechanism is further extended to the channel dimension, which allows the network to pay more attention to language characteristics that will not be activated at the same or similar time. Using the weighted statistical pooling layer, the speech-level feature vector is generated by calculating the weighted mean and standard deviation of the input feature sequence and gathering this statistical information together. The last two fully connected layers have 512 and 192 nodes, respectively, and project the speech-level feature vectors into the 192-dimensional language embedding.

We choose GELU [44] activation function or without activation function and use batch normalization [45] to accelerate training. Using the classification loss (e.g., Softmax, AAM-Softmax, DAM-Softmax, Subcenter, Softtriple, ProxyGML and our MMAM) to train the DNN, another fully connected with ten nodes or a multiple of them is attached to the last layer of the structure as the classification layer. All LSTM layers use tanh as the activation function between them and use the hard sigmoid as the activation function for recurrent steps. The early stop is also used in training. When the loss of the test set does not decrease in three consecutive periods, the training is completed. In our experiments, we do not set the validation set separately and perform validation directly on the test set, so there may be the risk of overfitting and the

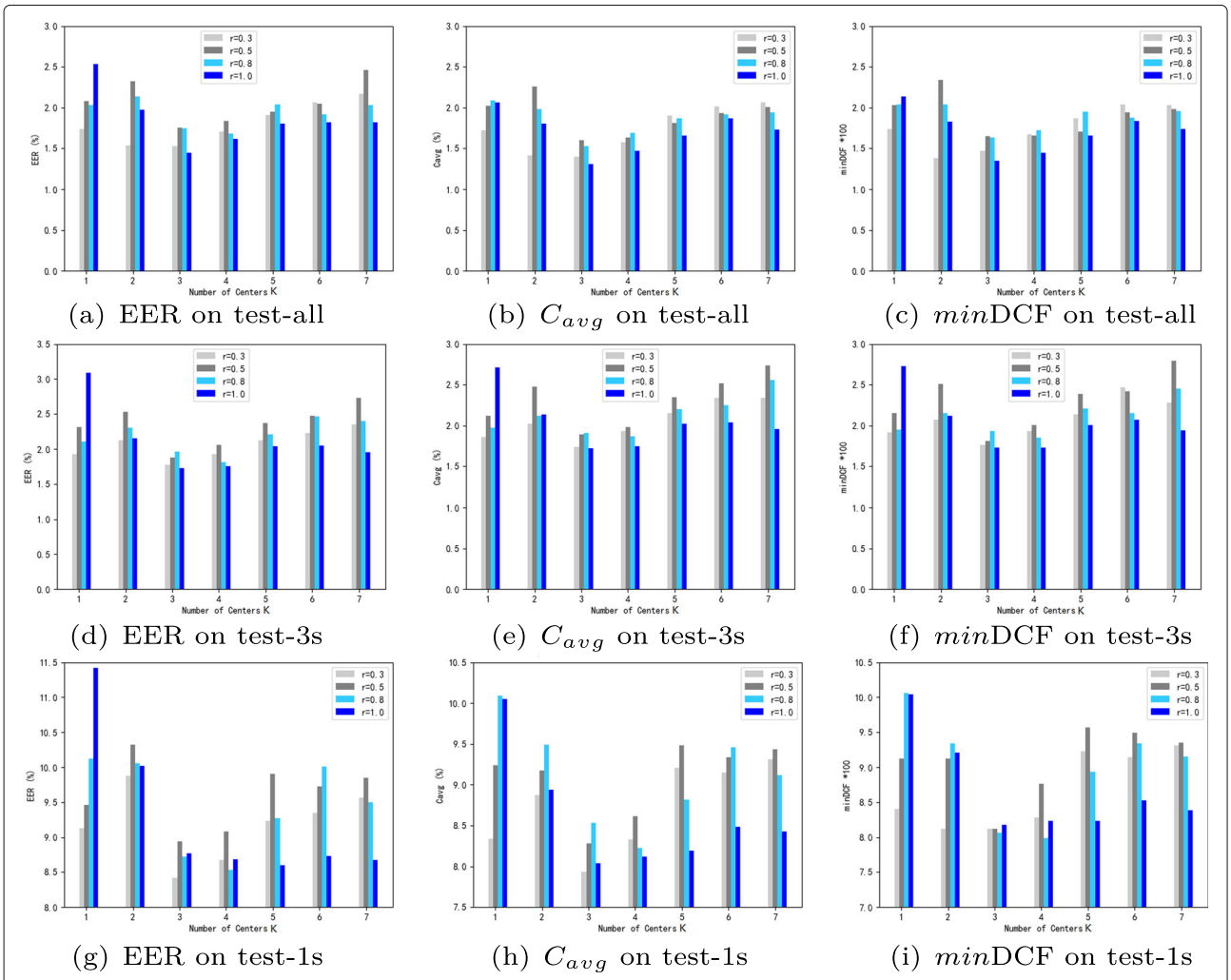


Fig. 3 Experimental results with different numbers of centers (*K*) under four different scale factors (*r*)

Table 2 Experimental results with different centers (K) under two different scale factors (r) of MMAM and ProxyGML. The performances are measured by EER (%), C_{avg} (%) and *min*DCF (100*)

ID	Loss	Parameter	Test-all			Test-3 s			Test-1 s		
			EER	C_{avg}	*min*DCF	EER	C_{avg}	*min*DCF	EER	C_{avg}	*min*DCF
1	MMAM	m=0.3, K=1, r=0.3	**1.74**	**1.72**	**1.74**	**1.92**	**1.86**	**1.92**	**8.13**	**8.34**	**8.4**
2	ProxyGML	K=1, r=0.3	2.37	2.41	2.35	2.49	2.62	2.47	8.77	8.6	8.69
3	MMAM	m=0.3, K=2, r=0.3	**1.53**	**1.41**	**1.38**	**2.12**	**2.02**	**2.07**	**8.83**	**8.74**	**8.8**
4	ProxyGML	K=2, r=0.3	2.53	2.48	2.51	2.74	2.9	2.72	9.88	8.94	8.87
5	MMAM	m=0.3, K=3, r=0.3	**1.52**	**1.4**	**1.47**	**1.77**	**1.74**	**1.76**	**8.42**	**7.93**	**8.12**
6	ProxyGML	K=3, r=0.3	2.7	2.62	2.68	2.92	2.96	2.87	9.34	9.33	9.23
7	MMAM	m=0.3, K=4, r=0.3	**1.61**	**1.47**	**1.45**	**1.73**	**1.75**	**1.73**	8.68	**8.12**	**8.23**
8	ProxyGML	K=4, r=0.3	1.7	1.58	1.67	1.92	1.93	1.93	**8.67**	8.33	8.28
9	MMAM	m=0.3, K=5, r=0.3	**1.8**	**1.66**	**1.67**	**2.04**	**2.02**	**2.01**	**8.6**	**8.19**	**8.23**
10	ProxyGML	K=5, r=0.3	1.91	1.9	1.87	2.12	2.15	2.14	9.23	9.23	9.21
11	MMAM	m=0.3, K=6, r=0.3	**2.06**	**2.01**	**2.04**	**2.23**	**2.47**	**2.34**	**8.82**	**8.68**	**8.79**
12	ProxyGML	K=6, r=0.3	2.37	2.35	2.34	2.58	2.66	2.57	9.35	9.15	9.14
13	MMAM	m=0.3, K=7, r=0.3	**2.17**	**2.06**	**2.03**	**2.35**	**2.28**	**2.34**	**8.71**	**8.63**	**8.69**
14	ProxyGML	K=7, r=0.3	2.21	2.14	2.2	2.53	2.58	2.51	9.56	9.31	9.31
15	MMAM	m=0.3, K=1, r=0.5	**2.07**	**2.02**	**2.03**	**2.31**	**2.12**	**2.15**	**8.32**	**8.19**	**8.27**
16	ProxyGML	K=1, r=0.5	2.43	2.29	2.41	2.46	2.57	2.42	9.46	9.24	9.12
17	MMAM	m=0.3, K=2, r=0.5	**2.49**	**2.26**	**2.34**	**2.53**	**2.48**	**2.51**	**9.27**	**9.13**	9.25
18	ProxyGML	K=2, r=0.5	2.84	2.77	2.82	2.85	3.03	2.85	10.32	9.17	**9.12**
19	MMAM	m=0.3, K=3, r=0.5	**1.75**	**1.6**	**1.65**	**1.88**	**1.89**	**1.81**	**8.93**	8.94	8.9
20	ProxyGML	K=3, r=0.5	2.47	2.44	2.46	2.56	2.63	2.54	8.95	**8.28**	**8.12**
21	MMAM	m=0.3, K=4, r=0.5	**1.73**	**1.63**	**1.66**	**2.06**	**1.98**	**2.01**	**8.95**	8.85	8.93
22	ProxyGML	K=4, r=0.5	2.44	2.47	2.43	2.58	2.65	2.56	9.08	**8.61**	**8.76**
23	MMAM	m=0.3, K=5, r=0.5	**1.94**	**1.81**	**1.91**	**2.37**	**2.35**	**2.39**	**9.49**	**9.4**	**9.47**
24	ProxyGML	K=5, r=0.5	2.88	2.86	2.84	3.06	3.17	2.99	9.91	9.48	9.57
25	MMAM	m=0.3, K=6, r=0.5	**2.04**	**1.93**	**1.94**	**2.47**	**2.52**	**2.42**	**9.73**	**9.34**	**9.49**
26	ProxyGML	K=6, r=0.5	2.4	2.23	2.14	2.73	2.82	2.71	10.23	10.62	10.34
27	MMAM	m=0.3, K=7, r=0.5	**2.46**	**2.34**	**2.43**	**2.73**	**2.74**	**2.79**	**9.85**	**9.42**	**9.35**
28	ProxyGML	K=7, r=0.5	2.64	2.61	2.56	2.93	2.85	2.84	10.34	10.45	10.53

overall experimental results may be optimistic. However, all the methods in this paper are optimized in this way, so they are comparable. Particularly, we do not use the test set to optimize the models with the loss, but only for the termination condition.

5 Experimental results

5.1 Parameter analysis

To find the optimal system of the proposed method, the parameters K and r in Section 3.2 and the parameter m in Section 3.4 will be experimentally and theoretically analyzed.

5.1.1 The number of initialization centers K

As described in Section 3.2, p is the number of optimized similarities selected from the similarity matrix S. The upper bound of p is $C \cdot K$. From the perspective of a network structure optimization, we hope to choose a smaller p so that the network can adaptively select appropriate optimization parameters. Therefore, the scale factor r is introduced to directly select p at different scales.

In terms of experiments, we use the four representative scales of r = 0.3, 0.5, 0.8, and 1.0 to conduct experiments to explore the influence of the number of centers K, as shown in 3. Figure 3a, d, and g illustrates the results of EER on test-all, test-3 s, and test-1 s, respectively, and Fig. 3b, e, and h illustrates the results of C_{avg} on test-all, test-3 s, and test-1 s, respectively, and Fig. 3c, f, and i illustrate the results of *minDCF* on test-all, test-3 s, and test-1 s, respectively. We see from Fig. 3 that among the four different scale factors (r), when K = 3, the performance is the best, confirming that the learned feature embedding can better capture intra-class changes through an appropriate number of local cluster centers. When K is further increased, the performance will decrease due to the overfitting when the center is over-parameterized.

In order to better illustrate the advantages of MMAM loss, the experimental results with different center numbers (K) under four different scale factors (r) of MMAM and ProxyGML are shown in Table 2. It is obvious that in all cases, MMAM is better than ProxyGML.

5.1.2 Scale factor r

We study the influence of the scale factor r when K is fixed at 3, and the margin m is fixed at 0.3. Figure 4a, b, and c illustrates the experimental results on test-all, test-3 s, and test-1 s, respectively. We see that when r = 0.4, the performance is the best, that is, the values of EER, C_{avg} and *min*DCF are the smallest, which shows that our proposed MMAM can make the network select the similarity matrix adaptively.

Specifically, as can be seen from Section 3.2, the number of positive centers is fixed at K = 3, and the number of negative centers is $p - K$, where $p = \lceil r \cdot C \cdot K \rceil$. On the

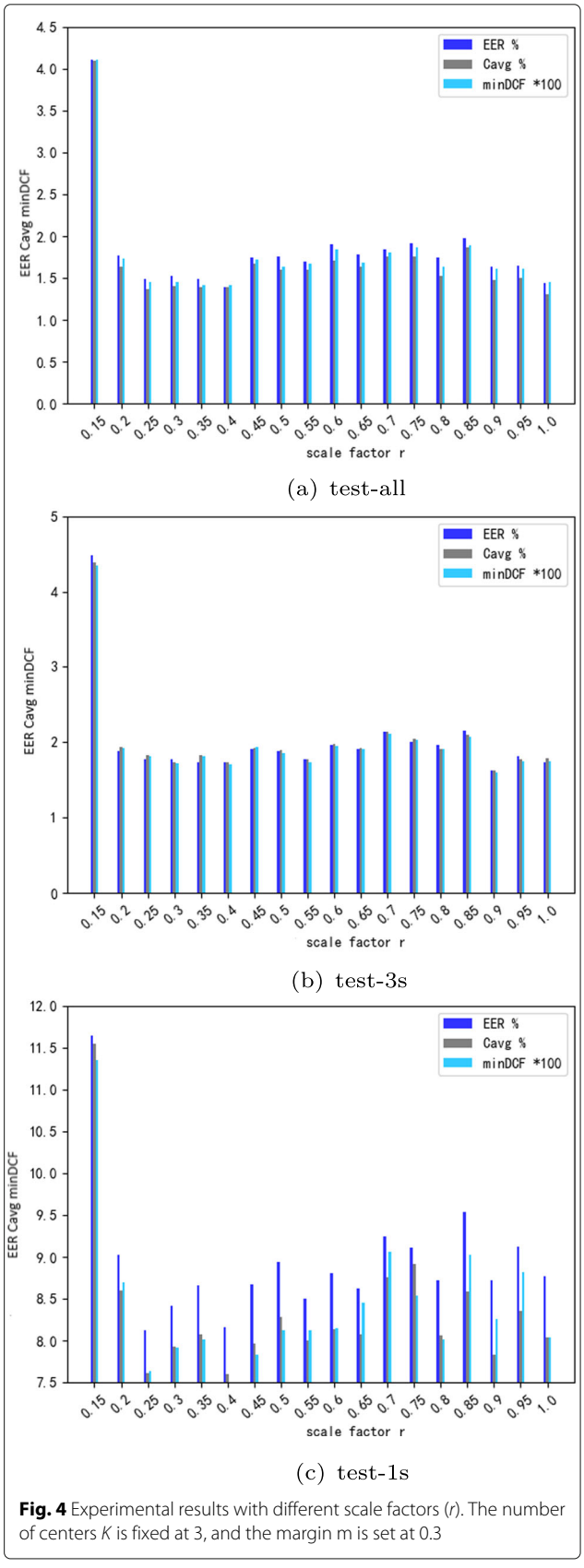


Fig. 4 Experimental results with different scale factors (r). The number of centers K is fixed at 3, and the margin m is set at 0.3

Table 3 Experimental results with different scale factors (r) of MMAM and ProxyGML. The number of centers K is fixed at 3. The performances are measured by EER (%), C_{avg} (%) and *min*DCF (*100)

ID	Loss	Parameter	Test-all			Test-3 s			Test-1 s		
			EER	C_{avg}	*min*DCF	EER	C_{avg}	*min*DCF	EER	C_{avg}	*min*DCF
1	MMAM	m=0.3, K=3, r=0.15	**4.1**	**4.11**	**4.09**	**4.48**	**4.34**	**4.38**	**11.64**	**11.35**	**11.54**
2	ProxyGML	K=3, r=0.15	5.3	5.35	5.12	5.66	5.63	5.78	13.34	13.46	13.64
3	MMAM	m=0.3, K=3, r=0.2	**1.76**	**1.73**	**1.63**	**1.88**	**1.93**	**1.94**	**9.03**	**8.7**	**8.6**
4	ProxyGML	K=3, r=0.2	2.83	2.86	2.92	3.01	3.3	3.15	10.23	10.47	10.36
5	MMAM	m=0.3, K=3, r=0.25	**1.49**	**1.45**	**1.37**	**1.77**	**1.82**	**1.83**	**8.21**	**7.64**	**7.61**
6	ProxyGML	K=3, r=0.25	2.57	2.83	2.77	2.79	2.78	2.93	9.65	9.34	9.73
7	MMAM	m=0.3, K=3, r=0.3	**1.52**	**1.47**	**1.4**	**1.77**	**1.76**	**1.74**	**8.42**	**8.12**	**7.93**
8	ProxyGML	K=3, r=0.3	2.7	2.62	2.68	2.92	2.96	2.87	9.34	9.23	9.33
9	MMAM	m=0.3, K=3, r=0.35	**1.49**	**1.42**	**1.39**	**1.74**	**1.81**	**1.83**	**8.66**	**8.01**	**8.07**
10	ProxyGML	K=3, r=0.35	2.67	2.57	2.54	2.85	2.78	2.84	9.23	9.12	9.35
11	MMAM	m=0.3, K=3, r=0.4	**1.39**	**1.31**	**1.39**	**1.73**	**1.72**	**1.71**	**8.16**	**7.3**	**7.6**
12	ProxyGML	K=3, r=0.4	2.65	2.62	2.62	2.66	2.73	2.64	9.37	9.16	9.32
13	MMAM	m=0.3, K=3, r=0.45	**1.74**	**1.72**	**1.67**	**1.91**	**1.94**	**1.93**	**8.67**	**7.83**	**7.96**
14	ProxyGML	K=3, r=0.45	2.69	2.65	2.73	2.94	2.97	2.91	9.41	9.26	9.35
15	MMAM	m=0.3, K=3, r=0.5	**1.75**	**1.6**	**1.64**	**1.88**	**1.89**	**1.85**	8.94	**8.28**	**8.12**
16	ProxyGML	K=3, r=0.5	2.47	2.46	2.44	2.56	2.63	2.54	**8.93**	8.94	8.9
17	MMAM	m=0.3, K=3, r=0.55	**1.7**	**1.6**	**1.67**	**1.78**	**1.77**	**1.74**	**8.5**	**8**	**8.12**
18	ProxyGML	K=3, r=0.55	2.35	2.41	2.42	2.49	2.48	2.47	9.43	9.56	9.34
19	MMAM	m=0.3, K=3, r=0.6	**1.9**	**1.71**	**1.84**	**1.97**	**1.98**	**1.95**	**8.8**	**8.14**	**8.15**
20	ProxyGML	K=3, r=0.6	2.56	2.54	2.55	2.84	3.03	2.82	9.19	9.18	9.18
21	MMAM	m=0.3, K=3, r=0.65	**1.78**	**1.63**	**1.68**	**1.91**	**1.92**	**1.91**	**8.62**	**8.07**	**8.45**
22	ProxyGML	K=3, r=0.65	2.55	2.46	2.52	2.81	2.9	2.8	9.17	9.09	9.13
23	MMAM	m=0.3, K=3, r=0.7	**1.84**	**1.8**	**1.75**	**2.14**	**2.11**	**2.14**	**9.24**	**9.06**	**8.76**
24	ProxyGML	K=3, r=0.7	3.01	2.95	3	3.16	3.36	3.14	9.42	9.52	9.31
25	MMAM	m=0.3, K=3, r=0.75	**1.91**	**1.76**	**1.86**	**2**	**2.05**	**2.03**	**9.11**	**8.91**	**8.54**
26	ProxyGML	K=3, r=0.75	3.14	3.15	3.42	3.24	3.43	3.45	9.32	9.53	9.42
27	MMAM	m=0.3, K=3, r=0.8	**1.74**	**1.53**	**1.63**	**1.96**	**1.91**	**1.91**	**8.72**	**8.06**	**8.01**
28	ProxyGML	K=3, r=0.8	2.36	2.37	2.34	2.51	2.58	2.48	8.71	8.5	8.67

Table 4 Experimental results with different margins (m). The number of centers K is fixed at 3, and the scale factor r is set at 0.4. The performances are measured by EER (%), C_{avg} (%) and *min*DCF (*100)

ID	Loss	Parameter	Test-all			Test-3 s			Test-1 s		
			EER	C_{avg}	*min*DCF	EER	C_{avg}	*min*DCF	EER	C_{avg}	*min*DCF
1	MMAM	m=0.30, K=3, r=0.4	1.39	1.39	1.41	1.73	1.74	1.71	8.16	7.60	**7.30**
2	MMAM	m=0.35, K=3, r=0.4	1.40	1.27	1.37	1.54	1.52	1.52	8.87	8.07	8.48
3	MMAM	m=0.40, K=3, r=0.4	1.42	1.27	1.39	1.63	1.56	1.55	8.16	7.53	7.89
4	MMAM	m=0.45, K=3, r=0.4	1.49	1.36	1.43	1.58	1.57	1.53	8.11	7.48	7.82
5	MMAM	m=0.50, K=3, r=0.4	**1.35**	**1.23**	**1.31**	**1.48**	**1.47**	**1.45**	**7.78**	**7.28**	7.63
6	MMAM	m=0.55, K=3, r=0.4	1.50	1.33	1.45	1.72	1.70	1.68	8.85	8.07	8.48
7	MMAM	m=0.60, K=3, r=0.4	1.53	1.37	1.49	1.65	1.64	1.63	8.31	8.10	7.69
8	MMAM	m=0.65, K=3, r=0.4	1.57	1.44	1.56	1.75	1.75	1.73	8.42	7.77	8.21
9	MMAM	m=0.70, K=3, r=0.4	1.61	1.45	1.56	1.72	1.71	1.67	8.61	7.96	8.29
10	MMAM	m=0.75, K=3, r=0.4	1.77	1.60	1.73	1.86	1.86	1.82	8.39	7.89	8.19
11	MMAM	m=0.80, K=3, r=0.4	1.81	1.61	1.76	1.96	1.95	1.90	9.00	8.14	8.58

one hand, when $r < 0.4$, the similarity matrix optimized by the network selection is too small, which leads to poor performance. On the other hand, when $r > 0.4$, the similarity matrix optimized by the network selection is too large, and too many negative centers are introduced, also leading to poor performance.

In order to better illustrate the advantages of MMAM loss, the experimental results under different scale factors (r) of MMAM and ProxyGML are shown in Table 3, with the number of centers K fixed at 3. Also, it is obvious that in all cases, MMAM is better than ProxyGML.

5.1.3 Classification margin m

Since the value of angular margin m has a great influence on the recognition performance and the convergence of the training process, we study different margins m when K is fixed at 3, and the scale factor r is fixed at 0.4. As shown in Table 4, m increases by 0.05 each time, and the performance reaches best when $m = 0.5$. When m is less than 0.5, the training is relatively stable, and the experimental results do not change much. On the contrary, when m is greater than 0.5, the training process cannot converge well, resulting in poor results, especially when m is 0.8. At this time, the performance of the system is equivalent to that of the best baseline. Overall, when $K = 3$, $r = 0.4$ and $m = 0.5$, the performance is the best. Different angular margin penalty m has a fluctuating effect on different evaluation metrics, which verifies that m has a great impact on the recognition performance and training process.

5.2 Comparison with different baselines

We compare the performance of the proposed MMAM loss with seven different baselines from four aspects, including the training epochs, the experimental results, the DET curve, and the visual analysis with T-SNE.

5.2.1 The training epochs

The performances of the existing multi-center loss functions, Sub-Center (Eq. (13)), Softtriple (Eq. (15)), and MMAM (Eq. (4)), under test-all during the training process are shown in Fig. 5. For the convenience of comparison, we only compare and display the three best K-valued multi-center loss functions. From Fig. 5, we see that the convergence speed of Sub-center is the same as that of Softtriple, and their performances are similar. MMAM has both the fastest convergence speed and best performance. There is a segment between epochs 90 to 120 where the EER is almost stable for MMAM, which we speculate is caused by the relative stability of the similarity optimization between the selected samples and the multi-center during this segment.

5.2.2 Comparison of performance

Table 5 shows the experimental results of seven different baselines and our proposed MMAM. The first col-

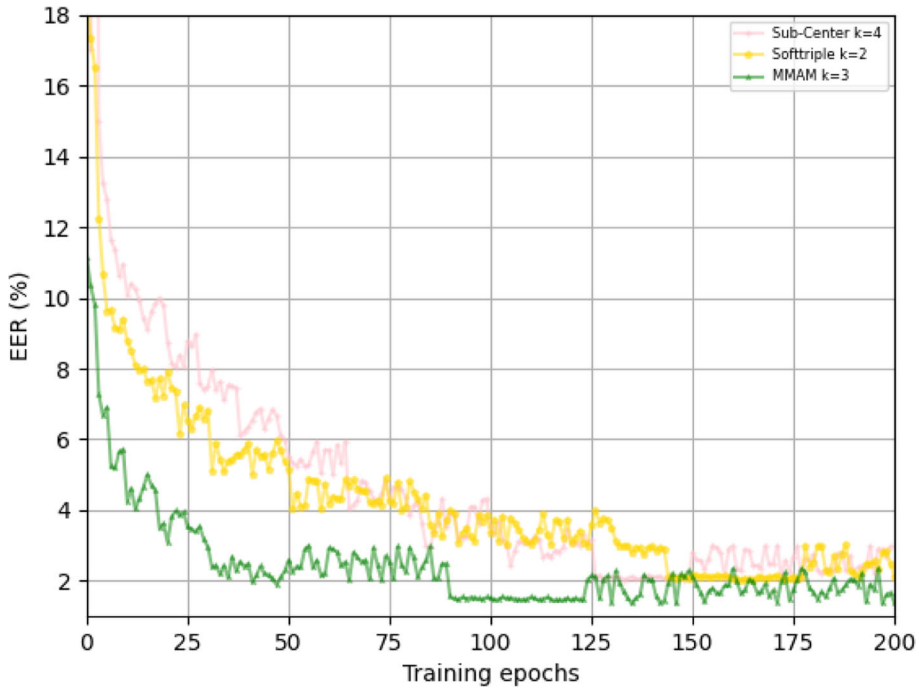


Fig. 5 MMAM converges faster with lower EER(%) values on AP17-OLR test-all

Table 5 Experimental results of seven different baselines and our proposed MMAM. "-" means there is no parameter

ID	Loss	Parameter	Test-all			Test-3 s			Test-1 s		
			EER	C_{avg}	*min*DCF	EER	C_{avg}	*min*DCF	EER	C_{avg}	*min*DCF
1	SoftMax	-	4.15	3.89	3.92	3.93	4.09	3.97	11.35	11.05	11.06
2	AAM-SoftMax	m=0.3	**1.83**	**1.79**	**1.92**	2.36	2.12	**2.34**	10.17	9.76	10.04
3	DAM-SoftMax	m=0.3	2.16	1.98	2.01	**2.23**	**2.08**	**2.34**	**9.44**	**8.76**	**9.09**
4	GE2E	-	3.60	3.51	3.61	3.99	3.71	3.25	14.23	13.56	13.73
5	AM-Centroid	m=0.3	3.43	3.26	3.33	3.87	3.45	3.48	13.27	12.12	12.66
6	Sub-center	m=0.3,K=4	**2.02**	**1.84**	2.12	**2.15**	**2.14**	2.14	9.13	8.69	9.19
7	Softtriple	m=0.3,K=2	2.03	1.87	**2.02**	2.26	2.33	**2.12**	**9.05**	**8.54**	**9.01**
8	MMAM(our)	m=0.3,r=0.4,K=3	1.39	1.39	1.41	1.73	1.74	1.71	8.16	7.60	**7.30**
9	MMAM(our)	m=0.5,r=0.4,K=3	**1.35**	**1.23**	**1.31**	**1.48**	**1.47**	**1.45**	**7.78**	**7.28**	7.63
-	SCL absolute improvement		0.48	0.56	0.61	0.75	0.61	0.89	1.65	1.47	1.79
-	SCL relative improvement		26.2%	31.3%	31.7%	33.6%	29.3%	38%	17.6%	16.9%	19.7%
-	MCL absolute improvement		0.67	0.61	0.71	0.67	0.66	0.67	1.27	1.26	1.71
-	MCL relative improvement		33.2%	33.2%	35.1%	31.2%	31.3%	31.6%	14%	14.8%	19%

umn "ID" marks each experiment's ID. The experimental results can be divided into three categories: Exp. 1–5 represent the baselines of the single-center loss; Exp. 6–7 represent the existing types of multi-center loss; Exp. 8–9 represent our proposed MMAM loss with different m. We use "SCL absolute improvement" and "SCL relative improvement" respectively to denote the absolute and relative improvements of our MMAM compared with the best experimental results for single-center loss (Exp. 1–5). Correspondingly, we use "MCL absolute improvement" and "MCL relative improvement" to respectively denote the absolute and relative improvements of our MMAM compared with the best experimental results for multi-center loss (Exp. 6–7).

In Fig. 6, MMAM is compared with the existing multi-center loss under different numbers of centers (K) and four different scale factors (r = 0.3, 0.5, 0.8, 1.0). Figure 6a, d, and g illustrates the results of EER on test-all, test-3 s, and test-1 s, respectively. Figure 6b, e, and h illustrates the results of C_{avg} on test-all, test-3 s, and test-1s, respectively. Figure 6c, f, and i illustrates the results of *minDCF* on test-all, test-3s, and test-1s, respectively. From Fig. 6, we see that MMAM can significantly reduce EER, C_{avg}, and minDCF, so it has better optimization than the existing types of multi-center loss.

5.2.3 The DET curve

The DET curve is another popular method for evaluating verification and identification systems. Compared with C_{avg}, EER, and *min*DCF, the DET curve represents the performance of all operating points, so it can more comprehensively evaluate the systems. The DET curves of various methods under different test conditions are plotted in Fig. 7. Figure 7a, b, and c represents the results

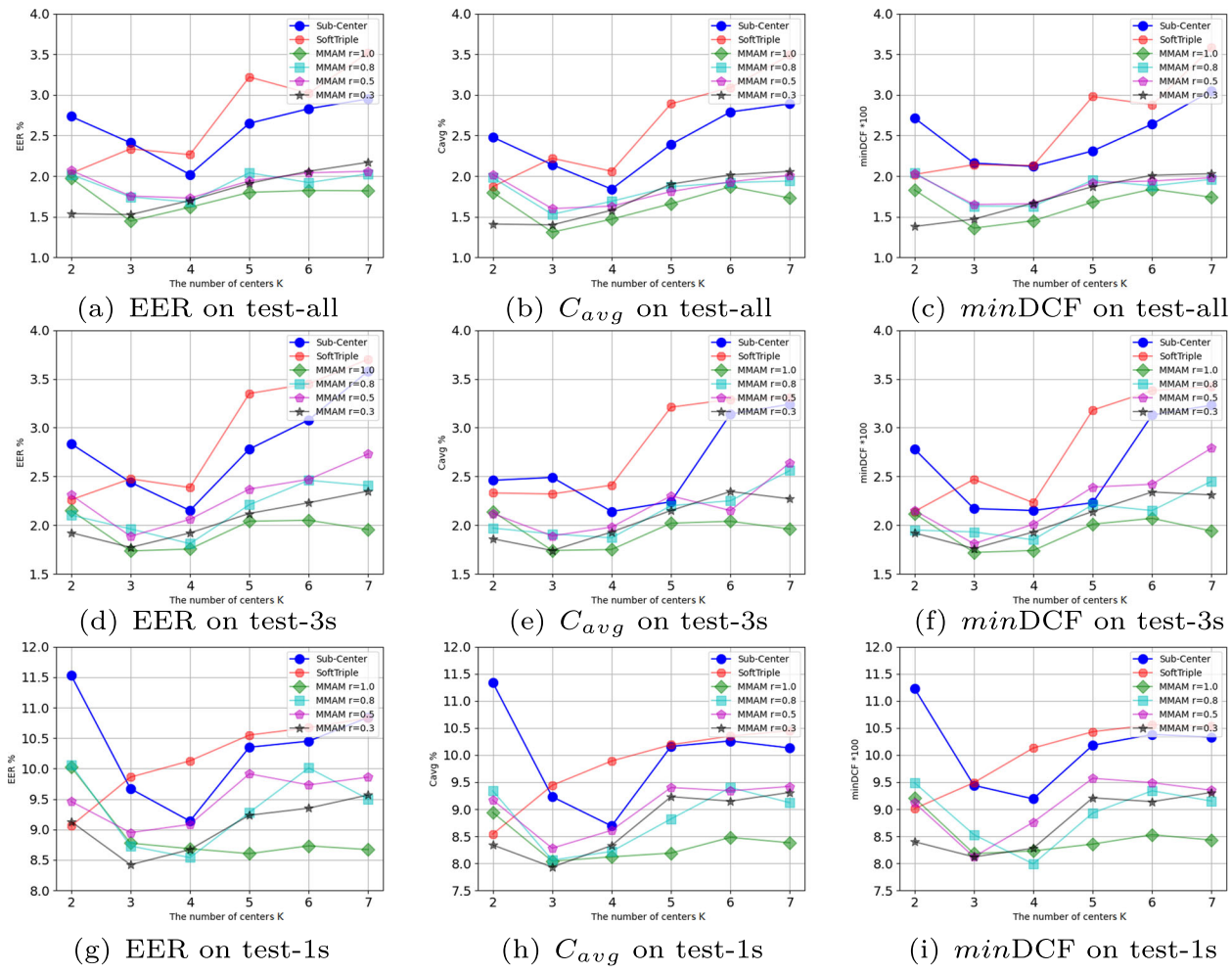
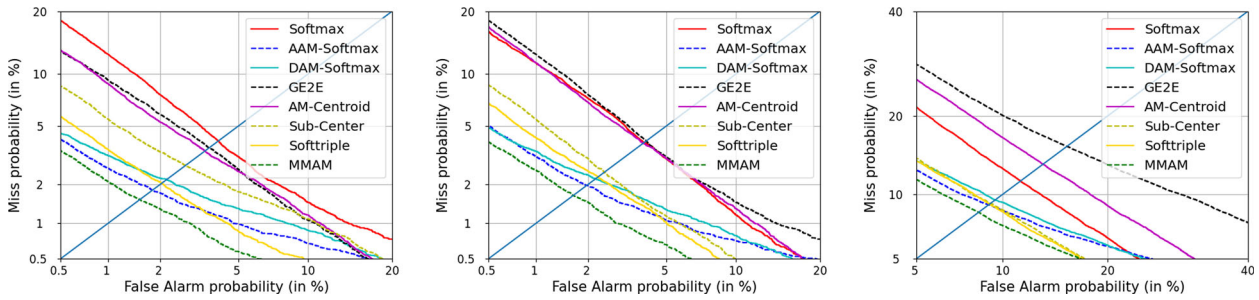
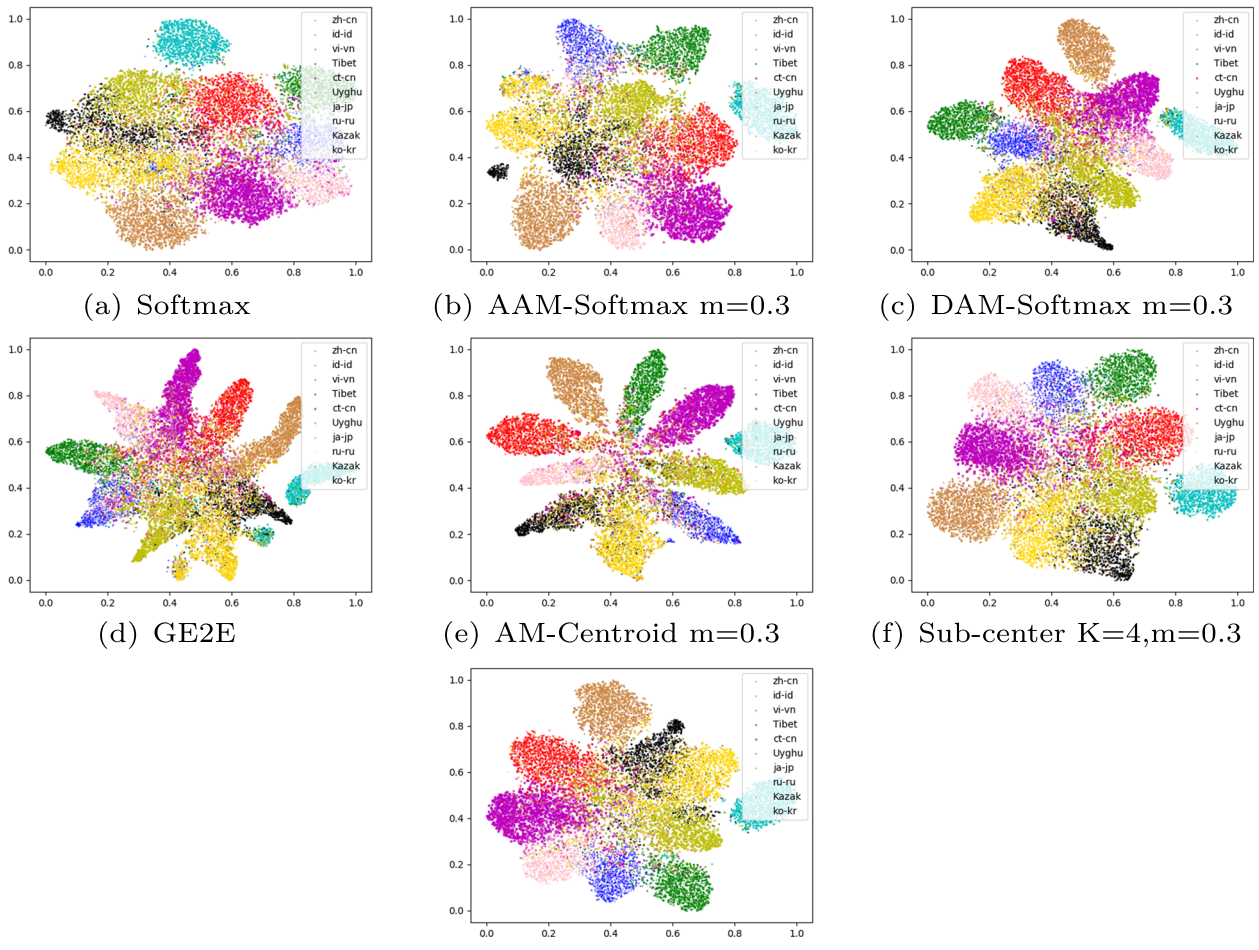


Fig. 6 Comparison with the existing types of multi-center loss under different number of centers (K) and four different scale factors (r = 0.3, 0.5, 0.8, 1.0)



(a) test-all (b) test-3s (c) test-1s

Fig. 7 DET curves of various methods under different test conditions

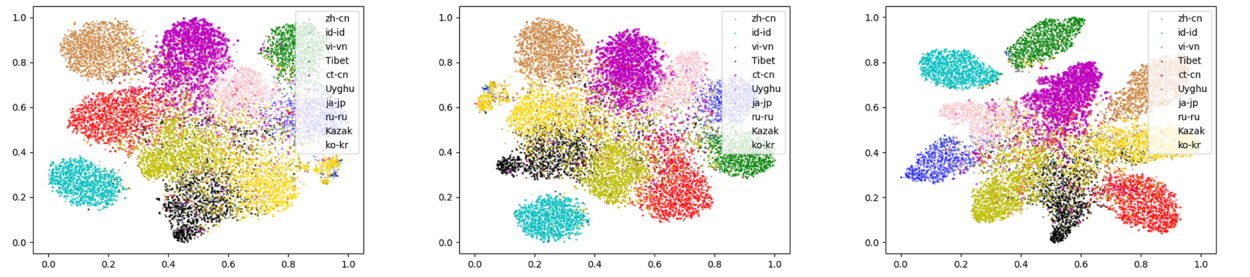


(a) Softmax (b) AAM-Softmax m=0.3 (c) DAM-Softmax m=0.3

(d) GE2E (e) AM-Centroid m=0.3 (f) Sub-center K=4,m=0.3

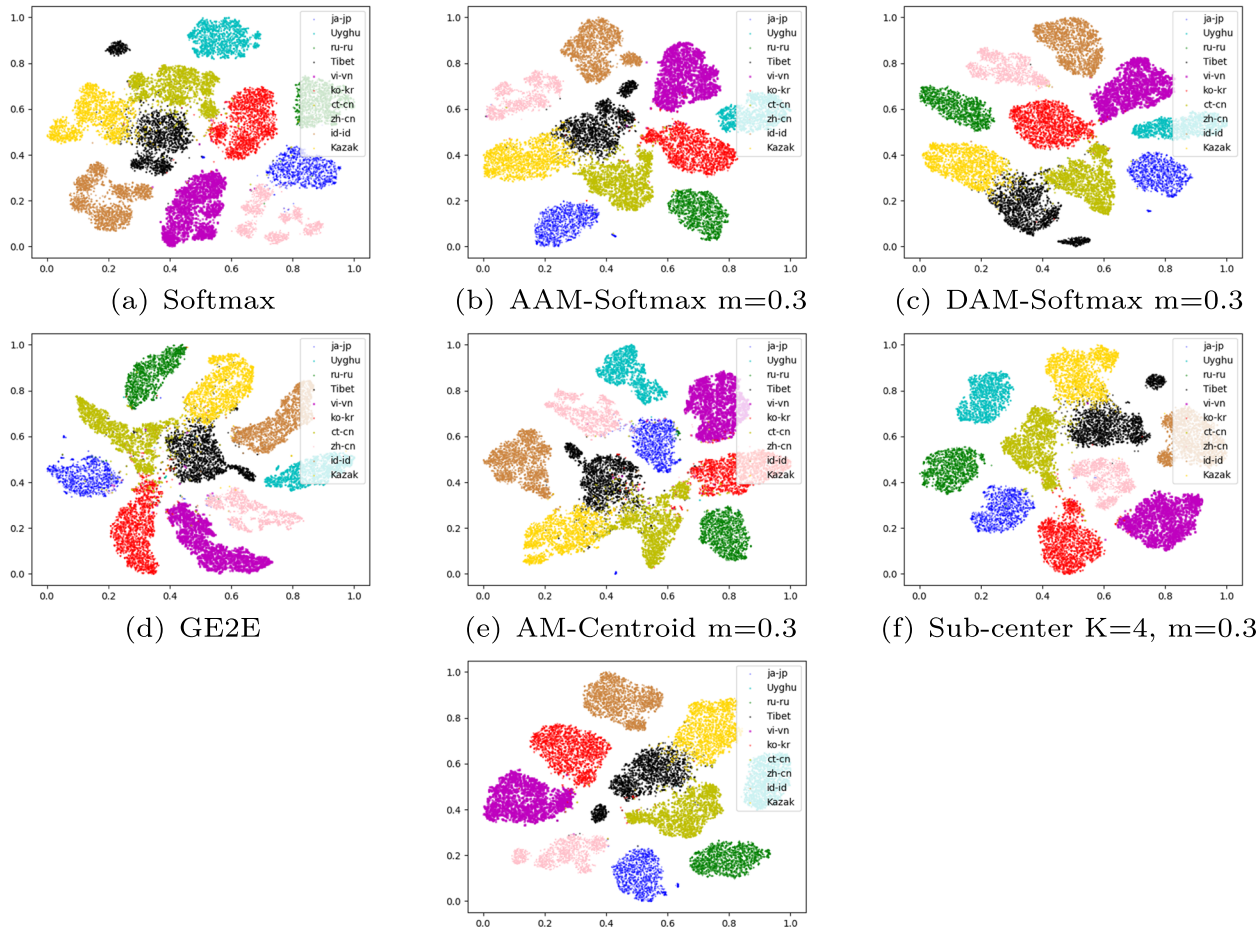
(g) Softtriple K=2,m=0.3

Fig. 8 Language embeddings of the baselines are plotted by t-SNE, under test-1s on AP17-OLR. Each shape/color represents a different language. **a** EER=11.35, C_{avg}=11.05, *min*DCF=11.09. **b** EER=10.17, C_{avg}=9.76, *min*DCF=10.04. **c** EER=9.44, C_{avg}=8.76, *min*DCF=9.09. **d** EER=14.23, C_{avg}=13.56, *min*DCF=13.73. **e** EER=13.27, C_{avg}=12.12, *min*DCF=12.66. **f** EER=9.13, C_{avg}=8.69, *min*DCF=9.19. **g** EER=9.05, C_{avg}=8.54, *min*DCF=9.01



(a) MMAM K=3,m=0.3,r=1.0 (b) MMAM K=3,m=0.3,r=0.4 (c) MMAM K=3,m=0.5,r=0.4

Fig. 9 Language embeddings of our proposed method are plotted by t-SNE, under test-1s on AP17-OLR. Each shape/color represents a different language. **a** EER=8.42, C_{avg}=7.93, *min*DCF=8.12. **b** EER=8.16, C_{avg}=7.60, *min*DCF=7.30. **c** EER=7.78, C_{avg}=7.28, *min*DCF=7.63



(a) Softmax (b) AAM-Softmax m=0.3 (c) DAM-Softmax m=0.3

(d) GE2E (e) AM-Centroid m=0.3 (f) Sub-center K=4, m=0.3

(g) Softtriple K=2, m=0.3

Fig. 10 Language embeddings of the baselines are plotted by t-SNE, under test-all on AP17-OLR. Each shape/color represents a different language. **a** EER=4.15, C_{avg}=3.89, *min*DCF=3.92. **b** EER=1.83, C_{avg}=1.79, *min*DCF=1.92. **c** EER=2.16, C_{avg}=1.98, *min*DCF=2.01. **d** EER=3.60, C_{avg}=3.51, *min*DCF=3.61. **e** EER=3.43, C_{avg}=3.26, *min*DCF=3.33. **f** EER=2.02, C_{avg}=1.84, *min*DCF=2.12. **g** EER=2.03, C_{avg}=1.87, *min*DCF=2.02

of test-all, test-3 s, and test-1 s, respectively. As we can see from the figure, the proposed MMAM exhibits the best performance at most points.

5.2.4 Visual analysis with T-SNE

To understand the model's ability to distinguish languages intuitively, we directly extract the embedding representations of different models from the test set. Since the visualization results of test-3 s and test-all are almost the same, we only show the visualization results of test-1 s and test-all on the AP17-OLR corpus. Then we visualize the embedding representations through t-SNE [46], which is a non-linear dimensionality reduction algorithm for visualizing high-dimensional data. To facilitate the observation of the difference between our proposed method and the baselines, we only list the visualization diagrams with the optimal parameters studied in Sections 5.1.1, 5.1.2, and 5.1.3. Similarly, the baselines also use the optimal parameters.

Language embeddings of the baselines and our proposed method under test-1s are plotted in Figs. 8 and 9, respectively. For single-center classification loss, Fig. 8a, b, and c represents the visual baselines of Eq. (6), Eq. (10), and Eq. (11), respectively, and Fig. 8d and e represents the visual baselines of Eq. (4) and Eq. (5), respectively. For multi-center loss, Fig. 8f and g represents the visual baselines of Eq. (13) and Eq. (15), respectively.

For our MMAM loss, Fig. 9a, b, and c represents the visual results of Eq. (26). Also, language embeddings of the baselines and our proposed method under test-all are plotted in Figs. 10 and 11, respectively. It can be seen that the visualizations of language embeddings by most of the single-center losses are not as good as those by the multi-center losses, and the multi-center losses do show traces of different centers. Although some single-center losses appear to distinguish all languages well compared with multi-center losses, their EERs are significantly higher than those of multi-center losses because scores and thresholds across languages determine the EER.

5.3 Comparison with state-of-the-arts

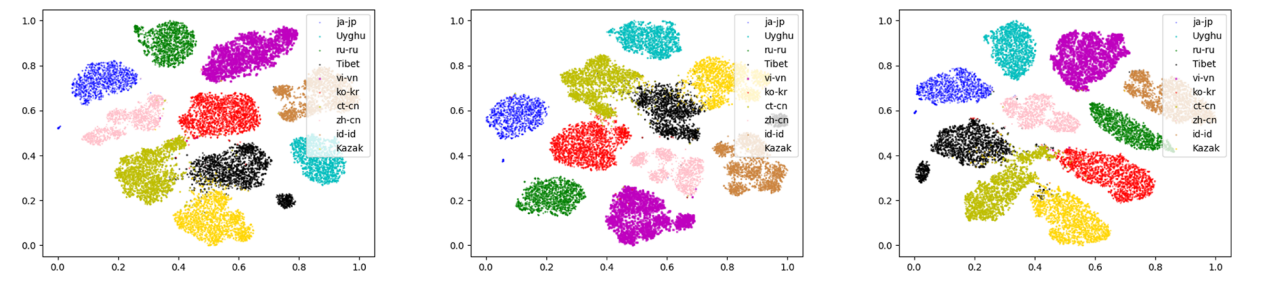
Table 6 compares the results of MMAM on the AP17-OLR test set with the current state-of-the-art results in terms of EER and C_{avg}. The first two lines are the two baselines released by the organizers of the AP17-OLR challenge [47]. LDA_HVS [48] uses a factorized hidden variability subspace (FHVS) learning technique for the adaptation of BLSTM RNNs model structure. AFs_ivector + AFs_xvector + AFs_TDNN [49] refers to the respectively established i-vector, x-vector and TDNN network-based fusion system based on articulatory features (AFs). Multi-head attention [50] represents the introduction of a multi-head attention mechanism in the self-attention network, assuming that each head can capture different information to distinguish languages. Wav2vec [51] uses Wav2vec 2.0 (a self-supervised framework for speech representation learning) to extend the self-supervised framework to speaker verification and language recognition.

Compared with these systems released above, MMAM shows obvious advantages under all test sets.

6 Conclusions and future work

The MMAM loss proposed in this paper has achieved the most advanced language recognition performance on the AP17-OLR corpus. The performance is improved by constraining the relationship between different centers and samples and between different centers and adding additional corner margins to the loss. EER, C_{avg}, *min*DCF are all greatly reduced. It is worth noting that in this paper, we choose the best performing hyper-parameters for all the methods based on the test set, which may risk overfitting.

Our future work is to design other novel functions different from Eq. (26) that can make the network more capable of identifying different languages. In theory, the MMAM loss can be used for language recognition and other classification tasks.



(a) MMAM K=3,m=0.3,r=1.0 (b) MMAM K=3,m=0.3,r=0.4 (c) MMAM K=3,m=0.5,r=0.4

Fig. 11 Language embeddings of our proposed method are plotted by t-SNE, under test-all on AP17-OLR. Each shape/color represents a different language. **a** EER=1.44, C_{avg}=1.31, *min*DCF=1.36. **b** EER=1.39, C_{avg}=1.39, *min*DCF=1.41. **c** EER=1.35, C_{avg}=1.23, *min*DCF=1.31

Table 6 Comparison with current state-of-the-art results on AP17-OLR in terms of EER and C_{avg}

System	Test-all		Test-3 s		Test-1 s	
	EER	C_{avg}	**EER**	C_{avg}	**EER**	C_{avg}
i-vecotr + PLDA [47]	5.96	5.86	-	-	17.51	17.46
TDNN [47]	11.31	10.34	-	-	14.04	12.82
LDA_HVS [48]	3.19	3.30	**3.79**	**3.77**	**8.47**	**8.82**
AFs_ivector + AFs_xvector + AFs_TDNN [49]	**1.92**	**1.84**	-	-	-	-
Multi-head attention [50]	5.65	-	-	-	-	-
Wav2vec [51]	3.47	3.10	-	-	12.02	11.58
MMAM(ours)	**1.35**	**1.23**	**1.48**	**1.47**	**7.78**	**7.28**
Absolute improvement	0.57	0.61	2.31	2.3	0.69	1.54
Relative improvement	29.6%	32.7%	60.9%	61%	8.1%	17.4%

Abbreviations
LR: Language recognition; OLR: Oriental language recognition; SCL: Single-center Loss; MCL: Multi-center Loss; MMAM: Masked multi-center angular margin; GE2E: Generalized end-to-end; AM-Centroid: Angular margin centroid; A-Softmax: Angular softmax; AM-Softmax: Additive margin softmax; Sub-center: Sub-center ArcFace; AAM-Softmax: Additive angular margin softmax; DAM-Softmax: Dynamic-additive-margin softmax; pm: Positive mask; LP: Label propagation; RLP: Reverse label propagation; ProxyGML: Proxy-based deep graph metric learning

Acknowledgements
Not applicable.

Authors' contributions
The first author mainly performed the experiments and wrote the paper, and the other authors reviewed and edited the manuscript. All of the authors discussed the final results. All of the authors read and approved the final manuscript.

Authors' information
Not applicable.

Funding
This work was supported by the Fundamental Research Funds for the Central Universities (grant number 2021ZY87).

Availability of data and materials
All data used in this study are included in the APSIPA 2017 Oriental Language Recognition (AP17-OLR) dataset [33, 34].

Declarations

Competing interests
The authors declare that they have no competing interests.

Author details
[1]School of Information Science and Technology, Beijing Forestry University, 35 Qing-Hua East Road, Beijing 100083, China. [2]Engineering Research Center for Forestry-oriented Intelligent Information Processing of National Forestry and Grassland Administration, Beijing Forestry University, 35 Qing-Hua East Road, Beijing 100083, China. [3]School of Information Science, Beijing Language and Culture University, 15 Xueyuan Road, Beijing 100083, China. [4]Institute for Integrated and Intelligent Systems, Griffith University, Nathan 4111, QLD, Australia.

Received: 16 December 2021 Accepted: 1 June 2022
Published online: 07 July 2022

References

1. H. Li, B. Ma, K. A. Lee, Spoken language recognition: from fundamentals to practice. Proc. IEEE. **101**(5), 1136–1159 (2013)
2. A. Waibel, P. Geutner, L. M. Tomokiyo, T. Schultz, M. Woszczyna, Multilinguality in speech and spoken language systems. Proc. IEEE. **88**(8), 1297–1313 (2000)
3. J. Yu, J. Zhang, in *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. Zero-resource language recognition (IEEE, New York, 2019), pp. 1907–1911
4. A. Nagrani, J. S. Chung, A. Zisserman, Voxceleb: a large-scale speaker identification dataset. arXiv preprint arXiv:1706.08612 (2017)
5. D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, S. Khudanpur, in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. X-vectors: Robust dnn embeddings for speaker recognition (IEEE, New York, 2018), pp. 5329–5333
6. S. J. Prince, J. H. Elder, in *2007 IEEE 11th International Conference on Computer Vision*. Probabilistic linear discriminant analysis for inferences about identity (IEEE, New York, 2007), pp. 1–8
7. M. Ravanelli, Y. Bengio, in *2018 IEEE Spoken Language Technology Workshop (SLT)*. Speaker recognition from raw waveform with sincnet (IEEE, New York, 2018), pp. 1021–1028
8. K. Okabe, T. Koshinaka, K. Shinoda, Attentive statistics pooling for deep speaker embedding. arXiv preprint arXiv:1803.10963 (2018)
9. D. Snyder, D. Garcia-Romero, G. Sell, A. McCree, D. Povey, S. Khudanpur, in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Speaker recognition for multi-speaker conversations using x-vectors (IEEE, New York, 2019), pp. 5796–5800
10. W. Cai, D. Cai, S. Huang, M. Li, in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Utterance-level end-to-end language identification using attention-based cnn-blstm (IEEE, New York, 2019), pp. 5991–5995
11. B. Padi, A. Mohan, S. Ganapathy, Towards relevance and sequence modeling in language recognition. IEEE/ACM Trans. Audio Speech Lang. Process. **28**, 1223–1232 (2020)
12. S. Ioffe, in *European Conference on Computer Vision*. Probabilistic linear discriminant analysis (Springer, Berlin, Heidelberg, 2006), pp. 531–542
13. W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, L. Song, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Sphereface: Deep hypersphere embedding for face recognition (IEEE, New York, 2017), pp. 212–220
14. S. Ramoji, P. Krishnan V, P. Singh, S. Ganapathy, Pairwise discriminative neural plda for speaker verification. arXiv preprint arXiv:2001.07034 (2020)
15. H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, W. Liu, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Cosface: Large margin cosine loss for deep face recognition (IEEE, New York, 2018), pp. 5265–5274
16. W. Xie, A. Nagrani, J. S. Chung, A. Zisserman, in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Utterance-level aggregation for speaker recognition in the wild (IEEE, New York, 2019), pp. 5791–5795
17. D. Garcia-Romero, D. Snyder, G. Sell, A. McCree, D. Povey, S. Khudanpur, in *INTERSPEECH*. x-vector dnn refinement with full-length recordings for speaker recognition (IEEE, New York, 2019), pp. 1493–1496
18. C. Luu, P. Bell, S. Renals, Dropclass and dropadapt: Dropping classes for deep speaker representation learning. arXiv preprint arXiv:2002.00453 (2020)
19. D. Snyder, J. Villalba, N. Chen, D. Povey, G. Sell, N. Dehak, S. Khudanpur, in *INTERSPEECH*. The jhu speaker recognition system for the voices 2019 challenge (IEEE, New York, 2019), pp. 2468–2472
20. F. Wang, J. Cheng, W. Liu, H. Liu, Additive margin softmax for face verification. IEEE Signal Proc. Lett. **25**(7), 926–930 (2018)
21. J. Deng, J. Guo, N. Xue, S. Zafeiriou, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Arcface: Additive angular margin loss for deep face recognition (IEEE, New York, 2019), pp. 4690–4699
22. D. Zhou, L. Wang, K. A. Lee, Y. Wu, M. Liu, J. Dang, J. Wei, in *Proc. Interspeech 2020*. Dynamic margin softmax loss for speaker verification (IEEE, New York, 2020), pp. 3800–3804
23. C. Zhang, K. Koishida, in *Interspeech*. End-to-end text-independent speaker verification with triplet loss on short utterances (IEEE, New York, 2017), pp. 1487–1491
24. V. Mingote, D. Castan, M. McLaren, M. K. Nandwana, A. O. Giménez, E. Lleida, A. Miguel, in *INTERSPEECH*. Language recognition using triplet neural networks (IEEE, New York, 2019), pp. 4025–4029
25. S. Chopra, R. Hadsell, Y. LeCun, in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Learning a similarity metric discriminatively, with application to face verification, vol. 1 (IEEE, New York, 2005), pp. 539–546
26. L. Wan, Q. Wang, A. Papir, I. L. Moreno, in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Generalized end-to-end loss for speaker verification (IEEE, New York, 2018), pp. 4879–4883
27. J. Deng, J. Guo, T. Liu, M. Gong, S. Zafeiriou, in *European Conference on Computer Vision*. Sub-center arcface: Boosting face recognition by large-scale noisy web faces (Springer, Heidelberg, 2020), pp. 741–757
28. Q. Qian, L. Shang, B. Sun, J. Hu, H. Li, R. Jin, in *Proceedings of the IEEE/CVF International Conference on Computer Vision*. Softtriple loss: Deep metric learning without triplet sampling (IEEE, New York, 2019), pp. 6450–6458
29. Y. Zhu, M. Yang, C. Deng, W. Liu, Fewer is more: A deep graph metric learning perspective using fewer proxies. arXiv preprint arXiv:2010.13636 (2020)
30. Y. Wei, J. Du, H. Liu, in *Proc. Interspeech 2020*. Angular margin centroid loss for text-independent speaker recognition (IEEE, New York, 2020), pp. 3820–3824

31. A. Iscen, G. Tolias, Y. Avrithis, O. Chum, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Label propagation for deep semi-supervised learning (IEEE, New York, 2019), pp. 5070–5079
32. W. Liu, J. Wang, S.-F. Chang, Robust and scalable graph-based semisupervised learning. Proc. IEEE. **100**(9), 2624–2638 (2012)
33. Z. Tang, D. Wang, Y. Chen, Q. Chen, in *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. Ap17-olr challenge: Data, plan, and baseline (IEEE, New York, 2017), pp. 749–753
34. D. Wang, L. Li, D. Tang, Q. Chen, in *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*. Ap16-ol7: A multilingual database for oriental languages and a language recognition baseline (IEEE, New York, 2016), pp. 1–5
35. Z. Ma, H. Yu, Language identification with deep bottleneck features. arXiv preprint arXiv:1809.08909 (2018)
36. T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, S. Khudanpur, in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. A study on data augmentation of reverberant speech for robust speech recognition (IEEE, New York, 2017), pp. 5220–5224
37. D. Snyder, G. Chen, D. Povey, Musan: A music, speech, and noise corpus. arXiv preprint arXiv:1510.08484 (2015)
38. W. Cai, J. Chen, J. Zhang, M. Li, On-the-fly data loader and utterance-level aggregation for speaker and language recognition. IEEE/ACM Trans Audio Speech Lang. Process. **28**, 1038–1051 (2020)
39. Z. Qi, Y. Ma, M. Gu, in *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. A study on low-resource language identification (IEEE, New York, 2019), pp. 1897–1902
40. 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. arXiv preprint arXiv:1912.01703 (2019)
41. D. P. Kingma, J. Ba, Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
42. B. Desplanques, J. Thienpondt, K. Demuynck, Ecapa-tdnn: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification. arXiv preprint arXiv:2005.07143 (2020)
43. Z. Gao, Y. Song, I. McLoughlin, P. Li, Y. Jiang, L.-R. Dai, in *INTERSPEECH*. Improving aggregation and loss function for better embedding learning in end-to-end speaker verification system (IEEE, New York, 2019), pp. 361–365
44. D. Hendrycks, K. Gimpel, Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415 (2016)
45. S. Ioffe, C. Szegedy, in *International Conference on Machine Learning*. Batch normalization: Accelerating deep network training by reducing internal covariate shift (PMLR, New York, 2015), pp. 448–456
46. L. Van der Maaten, G. Hinton, Visualizing data using t-sne. J. Mach. Learn. Res. **9**(11) (2008)
47. Z. Tang, D. Wang, Q. Chen, AP18-OLR challenge: Three tasks and their baselines. CoRR. **abs/1806.00616** (2018). 1806.00616. Accessed 1 Nov 2018
48. S. Fernando, V. Sethu, E. Ambikairajah, in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Factorized hidden variability learning for adaptation of short duration language identification models (IEEE, New York, 2018), pp. 5204–5208
49. J. Yu, M. Guo, Y. Xie, J. Zhang, in *2019 International Conference on Asian Language Processing (IALP)*. Articulatory features based tdnn model for spoken language recognition (IEEE, New York, 2019), pp. 308–312
50. R. K. Vuddagiri, T. Mandava, H. K. Vydana, A. K. Vuppala, in *2019 Twelfth International Conference on Contemporary Computing (IC3)*. Multi-head self-attention networks for language identification (IEEE, New York, 2019), pp. 1–5
51. Z. Fan, M. Li, S. Zhou, B. Xu, Exploring wav2vec 2.0 on speaker verification and language identification. arXiv preprint arXiv:2012.06185 (2020)

Publisher's Note