PARTIAL AUC OPTIMIZATION BASED DEEP SPEAKER EMBEDDINGS WITH CLASS-CENTER LEARNING FOR TEXT-INDEPENDENT SPEAKER VERIFICATION

Zhongxin Bai, Xiao-Lei Zhang, and Jingdong Chen

Center of Intelligent Acoustics and Immersive Communications and
School of Marine Science and Technology, Northwestern Polytechnical University
zxbai@mail.nwpu.edu.cn, xiaolei.zhang@nwpu.edu.cn, jingdongchen@ieee.org

ABSTRACT

Deep embedding based text-independent speaker verification has demonstrated superior performance to traditional methods in many challenging scenarios. Its loss functions can be generally categorized into two classes, i.e., verification and identification. The verification loss functions match the pipeline of speaker verification, but their implementations are difficult. Thus, most state-of-the-art deep embedding methods use the identification loss functions with softmax output units or their variants. In this paper, we propose a verification loss function, named the maximization of partial area under the Receiver-operating-characteristic (ROC) curve (pAUC), for deep embedding based text-independent speaker verification. We also propose a class-center based training trial construction method to improve the training efficiency, which is critical for the proposed loss function to be comparable to the identification loss in performance. Experiments on the Speaker in the Wild (SITW) and NIST SRE 2016 datasets show that the proposed pAUC loss function is highly competitive with the state-of-the-art identification loss functions.

Index Terms— speaker verification, pAUC optimization, speaker centers, verification loss

1. INTRODUCTION

Text independent speaker verification aims to verify whether an utterance is pronounced by a hypothesized speaker according to his/her pre-recorded utterances without limiting the speech contents. The state-of-the-art text-independent speaker verification systems [1–4] use deep neural networks (DNNs) to project speech recordings with different lengths into a common low dimensional embedding space where the speakers' identities are represented. Such a method is called *deep embedding*, where the embedding networks have three key components—network structure [1, 3, 5–7], pooling layer [1, 8–12], and loss function [13–17]. This paper focuses on the last part, i.e., the loss functions.

Generally, there are two types of loss functions, i.e., identification and verification loss functions. The former is mainly cross-entropy loss with softmax output units or softmax variants [3, 18] such as ASoftmax, AMSoftmax, and ArcSoftmax whose role is to change the linear transform of the softmax function to a cosine function with a margin controlling the distance between speakers's spaces. Different from [3, 18], which conducts multi-class classification from a single classifier, [19] conducts one-versus-all classifica-

tion with multiple binary classifiers. In comparison, the verification loss functions mainly consist of pairwise- or triplet-based loss functions [5, 15–17, 20, 21]. The fundamental difference between the verification and identification loss functions is that the former needs to construct pairwise or triplet training trials, which imitates the enrollment and test stages of speaker verification. Although this imitation matches the pipeline of speaker verification ideally, its implementation faces many difficulties in practice. One of those is that the number of all possible training trials increase cubically or quadratically with the number of training utterances. One way to circumvent this issue is through selecting part of the informative training trials that are difficult to be discriminated. But finding a good selection method is a challenging task. Moreover, the optimization process with the verification function is not so stable in comparison with that with the identification loss. As a result, the state-of-the-art deep embedding methods optimize the identification loss directly, or fine-tune an identification loss based DNN with the verification loss [21, 22].

Despite of some disadvantages, a great property of the verification loss functions is that the training process is consistent with the evaluation procedure, which make it more proper for speaker verification in comparison with the identification loss. In [23], we proposed a new verification loss, named *maximizing partial area under the ROC curve* (pAUC), for training back-ends. The motivation and advantages of the pAUC maximization over other verification loss functions were shown in [23].

In this paper, we extend the work in [23]. Motivated from the advantages of the identification loss, we improve the procedure of its training trial construction in [23] by a *class-center learning* method. This approach first learns the centers of classes of the training speakers, and then uses the class-centers as enrollments to construct training trials at each optimization epoch of the pAUC deep embedding. Experiments are conducted on the Speaker in the Wild (SITW) and NIST SRE 2016 datasets. Results demonstrated that the proposed pAUC deep embedding is highly competitive in performance with the state-of-the-art identification loss based deep embedding methods with the Softmax and ArcSoftmax output units. Note that a very recent work proposed at the same time as our work in [23] maximizes the area under the ROC curve (AUC) for text-dependent speaker verification [17]. It can be shown that AUC is a particular case of pAUC and experimental results show the pAUC deep embedding outperforms the AUC deep embedding significantly.

2. DEEP EMBEDDING VIA PAUC OPTIMIZATION

2.1. Objective function

In the training stage, the DNN model of the pAUC deep embedding system outputs a similarity score s for a pair of input utterances. In the test stage, it outputs an embedding vector $\mathbf{h}$ from the top hidden

layer for each input utterance, and uses $\mathbf{h}$ for verification. Because the gradient at the output layer can be transferred to the hidden layers by backpropagation, we focus on presenting the pAUC optimization at the output layer as follows:

In the training stage, we construct a pairwise training set $\mathcal{T} = \{(\mathbf{x}_n, \mathbf{y}_n; l_n) | n = 1, 2, \cdots, N\}$ where $\mathbf{x}_n$ and $\mathbf{y}_n$ are the representations of two utterances at the output layer of the DNN model, and l_n is the ground-truth label indicating the similarity of $\mathbf{x}_n$ and $\mathbf{y}_n$ (if $\mathbf{x}_n$ and $\mathbf{y}_n$ come from the same speaker, $l_n = 1$; otherwise, $l_n = 0$). Given a soft similarity function $f(\cdot)$, we obtain a similarity score of $\mathbf{x}_n$ and $\mathbf{y}_n$, denoted as $s_n = f(\mathbf{x}_n, \mathbf{y}_n)$ where $s_n \in \mathbb{R}$. The hard decision of the similarity of $\mathbf{x}_n$ and $\mathbf{y}_n$ is:

$$\hat{l}_n = \left\{ \begin{array}{ll} 1, & \text{if } s_n \geq \theta \\ 0, & \text{otherwise} \end{array} \right., \forall n = 1, \ldots, N, \quad (1)$$

where θ is a decision threshold. Given a fixed value of θ, we are able to compute a true positive rate (TPR) and a false positive rate (FPR) from $\hat{l}_n, \forall n = 1, \ldots, N$. TPR is defined as the ratio of the positive trials (i.e. $l_n = 1$) that are correctly predicted (i.e. $\hat{l}_n = 1$) over all positive trials. FPR is the ratio of the negative trials (i.e. $l_n = 0$) that are wrongly predicted (i.e. $\hat{l}_n = 1$) over all negative trials. Varying θ gives a series of $\{\text{TPR}(\theta), \text{FPR}(\theta)\}$, which form an ROC curve as illustrated in Fig.1. The gray area in Fig.1 illustrates how pAUC is defined. Specifically, it is defined as the area under the ROC curve when the value of FPR is between $[\alpha, \beta]$, where α and β are two hyperparameters. To calculate pAUC, we first construct two sets $\mathcal{P} = \{(s_i, l_i = 1) | i = 1, 2, \cdots, I\}$ and $\mathcal{N} = \{(s_j, l_j = 0) | j = 1, 2, \cdots, J\}$, where $I + J = N$. We then obtain a new subset $\mathcal{N}_0$ from $\mathcal{N}$ by adding the constraint $\text{FPR} \in [\alpha, \beta]$ to $\mathcal{N}$ via following steps:

1) $[\alpha, \beta]$ is replaced by $[j_\alpha/J, j_\beta/J]$ where $j_\alpha = \lceil J\alpha \rceil + 1$ and $j_\beta = \lfloor J\beta \rfloor$ are two integers.

2) $\{s_j\}_{\forall j: s_j \in \mathcal{N}}$ are sorted in descending order, where the operator $\forall a : b$ denotes that every a that satisfies the condition b will be included in the computation.

3) $\mathcal{N}_0$ is selected as the set of the samples ranked from the top j_αth to j_βth positions of the resorted $\{s_j\}_{\forall j: s_j \in \mathcal{N}}$, denoted as $\mathcal{N}_0 = \{(s_k, l_k = 0) | k = 1, 2, \cdots, K\}$ with $K = j_\beta - j_\alpha + 1$.

Finally, pAUC is calculated as a normalized AUC over $\mathcal{P}$ and $\mathcal{N}_0$:

$$\text{pAUC} = 1 - \frac{1}{IK} \sum_{\forall i: s_i \in \mathcal{P}} \sum_{\forall k: s_k \in \mathcal{N}_0} \left[\mathbb{I}(s_i < s_k) + \frac{1}{2}\mathbb{I}(s_i = s_k) \right] \quad (2)$$

where $\mathbb{I}(\cdot)$ is an indicator function that returns 1 if the statement is true, and 0 otherwise. However, directly optimizing (2) is NP-hard. A common way to overcome the NP-hard problem is to relax the indicator function by a hinge loss function [23]:

$$\ell_{\text{hinge}}(z) = \max(0, \delta - z) \quad (3)$$

where $z = s_i - s_k$, and $\delta > 0$ is a tunable hyper-parameter. Because the gradient of (3) is a constant with respect to z, it does not reflect the difference between two samples that cause different errors. Motivated by the loss function of the least-squares support vector machine, we replace (3) by (4),

$$\ell'_{\text{hinge}}(z) = \max(0, \delta - z)^2 \quad (4)$$

Substituting (4) into (2) and changing the maximization problem (2) into an equivalent minimization, one can derive the following pAUC
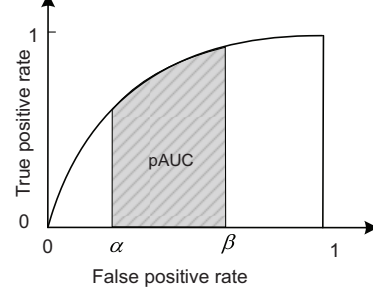

Fig. 1. Illustrations of the ROC curve, AUC, and pAUC.

optimization objective:

$$\min \frac{1}{IK} \sum_{\forall i: s_i \in \mathcal{P}} \sum_{\forall k: s_k \in \mathcal{N}_0} \max(0, \delta - (s_i - s_k))^2 \quad (5)$$

The minimization of (5) needs to define a similarity function f as illustrated in (1). We adopt the cosine similarity:

$$s_n = f(\mathbf{x}_n, \mathbf{y}_n) = \frac{\mathbf{x}_n^T \mathbf{y}_n}{||\mathbf{x}_n|| ||\mathbf{y}_n||} \quad (6)$$

where $|| \cdot ||$ is the ℓ_2-norm operator.

2.2. Pairwise training set construction

Suppose that we have a training set $\mathcal{X} = \{\mathbf{x}_{uv} | u = 1, \cdots, U; v = 1, \cdots, V_u\}$, where u and v represent the v_{th} utterance of the u_{th} speaker, U is the total number of the speakers and V_u is the utterance number of the u_{th} speaker. If we construct $\mathcal{T}$ by using all of the above $\sum_{u=1}^{U} V_u$ utterances, the size of $\mathcal{T}$ would be enormous. So, in this work we propose the following two kinds of methods to construct the training set.

1) Random sampling: We construct a set $\mathcal{T}^t$ at each mini-batch iteration of the DNN training by a random sampling strategy as follows. We first randomly select t speakers from $\mathcal{X}$, then randomly select two utterances from each of the selected speakers, and finally construct $\mathcal{T}^t$ by a full permutation of the $2t$ utterances. It is easy to see that $\mathcal{T}^t$ contains t true training trials and $[t(2t-1) - t]$ imposter training trials.

2) Class-center learning: We construct a set $\mathcal{T}^{t_1}$ at each mini-batch iteration of the DNN training by a class-center learning algorithm as follows. Motivated from the identification loss, which learns a class center for each speaker, we assign a class center $\mathbf{w}$ to each speaker, denoted as $\{\mathbf{w}_u\}_{u=1}^{U}$. At each iteration, we first select t_1 utterances randomly, and then combine them with $\{\mathbf{w}_u\}_{u=1}^{U}$ in pairwise to form $\mathcal{T}^{t_1}$, which contains t_1 true training trials and $(t_1 U - t_1)$ imposter training trials. In the training stage, $\{\mathbf{w}_u\}_{u=1}^{U}$ is randomly initialized and updated at each iteration by the back propagation.

In comparison with the random sampling strategy, the class-center learning algorithm aggregates all training utterances of a speaker to its class-center. The class-centers should be more discriminative and robust than the random samples. Hence, training on $\mathcal{T}^{t_1}$ should be easier and more consistent than training on $\mathcal{T}^t$. Note that the class-center learning algorithm requires the ground-truth speaker identities in training, which is the same as the identification loss based speaker verification methods, because the class-centers rely on the ground-truth speaker identities. So, this algorithm is not applicable if the training labels are speaker trials instead of speaker identities.

3. CONNECTIONS TO OTHER LOSS FUNCTIONS

3.1. Connection to cross-entropy minimization with softmax

The softmax classifier is presented as

$$\mathcal{L}_{\text{softmax}} = -\frac{1}{R} \sum_{r=1}^{R} \log \frac{e^{\mathbf{w}_{y_r}^{\mathsf{T}} \mathbf{x}_r + b_{y_r}}}{\sum_{u=1}^{U} e^{\mathbf{w}_u^{\mathsf{T}} \mathbf{x}_r + b_u}}, \qquad (7)$$

where $\mathbf{x}_r$ represents the r-th training sample, y_r is its ground truth speaker identity, $\mathbf{w}_u$ is the u-th column of the weights of the output layer, b_u is the bias term, and R is the total number of the training samples.

The proposed method has a close connection to the softmax classifier on the class-center learning method. The objective function (5) aims to maximize the pAUC of the pairwise training set $\mathcal{T}^{t_1}$ at a mini-batch iteration, while the cross-entropy minimization with softmax aims to classify the t_1 utterances that are used to construct the $\mathcal{T}^{t_1}$. The class centers $\{\mathbf{w}_u\}_{u=1}^{U}$ are used for constructing $\mathcal{T}^{t_1}$ in the pAUC optimization, and used as the parameters of the softmax classifier in (7).

3.2. Connection to triplet loss

Triplet loss requires that the utterances from the same speaker are closer than those from different speakers in a triplet trial [21], i.e.,

$$f(\mathbf{x}^a, \mathbf{x}^p) - f(\mathbf{x}^a, \mathbf{x}^n) > \sigma \qquad (8)$$

where $\mathbf{x}^a$, $\mathbf{x}^p$, and $\mathbf{x}^n$ represent the anchor, positive, and negative samples respectively, and σ is a tunable hyperparameter.

The difference between pAUC and triplet loss lies in the following two aspects. First, according to (2) and (3), the relative constraint of pAUC can be written as

$$s_i - s_k > \delta \qquad (9)$$

where s_i and s_k are constructed by four utterances. In other words, the relative constraint of pAUC is tetrad, which matches the pipeline of speaker verification, while the relative constraint (8) is triplet. Second, pAUC is able to pick difficult training trials from the exponentially large number of training trials during the training process, while the triplet loss lacks such an ability.

3.3. Connection to AUC maximization

The AUC optimization [17] is a special case of the pAUC optimization with $\alpha = 0$ and $\beta = 1$. It is known that the performance of a speaker verification system is determined on the discriminability of the difficult trials. However, the AUC optimization is trained on $\mathcal{P}$ and $\mathcal{N}$. The two sets may contain many easy trials, which hinders the focus of the AUC optimization on solving the difficult problem. In comparison, the pAUC optimization with a small β is able to select difficult trials at each mini-batch iteration. Experimental results in the following Section also demonstrate that the pAUC optimization is more effective than the AUC optimization.

4. EXPERIMENTS

4.1. Data sets

We conducted two experiments with the Kaldi recipes [24] of *"/egs/sitw/v2"* and *"/egs/sre16/v2"* respectively. Because the sampling rates of the training data of the first and second recipes are 16 kHz and 8 kHz respectively, we name the two experiments as the

16KHZ system and *8KHZ system* accordingly for simplicity. In the 16KHZ system, the deep embedding models were trained using the speech data extracted from the combined VoxCeleb 1 and 2 corpora [22,25]. The back-ends were trained on a subset of the augmented VoxCeleb data, which contains 200000 utterances. The evaluation was conducted on the Speakers in the Wild (SITW) [26] dataset, which has two evaluation tasks–Dev.Core and Eval.Core. In the 8KHZ system, the training data for the deep embedding models consist of Switchboard Cellular 1 and 2, Switchboard 2 Phase 1, 2, and 3, NIST SREs from 2004 to 2010, and Mixer 6. The back-ends were trained with the NIST SREs along with Mixer 6. The evaluation data is the Cantonese language of NIST SRE 2016. We followed Kaldi to train the PLDA adaptation model via the unlabeled data of NIST SRE 2016.

4.2. Experimental setup

We compared five loss functions, which are the the cross-entropy loss with softmax (Softmax) and additive angular margin softmax (ArcSoftmax) [18], random sampling based pAUC optimization (pAUC-R), class-center learning based pAUC optimization (pAUC-L), and class-center learning based AUC optimization (AUC-L), respectively. Besides, we also cited the published results in the Kaldi source code, denoted as Softmax (Kaldi), for comparison.

We followed kaldi for the data preparation including the MFCC extraction, voice activity detection, and cepstral mean normalization. For all comparison methods, the deep embedding models were trained with the same data augmentation strategy and DNN structure (except the output layer) as those in [1]. They were implemented by Pytorch with the Adam optimizer. The learning rate was set to 0.001 without learning rate decay and weight decay. The batch-size was set to 128, except for pAUC-R whose batch-size was set to 512. The deep embedding models in the 16KHZ and 8 KHZ systems were trained with 50 and 300 epochs respectively. We adopted the L-DA+PLDA back-end for all comparison methods. The dimension of LDA was set to 256 for the pAUC-L, AUC-L and ArcSoftmax of the 16KHZ system, and was set to 128 for the other evaluations.

For pAUC-R, the hyperparameter α was fixed to 0; the hyperparameter β was set to 0.01 for the 16KHZ system and 0.1 for the 8KHZ system; the hyperparameter δ was set to 1.2 for the 16KHZ system and 0.4 for the 8KHZ system;. For pAUC-L, α and δ were set the same as those of pAUC-R; β was set to 0.001 for the 16KHZ system and 0.01 for the 8KHZ system. For ArcSoftmax, we adopted the best hyperparameter setting as that in [18].

The evaluation metrics include the equal error rate (EER), minimum detection cost function with $P_{\text{target}} = 10^{-2}$ (DCF10^{-2}) and $P_{\text{target}} = 10^{-3}$ (DCF10^{-3}) respectively, and detection error tradeoff (DET) curve.

4.3. Main results

The experimental results on SITW and NIST SRE 2016 are listed in Tables 1 and 2 respectively. From the results of Softmax, one can see that our implementation of Softmax via Pytorch achieves similar performance with the Kaldi's implementation, which validates the correctness of our deep embedding model. We also observed that, if the stochastic gradient descent algorithm was carefully tuned with suitable weight decay, the performance can be further improved, which will not be reported in this paper due to the length limitation. Moreover, ArcSoftmax significantly outperforms Softmax, which corroborates the results in [3, 18].

pAUC-L reaches an EER score of over 25% and 10% relatively lower than Softmax in the two experimental systems respectively. It also achieves comparable performance to the state-of-the-art

Table 1. Results on SITW.

Name	Loss	EER(%)	DCF10^{-2}	DCF10^{-3}
Dev.Core	Softmax (Kaldi)	3.0	-	-
	Softmax	3.04	0.2764	**0.4349**
	ArcSoftmax	**2.16**	**0.2565**	0.4501
	pAUC-R	3.20	0.3412	0.5399
	pAUC-L	**2.23**	**0.2523**	**0.4320**
	AUC-L	4.27	0.4474	0.6653
Eval.Core	Softmax (Kaldi)	3.5		
	Softmax	3.45	0.3339	**0.4898**
	ArcSoftmax	**2.54**	**0.3025**	0.5142
	pAUC-R	3.74	0.3880	0.5797
	pAUC-L	**2.56**	**0.2949**	0.5011
	AUC-L	4.76	0.5005	0.7155

Table 2. Results on the Cantonese language of NIST SRE 2016.

Back-end	Loss	EER(%)	DCF10^{-2}	DCF10^{-3}
No-adaptation	Softmax (Kaldi)	7.52	-	-
	Softmax	6.76	0.5195	0.7096
	ArcSoftmax	**5.59**	**0.4640**	**0.6660**
	pAUC-R	15.25	0.8397	0.9542
	pAUC-L	**6.01**	**0.5026**	**0.7020**
	AUC-L	7.92	0.5990	0.8072
Adaptation	Softmax (Kaldi)	4.89	-	-
	Softmax	4.94	0.4029	0.5949
	ArcSoftmax	**4.13**	**0.3564**	**0.5401**
	pAUC-R	8.65	0.6653	0.8715
	pAUC-L	**4.25**	**0.3704**	**0.5471**
	AUC-L	5.36	0.4439	0.6480
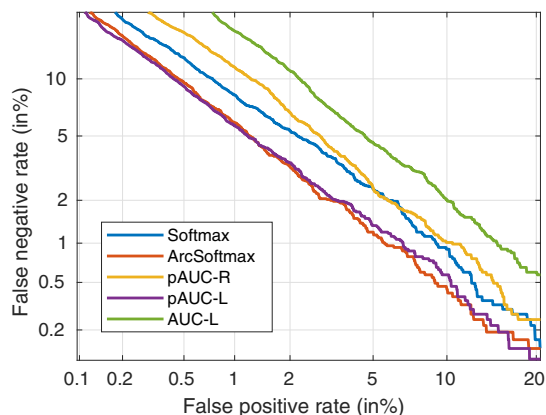
Fig. 2. DET curves of the comparison methods on the Eval.Core. task.

ArcSoftmax, which demonstrates that the verification loss functions are comparable to the identification loss functions in performance. pAUC-L also outperforms pAUC-R significantly, which demonstrates that the class-center learning algorithm is a better training set construction method than the random sampling strategy. It is also seen that AUC-L cannot reach the state-of-the-art performance. The DET curves of the comparison methods are plotted in Fig. 2. From the figure, we observe that the DET curve of pAUC-L is close to that of ArcSoftmax, both of which perform the best among the studied methods.

Table 3. EER (%) of pAUC-L on the SITW DEV.Core task, where the keyword "NaN" means that the optimization process encounters numerical problems.

	$\delta = 0.0$	$\delta = 0.4$	$\delta = 0.8$	$\delta = 1.2$	$\delta = 1.6$
$\beta = 0.0001$	-	NaN	-	-	-
$\beta = 0.001$	4.69	3.04	2.71	2.58	2.81
$\beta = 0.01$	4.57	3.17	2.93	3.00	2.81
$\beta = 0.1$	-	3.14	-	-	-
$\beta = 1$	-	4.12	-	-	-

Table 4. EER (%) of pAUC-L on the Cantonese language.

	$\delta = 0.0$	$\delta = 0.4$	$\delta = 0.8$	$\delta = 1.2$	$\delta = 1.6$
$\beta = 0.001$	24.07	8.29	9.70	9.58	10.85
$\beta = 0.01$	11.74	7.40	7.52	7.64	7.38
$\beta = 0.1$	12.57	8.54	9.07	9.30	9.94

4.4. Effects of hyperparameters on performance

This subsection investigates the effects of the hyperparameters of pAUC-L on performance. The hyperparameters were selected via $\alpha = 0$, $\beta = (0, 1]$, and $\delta = [0, 2)$. To accelerate the evaluation, we trained a pAUC-L model with 50 epochs using one quarter of the training data at each hyperparameter setting in the 16KHZ system. The evaluation results are listed in Table 3. From the table, one can see that the parameter β, which controls the range of FPR for the pAUC-L optimization, plays a significant role on the performance. The performance is stable if $\beta \leq 0.1$, and drops significantly when $\beta = 1$, i.e., the AUC-L case. This is because that pAUC-L focuses on discriminating the difficult trials automatically instead of considering all training trials as AUC-L did. It is also observed that the performance with the margin $\delta \geq 0.4$ is much better than that with $\delta = 0$. We also evaluated pAUC-L in the 8KHZ system where the models were trained with 100 epochs using half of the training data. The results are presented in Table 4, which exhibits the similar phenomena as in Table 3.

Comparing Tables 3 and 4, one may see that the optimal values of β in the two evaluation systems are different. This is mainly due to the different difficulty levels of the two evaluation tasks. Specifically, the classification accuracies on the training data of the 16KHZ and 8KHZ systems are 97% and 85% respectively, which indicates that the training trials of the 16KHZ system are much easier to classify than the training trials of the 8KHZ system. Because the main job of β is to select the training trials that are most difficult to be discriminated, setting β in the 16KHZ system to a smaller value than that in the 8KHZ system helps both of the systems reach a balance between the problem of selecting the most difficult trials and gathering enough number of training trials for the DNN training.

5. CONCLUSIONS

This paper presented a method to train deep embedding based text-independent speaker verification with a new verification loss function—pAUC. The major contribution of this paper consists of the following three respects. 1) A pAUC based loss function is proposed for deep embedding. 2) A method is presented to learn the class-centers of the training speakers for the training set construction. 3) we analyzed the connection between pAUC and the representative loss functions. The experimental results demonstrated that the proposed loss function is comparable to the state-of-the-art identification loss functions in speaker verification performance.

6. REFERENCES

[1] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018.

[2] D. Snyder, D. Garcia-Romero, G. Sell, A. McCree, D. Povey, and S. Khudanpur, "Speaker recognition for multi-speaker conversations using x-vectors," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5796–5800.

[3] W. Xie, A. Nagrani, J. S. Chung, and A. Zisserman, "Utterance-level aggregation for speaker recognition in the wild," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5791–5795.

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

[5] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer, "End-to-end text-dependent speaker verification," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 5115–5119.

[6] J.-W. Jung, H.-S. Heo, I.-H. Yang, H.-J. Shim, and H.-J. Yu, "A complete end-to-end speaker verification system using deep neural networks: From raw signals to verification result," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5349–5353.

[7] L. Li, Y. Chen, Y. Shi, Z. Tang, and D. Wang, "Deep speaker feature learning for text-independent speaker verification," *Proc. Interspeech 2017*, pp. 1542–1546, 2017.

[8] Y. Zhu, T. Ko, D. Snyder, B. Mak, and D. Povey, "Self-attentive speaker embeddings for text-independent speaker verification," *Proc. Interspeech 2018*, pp. 3573–3577, 2018.

[9] Y. Tang, G. Ding, J. Huang, X. He, and B. Zhou, "Deep speaker embedding learning with multi-level pooling for text-independent speaker verification," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6116–6120.

[10] W. Cai, J. Chen, and M. Li, "Exploring the encoding layer and loss function in end-to-end speaker and language recognition system," in *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, 2018, pp. 74–81.

[11] G. Bhattacharya, M. J. Alam, and P. Kenny, "Deep speaker embeddings for short-duration speaker verification." in *Interspeech*, 2017, pp. 1517–1521.

[12] Z. Gao, Y. Song, I. McLoughlin, P. Li, Y. Jiang, and L. Dai, "Improving aggregation and loss function for better embedding learning in end-to-end speaker verification system," *Proc. Interspeech 2019*, pp. 361–365, 2019.

[13] S. Wang, Z. Huang, Y. Qian, and K. Yu, "Discriminative neural embedding learning for short-duration text-independent speaker verification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 11, pp. 1686–1696, 2019.

[14] R. Li, N. Li, D. Tuo, M. Yu, D. Su, and D. Yu, "Boundary discriminative large margin cosine loss for text-independent speaker verification," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6321–6325.

[15] C. Zhang, K. Koishida, and J. H. Hansen, "Text-independent speaker verification based on triplet convolutional neural network embeddings," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 26, no. 9, pp. 1633–1644, 2018.

[16] 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)*. IEEE, 2018, pp. 4879–4883.

[17] V. Mingote, A. Miguel, A. Ortega, and E. Lleida, "Optimization of the area under the roc curve using neural network supervectors for text-dependent speaker verification," *arXiv preprint arXiv:1901.11332*, 2019.

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

[19] V. Mingote, A. Miguel, D. Ribas, A. Ortega, and E. Lleida, "Optimization of false acceptance/rejection rates and decision threshold for end-to-end text-dependent speaker verification systems," *Proc. Interspeech 2019*, pp. 2903–2907, 2019.

[20] S. Novoselov, V. Shchemelinin, A. Shulipa, A. Kozlov, and I. Kremnev, "Triplet loss based cosine similarity metric learning for text-independent speaker recognition," *Proc. Interspeech 2018*, pp. 2242–2246, 2018.

[21] C. Li, X. Ma, B. Jiang, X. Li, X. Zhang, X. Liu, Y. Cao, A. Kannan, and Z. Zhu, "Deep speaker: an end-to-end neural speaker embedding system," *arXiv preprint arXiv:1705.02304*, 2017.

[22] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," in *Proc. Interspeech 2018*, 2018, pp. 1086–1090. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2018-1929

[23] Z. Bai, X.-L. Zhang, and J. Chen, "Partial auc metric learning based speaker verification back-end," *arXiv preprint arXiv:1902.00889*, 2019.

[24] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. EPFL-CONF-192584. IEEE Signal Processing Society, 2011.

[25] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: A large-scale speaker identification dataset," *Proc. Interspeech 2017*, pp. 2616–2620, 2017.

[26] M. McLaren, L. Ferrer, D. Castan, and A. Lawson, "The speakers in the wild (sitw) speaker recognition database." in *Interspeech*, 2016, pp. 818–822.