Speaker Verification by Partial AUC Optimization With Mahalanobis Distance Metric Learning

Zhongxin Bai, Xiao-Lei Zhang, and Jingdong Chen

***Abstract*—Receiver operating characteristic (ROC) and detection error tradeoff (DET) curves are two widely used evaluation metrics for speaker verification. They are equivalent since the latter can be obtained by transforming the former's true positive y-axis to false negative y-axis and then re-scaling both axes by a probit operator. Real-world speaker verification systems, however, usually work on part of the ROC curve instead of the entire ROC curve given an application. Therefore, we propose in this article to use the area under part of the ROC curve (pAUC) as a more efficient evaluation metric for speaker verification. A Mahalanobis distance metric learning based back-end is applied to optimize pAUC, where the Mahalanobis distance metric learning guarantees that the optimization objective of the back-end is a convex one so that the global optimum solution is achievable. To improve the performance of the state-of-the-art speaker verification systems by the proposed back-end, we further propose two feature preprocessing techniques based on length-normalization and probabilistic linear discriminant analysis respectively. We evaluate the proposed systems on the major languages of NIST SRE16 and the core tasks of SITW. Experimental results show that the proposed back-end outperforms the state-of-the-art speaker verification back-ends in terms of seven evaluation metrics.**

***Index Terms*—Metric learning, pAUC, speaker verification, squared Mahalanobis distance.**

I. Introduction

SPEAKER verification aims to verify whether an utterance is pronounced by a hypothesized speaker based on some utterances pre-recorded from that speaker. Depending on whether it requires the to-be-verified speaker to pronounce some predefined text or not, speaker verification can be classified into two classes, i.e., *text-dependent* and *text-independent*. This paper focuses on the text-independent case. There are generally two approaches to this problem: a two-step one, which consists of a front-end feature extractor and a back-end classifier, and a one-step approach, which trains an end-to-end system [1]–[4]. This paper focuses on the two-step approach.

In a two-step approach, it is important to have a good front-end. In the literature, the Gaussian mixture model (GMM) based universal background model (UBM) [5] plus identity vector (i-vector) [6] is commonly used. In such a front-end, a GMM-UBM is first trained to collect Baum-Welch statistics, which is formed as a supervector for each utterance. Then, factor analysis is used to reduce the dimensionality of the supervectors to low-dimensional i-vectors. Many extensions of the GMM-UBM/ivector front-end were proposed recently, e.g., [7]. Motivated by the paradigm shift of speech recognition from GMM-based acoustic modeling to deep neural network (DNN) based one, a DNN-UBM/i-vector front-end was developed [8]–[10]. It essentially uses the DNN-based acoustic model trained for speech recognition to generate the posterior probabilities instead of GMM-UBM. Tan *et al.* further employed a denoising autoencoder to replace the DNN-based acoustic model for dealing with environmental noise [11]. These method, however, needs transcriptions of the training data to train the acoustic models, which may not be always available.

An emerging direction of the front-end research is deep embedding. Deep embedding uses a DNN to distinguish the training speakers in a closed set by a classification-based loss function, and takes the outputs of the hidden layers of the DNN for verification. An early deep embedding front-end is the so-called d-vector [12], [13], in which frame-level speaker features are extracted from the top hidden layer, and then utterance-level speaker features are derived as the average of the frame-level features. However, the average of the frame level features does not consider the dependency of the contextual frames. Several efforts have been made to address this problem [14]–[17]. For example, in [14], [15], Snyder *et al.* proposed to insert an average pooling layer into DNN to handle variable-length segments. In [18], Gao *et al.* exploited a cross-convolutional-layer pooling method to extract the first-order statistics of the input segments. Attention mechanism was also studied to generate utterance-level features [16], [17]. Another problem with the deep embedding front-end is on the training loss function. Because the classification-based loss is only a surrogate loss function of the final evaluation metrics of speaker verification, finding more effective loss functions become an important issue. In [19], [20], the authors proposed to minimize the classification-based loss and center loss together. In [21], Zhang *et al.* took triplet loss as the training objective of a deep embedding network. Although employing the above training loss functions is shown to be able

Manuscript received October 13, 2019; revised February 27, 2020 and April 17, 2020; accepted April 21, 2020. Date of publication April 27, 2020; date of current version June 2, 2020. This work was supported in part by the National Key Research and Development Program of China under Grant No. 2018AAA0102200 and in part by National Science Foundation of China (NSFC) and Israel Science Foundation (ISF) Joint Research Program under Grant 61761146001 and by the NSFC Key Program under Grant 61831019, and in part by the NSFC Program under Grant 61671381. *(Corresponding author: Xiao-Lei Zhang.)*

Zhongxin Bai and Xiao-Lei Zhang are with the Center of Intelligent Acoustics and Immersive Communications (CIAIC) and the School of Marine Science and Technology, Northwestern Polytechnical University, Xi'an 710072, China (e-mail: zxbai@mail.nwpu.edu.cn; xiaolei.zhang@nwpu.edu.cn).

Jingdong Chen is with CIAIC, Northwestern Polytechnical University, Xi'an 710072, China (e-mail: jingdongchen@ieee.org).

Digital Object Identifier 10.1109/TASLP.2020.2990275

to improve the performance, the extracted speaker features still have significant intra-class variations, which need to be handled by back-ends.

Regarding the back-end, commonly used back-end classifiers include cosine similarity scoring [6], support vector machine [22], and probabilistic linear discriminant analysis (PLDA) [23]–[25]. DNNs have also been investigated [26], [27]. Inter-session variability compensation is a main task of back-ends, since the front-ends are inter-session- and speaker-dependent. Linear compensation techniques such as linear discriminant analysis (LDA) and within class covariance normalization [28] are often used. Recently, nonlinear compensation methods have been studied as well: Cumani *et al.* [29], [30] proposed a nonlinear transformation to i-vectors to make them more suitable for PLDA [31]; Zheng *et al.* developed a DNN-based dimensionality reduction method as an alternative to LDA [32]. However, because the aforementioned back-ends do not optimize the evaluation metrics directly, such as equal error rate (EER), their performance may be suboptimal.

To optimize the evaluation metrics directly, metric learning needs to be used, which attempts to learn an appropriate similarity measurement space of data points. It has been widely studied in the machine learning community. One of the most popular metric learning methods is to optimize the parameters of a Mahalanobis distance in a linear space [33]. Recently, deep metric learning [34]–[36], which uses a DNN to learn a nonlinear similarity measurement, has also received much attention. Metric learning has been recently studied in speaker verification as well. For example, some metric learning based back-ends [37], [38] have been proposed to compensate the inter-session variability of the embedding features, where the work in [37] minimizes the EER of speaker verification directly. It is also popular to train an end-to-end speaker verification system [1]–[4] or an embedding DNN [21] by deep metric learning. In our recent work [37], we proposed a linear cosine metric learning algorithm to minimize the overlap region of decision scores. Similarly, in [38], Novoselov *et al.* proposed a triplet-loss-based cosine similarity metric learning back-end.

Although directly optimizing an evaluation metric of speaker verification improves the performance, current methods focus mainly on optimizing EER. Since it needs to work at a different point of its receiver operating characteristic (ROC) curve for different applications, a speaker verification system tuned to yield the minimum EER in one scenario may not produce the best performance in another scenario. To address this issue, this paper proposes a back-end to directly optimize part of the area under the ROC curve (named *partial AUC*, or pAUC for short). The main contributions of this paper are summarized as follows:

- A new calibration-insensitive evaluation metric named "pAUC" is proposed for speaker verification. pAUC represents partial area under the ROC curve. It meets the evaluation requirement of real-world applications that work on different parts of ROC curves, such as bank security systems or terrorist detection systems. It is a supplement evaluation metric to the existing metrics. As shown in Fig. 1, the pAUC for a specific application is defined by two false positive rate (FPR) parameters: α and β.
- A Mahalanobis metric learning back-end is proposed to maximize pAUC (pAUCMetric). pAUCMetric evaluates the similarity between two speaker features by a squared Mahalanobis distance, and optimizes the parameters of the distance metric to maximize pAUC where the working points of the speaker verification system locate. pAUCMetric is formulated as a convex optimization problem, where the global optimum solution is guaranteed. We further combine pAUCMetric with two feature preprocessing techniques: 1) length-normalization, and 2) latent variables of PLDA, which combine the ranking property of pAUC into the Cosine similarity or PLDA back-ends for further performance improvement. It is shown that the AUC optimization, such as the one in [39], [40], can be viewed as a special case of pAUC with $\alpha = 0$ and $\beta = 1$.

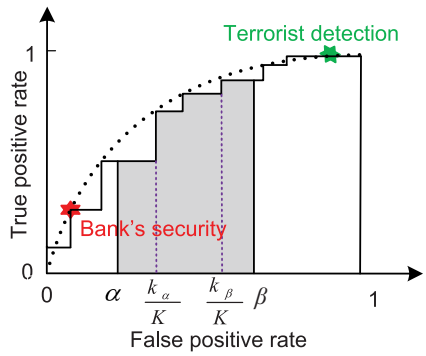

Fig. 1. Illustrations of the ROC curve, AUC, and pAUC.

Experiments are conducted to evaluate the effectiveness of pAUCMetric and compare pAUCMetric with PLDA and cosine similarity scoring back-ends that do not optimize evaluation metrics directly. For each experiment, all back-ends use the same front-end, which is either the GMM/i-vector or the x-vector. We train the comparison methods on switchboard, NIST SRE04–SRE10 and VoxCeleb datasets, and evaluate them on the major languages of NIST SRE16 and the core tasks of SITW. The evaluation is conducted under the conditions of both noise-matching and -mismatching, as well as both language-matching and -mismatching. The experimental results show that pAUCMetric outperforms PLDA by relatively 10%, 9% and 20% in terms of EER, pAUC and AUC metrics respectively.

The rest of this paper is organized as follows: Section II presents the motivations. Section III and V describe the proposed algorithm. Section VI presents the experiment results. Finally, important conclusions are drawn in Section VII.

II. Motivation

A. Motivation for the pAUC Evaluation Metric

It is known that a speaker verification system first generates a similarity score of a trial by a speaker detection algorithm, and then makes a hard decision according to a threshold as illustrated in Fig. 2. The speaker detection algorithm assigns

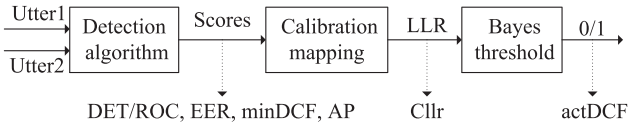

Fig. 2. Diagram of a speaker verification system with common evaluation metrics.

higher scores to *target trials* than *non-target trials*, which determines the *discriminability* of the system. The decision threshold is usually determined by first *calibrating* the similarity scores to the log-likelihood ratios (LLR) and then applying the Bayes decision theory [41] using application-dependent priors, i.e., the prior of targets and the costs of false negative rate (also known as miss detection rate) and false positive rate (also known as false alarm rate).

The evaluation metrics of speaker verification in Fig. 2 can be categorized to two classes—*calibration-sensitive* metrics and *calibration-insensitive* ones. The calibration-sensitive metrics, which include the actual detection cost function (actDCF) and cost of LLR (C_{llr}), aim to evaluate a calibrated speaker verification system under the framework of Bayes decision theory. Specifically, the application-dependent actDCF evaluates the empirical Bayes risk of a system at the Bayes decision threshold [41], which determines how good is the hard decision. C_{llr} evaluates the discrimination of the calibrated LLR in an application-independent manner [42]. While calibration-sensitive evaluation metrics have many pros in evaluating the suitability of a calibrated system, we often need to evaluate the detection algorithm of an uncalibrated system directly.

In contrast, calibration-insensitive metrics evaluate the discriminability of the detection algorithm. They include the detection error tradeoff (DET) curve, EER, minimum detection cost function (minDCF), and average precision. DET curve is an alternative form of the ROC curve. As a matter of fact, the DET curve can be obtained by transforming the ROC curve's true positive y-axis to false negative y-axis and then re-scaling both axes by a non-linear warping named the probit operator [41], [43]. It reflects the global discriminability of a speaker verification system. EER and minDCF are two points on the DET curve, which reflect the discriminability of the system to some extent. Like the DET curve, average precision is a global metric that combines recall and precision for ranked retrieval results, which is however sensitive to class-imbalanced problems such as speaker verification. To summarize, DET curve and average precision are two global metrics, while EER and minDCF are two local points on the DET curve.

In practice, a speaker verification system usually works on a local fraction of the DET curve with a tunable threshold, instead of a single local point. For example, a bank security system is tuned in a range where the false positive rate is controlled below 0.01%. In contrast, a terrorist detection system of a public security department is tuned in a range whose recall rate is required in a range of higher than 99%. As shown in Fig. 1, pAUC may meet such a requirement. First, $[\alpha, \beta]$ in Fig. 1 defines the interested operating points of a real-world working scenario. Second, pAUC, which is a scalar in the range of [0,1], describes the interested part of the ROC curve efficiently. At last, its calculation method, which will be presented in (7), does not depend on a decision threshold. Hence, we adopt pAUC as a new calibration-insensitive evaluation metric.

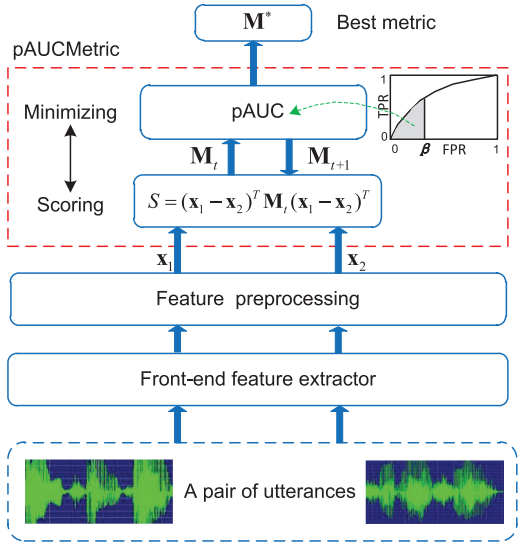

Fig. 3. Diagram of the pAUCMetric based speaker verification system.

B. Motivation for the pAUCMetric Back-End

How to optimize calibration-sensitive evaluation metrics has been well studied and a number of methods were developed [44], [45]. But those methods do not improve the discriminability of the detection algorithm as the order of the similarity scores of training trials is not changed. In order to improve the discriminability of the detection algorithm, it is better to optimize the ROC curve directly by maximizing its AUC. However, optimizing the entire AUC is not only costly but also unnecessary as that most practical systems work only on part of their ROC curves. Therefore, we propose a metric learning back-end based on Mahalanobis distance to optimize pAUC accordingly.

Another advantage of pAUCMetric is that it can select difficult negative training trials by setting β to a small value, which is a well-known challenging problem for the algorithms that need to group training utterances into training trials. As will be shown in the experiments, the proposed pAUCMetric performs better than a triplet-loss-based algorithm, which differs from pAUCMetric only in the loss function, for all the aforementioned evaluation metrics.

III. pAUC Metric Learning Back-End

In this section, we first provide an overview to the speaker verification system in Section III-A, and then present the objective function and optimization algorithm of the proposed back-end in Sections III-B and III-C respectively.

A. System Overview

The diagram of the pAUCMetric based speaker verification system is shown in Fig. 3. The front-end is used to extract speaker features from speech signals. We use i-vector [6] or x-vector [15] as the front-end. After feature extraction by the front-end, we

further preprocess the features as described in Section V, and then use the preprocessed feature as the input of pAUCMetric.

The role of pAUCMetric is to judge whether two preprocessed features $\mathbf{x}_{q_1}$ and $\mathbf{x}_{q_2}$ belong to the same speaker based on their similarity. The similarity is measured by the following squared Mahalanobis distance:

$$S(\mathbf{x}_{q_1}, \mathbf{x}_{q_2}; \mathbf{M}) = (\mathbf{x}_{q_1} - \mathbf{x}_{q_2})^T \mathbf{M} (\mathbf{x}_{q_1} - \mathbf{x}_{q_2}) \tag{1}$$

where $\mathbf{M}$ is a symmetric positive semi-definite matrix, which is to be learned by pAUCMetric. If the squared Mahalanobis distance between $\mathbf{x}_{q_1}$ and $\mathbf{x}_{q_2}$ is smaller than a pre-specified threshold θ^*, $\mathbf{x}_{q_1}$ and $\mathbf{x}_{q_2}$ are regarded as from the same speaker; otherwise, they are regarded as from different speakers. We denote $\mathbf{z}_i = \mathbf{x}_{q_1} - \mathbf{x}_{q_2}$, and denote $S(\mathbf{x}_{q_1}, \mathbf{x}_{q_2}; \mathbf{M})$ as $S(\mathbf{z}_i; \mathbf{M})$ for simplicity. A probabilistic explanation of the Mahalanobis distance is given in Appendix A.

B. Objective Function

Given a training set with N speakers and Q embedding vectors $\mathcal{X} = \{(\mathbf{x}_q, y_q)\}_{q=1}^{Q}$, where $y_q = 1, \ldots, N$ is the identity of $\mathbf{x}_q$, we first construct a pairwise training set

$$\mathcal{T} = \{(\mathbf{z}_i, l_i)\}_{i=1}^{I} \tag{2}$$

where $\mathbf{z}_i = \mathbf{x}_{q_1} - \mathbf{x}_{q_2}$ with $q_1 = 1, \ldots, Q$ and $q_2 = 1, \ldots, Q$ ($q_1 \neq q_2$), I is the size of $\mathcal{T}$, and l_i is the ground-truth label of $\mathbf{z}_i$ satisfying:

$$l_i = \begin{cases} 1, & \text{if } y_{q_1} = y_{q_2} \\ -1, & \text{otherwise} \end{cases} \tag{3}$$

We define the subset of the true trials of $\mathcal{T}$ as:

$$\mathcal{P} = \{(\mathbf{z}_j^+, l_j = 1)\}_{j=1}^{J} \tag{4}$$

and the subset of the imposter trials of $\mathcal{T}$ as:

$$\mathcal{N} = \{(\mathbf{z}_k^-, l_k = -1)\}_{k=1}^{K} \tag{5}$$

where J and K are the sizes of $\mathcal{P}$ and $\mathcal{N}$ respectively.

After the above preliminary setting, the pAUC is calculated as follows. We define a subset of $\mathcal{N}$ that defines the pAUC over the FPR range $[\alpha, \beta]$:

$$\mathcal{N}_0 = \{(\mathbf{z}_r^-, l_r = -1)\}_{r=1}^{R} \tag{6}$$

where $R \leq K$, and $\mathcal{N}_0$ is determined as following. Because the imposter set $\mathcal{N}$ contains only a limited number of trials, we first replace $[\alpha, \beta]$ by $[k_\alpha/K, k_\beta/K]$ where $k_\alpha = \lceil K\alpha \rceil$ and $k_\beta = \lfloor K\beta \rfloor$ are two integers. Then, $\{S(\mathbf{z}_k^-; \mathbf{M})\}_{\mathbf{z}_k^- \in \mathcal{N}}$ are sorted in ascending order. Finally, we pick the trials ranked from the top k_αth to k_βth positions to form $\mathcal{N}_0$. The calculation of pAUC is equivalent to that of the normalized AUC over $\mathcal{P}$ and $\mathcal{N}_0$, which is computed as:

$$\text{pAUC} = 1 - \frac{1}{JR} \sum_{j=1}^{J} \sum_{r=1}^{R} \left[\mathbb{I}(S(\mathbf{z}_j^+; \mathbf{M}) > S(\mathbf{z}_r^-; \mathbf{M})) + \frac{1}{2} \mathbb{I}(S(\mathbf{z}_j^+; \mathbf{M}) = S(\mathbf{z}_r^-; \mathbf{M})) \right] \tag{7}$$

where $\mathbb{I}(\cdot)$ is an indicator function that returns 1 if the statement is true, and 0 otherwise.

However, directly optimizing (7) is an NP-hard problem. To circumvent this, let us relax (7) by replacing the indicator function by a hinge loss function:

$$\ell_{\text{hinge}}(S(\mathbf{z}_j^+; \mathbf{M}) > S(\mathbf{z}_r^-; \mathbf{M})) = \max\left[0, \delta - \left(S(\mathbf{z}_r^-; \mathbf{M}) - S(\mathbf{z}_j^+; \mathbf{M})\right)\right] \tag{8}$$

where $\delta > 0$ is a tunable hyper-parameter controlling the distance margin between $\{S(\mathbf{z}_r^-; \mathbf{M})\}_{\mathbf{z}_r^- \in \mathcal{N}_0}$ and $\{S(\mathbf{z}_j^+; \mathbf{M})\}_{\mathbf{z}_j^+ \in \mathcal{P}}$. Substituting (8) into (7) and further changing the maximization problem (7) into an equivalent minimization one gives (9).

$$\ell = \frac{1}{JR} \sum_{j=1}^{J} \sum_{r=1}^{R} \max\left(0, \delta - S(\mathbf{z}_r^-; \mathbf{M}) + S(\mathbf{z}_j^+; \mathbf{M})\right) \tag{9}$$

The proposed pAUCMetric minimizes (9) over $\mathcal{P}$ and $\mathcal{N}_0$. To prevent overfitting to the training data, we add a regularization term $\lambda\Omega(\cdot)$ to the minimization problem according to a plausible formulation in [46], which gives the objective function of pAUCMetric:

$$\mathbf{M}^* = \arg\min_{\mathbf{M}} \ell(\mathcal{P}, \mathcal{N}; \mathbf{M}) + \lambda\Omega(\mathbf{M}), \tag{10}$$

where λ is a regularization hyperparameter, and $\lambda\Omega(\cdot)$ is defined as:

$$\lambda\Omega(\mathbf{M}) = \frac{\gamma}{J} \sum_{j=1}^{J} S(\mathbf{z}_j^+; \mathbf{M}) + \mu[\text{tr}(\mathbf{M}) - \text{logdet}(\mathbf{M})] \tag{11}$$

with γ and μ being two tunable hyper-parameters. The first term on the right-hand side of (11), i.e., $\frac{1}{J}\sum_{j=1}^{J} S(\mathbf{z}_j^+; \mathbf{M})$, which was first introduced in [47], aims to bound $S(\mathbf{z}_j^+; \mathbf{M})$ in (9). The second term, i.e., $\text{tr}(\mathbf{M}) - \text{logdet}(\mathbf{M})$, which is a specifical case of *LogDet divergence* [48] defined over positive semi-definite (PSD) matrices [33], is used to improve the generalization ability and further constrain $\mathbf{M}$ to be PSD.

(10) can also be interpreted from another viewpoint using the following lemma.

Lemma 1: The maximization of pAUC in (10) is a problem of enlarging a weighted margin between the positive and negative trials while minimizing the within-class variances of the two class trials simultaneously.

Proof: Let us define an index matrix $\mathbf{\Pi} \in \{0, 1\}^{J \times R}$:

$$\mathbf{\Pi}(j, r) = \begin{cases} 1, & \text{if } \delta + S(\mathbf{z}_j^+; \mathbf{M}) > S(\mathbf{z}_r^-; \mathbf{M}) \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

and rewrite the loss function of (9) as:

$$\begin{aligned} \ell &= \frac{\delta}{JR} \sum_{j=1}^{J} \sum_{r=1}^{R} \mathbf{\Pi}(j, r) + \frac{1}{J} \sum_{j=1}^{J} \left(\frac{1}{R} \sum_{r=1}^{R} \mathbf{\Pi}(j, r) \right) S(\mathbf{z}_j^+; \mathbf{M}) \\ &\quad - \frac{1}{R} \sum_{r=1}^{R} \left(\frac{1}{J} \sum_{j=1}^{J} \mathbf{\Pi}(j, r) \right) S(\mathbf{z}_r^-; \mathbf{M}) \\ &= c + \frac{1}{J} \sum_{j=1}^{J} p_j S(\mathbf{z}_j^+; \mathbf{M}) - \frac{1}{R} \sum_{r=1}^{R} p_r S(\mathbf{z}_r^-; \mathbf{M}) \end{aligned} \tag{13}$$

Algorithm 1: Mini-batch PPA [46] Algorithm for pAUC-Metric Optimization.

Require:
Development set: $\mathcal{X}$;
False positive rate: $\alpha \geq 0, \beta > 0$;
Hyperparameter: $\delta \geq 0$, $\gamma \geq 0$, $\mu \geq 0$;
Batch size: s;
Step size parameter: $\eta > 0$;
Initialize: $t \leftarrow 0$, $\mathbf{M}_0 = \mathbf{I}_0$, where $\mathbf{I}_0$ is the identity matrix;

repeat
  Construct a mini-batch subset of $\mathcal{X}$ by random sampling;
  Construct $\mathcal{T}$ from the subset of $\mathcal{X}$ by (2)
  Compute $\mathcal{P}$ and $\mathcal{N}_0$ by (4) and (6);
  Calculate $\mathbf{P}^t$ and $\mathbf{P}^t_{\mathcal{P}}$ on $\mathcal{P}$ and $\mathcal{N}_0$ by (15) and (16);
  $\mathbf{M}_{t+1} \leftarrow \phi^+_\lambda(\mathbf{M}_t - \eta(\mathbf{P}^t + \gamma\mathbf{P}^t_{\mathcal{P}} + \mu\mathbf{I}_0))$, where $\lambda = \eta\mu$;
  $t \leftarrow t + 1$.
until *converged*;
Output : $\mathbf{M}_t$

where $c = \frac{\delta}{JR}\sum_{j=1}^{J}\sum_{r=1}^{R}\mathbf{\Pi}(j,r)$ is a constant in a single iteration, $p_j = \frac{1}{R}\sum_{r=1}^{R}\mathbf{\Pi}(j,r)$ and $p_r = \frac{1}{J}\sum_{j=1}^{J}\mathbf{\Pi}(j,r)$ are the weights of the positive and negative trials respectively. It is clear that minimizing (13) is a problem of enlarging the weighted margin between the positive and negative trials. ■

Because the regularization term $\frac{\gamma}{J}\sum_{j=1}^{J}S(\mathbf{z}_j^+;\mathbf{M})$ minimizes the within-class variance, we see that the objective (10) enlarges the between-class distance and minimizes the within-class variance simultaneously, which is also the principle behind many well-known back-ends, such as LDA, WCCN, and PLDA. The difference lies in that pAUCMetric works in the squared Mahalanobis distance space and encodes the pAUC information into the weights p_j and p_r.

C. Optimization Algorithm

In order to solve the optimization problem in (6), substituting (12) into (10) gives

$$\mathbf{M}^* = \arg\min_{\mathbf{M}} \langle \mathbf{P} + \gamma\mathbf{P}_{\mathcal{P}}, \mathbf{M}\rangle_F + \mu\left[\mathrm{tr}(\mathbf{M}) - \mathrm{logdet}(\mathbf{M})\right], \tag{14}$$

where $\langle\cdot\rangle_F$ denotes the Frobenius norm operator, and

$$\mathbf{P}_{\mathcal{P}} = \frac{1}{J}\sum_{j=1}^{J}\mathbf{z}_j^+\mathbf{z}_j^{+T}, \tag{15}$$

$$\mathbf{P} = \frac{1}{JR}\sum_{j=1}^{J}\sum_{r=1}^{R}\mathbf{\Pi}(j,r)(\mathbf{z}_j^+\mathbf{z}_j^{+T} - \mathbf{z}_r^-\mathbf{z}_r^{-T}). \tag{16}$$

We employ the proximal point algorithm (PPA) [46] to optimize (14). The resulting algorithm, which is summarized in Algorithm 1, consists of the following three steps at each iteration:

- The first step constructs the training set $\mathcal{T}$ from $\mathcal{X}$. However, if we consider all trials in $\mathcal{X}$ during the construction of $\mathcal{T}$, the size of $\mathcal{T}$ becomes enormous. To prevent the overload of computing, we construct a pairwise set $\mathcal{T}^t$ at each iteration by a random sampling strategy as follows. We first randomly select s speakers from $\mathcal{X}$, then randomly select two embedding vectors from each of the selected speakers, and finally construct $\mathcal{T}^t$ by a full permutation of the $2s$ embedding vectors. $\mathcal{T}^t$ contains s true training trials and $s(2s-1)-s$ imposter training trials.
- The second step calculates $\mathcal{N}_0^t$ according to (6), and calculates $\mathbf{P}^t$ and $\mathbf{P}^t_{\mathcal{P}}$ according to (15) and (16) respectively.
- The third step updates $\mathbf{M}$ by PPA [46], which first applies eigenvalue decomposition to $\mathbf{X} = \mathbf{M}_t - \eta(\mathbf{P}^t + \gamma\mathbf{P}^t_{\mathcal{P}} + \mu\mathbf{I}_0)$, i.e., $\mathbf{X} = \mathbf{U}\mathbf{V}\mathbf{U}^T$ where $\mathbf{V} = \mathrm{diag}([v_1, v_2, \ldots, v_d])$ with $v_1 \geq v_2 \geq \cdots \geq v_d$, and then adopts the following updating equation:

$$\phi^+_\lambda(\mathbf{x}) = \mathbf{U}\mathrm{diag}([\phi^+_\lambda(v_1), \ldots, \phi^+_\lambda(v_d)])\mathbf{U}^T, \tag{17}$$

where $\phi^+_\lambda(v) = [(v^2 + 4\lambda)^{1/2} + v]/2$, and d is the dimension of the input feature.

D. Connection to the Back-Ends Trained With Training Trials

There are two basic classes of back-ends depending on how they construct the training data. One class takes training utterances as the training data for training a generative PLDA. The other groups training utterances into training trials for training binary-class classifiers, in which back-ends differ in two aspects—basic classifiers and loss functions. Here we focus on discussing the difference between the loss functions of the second class, which include the pairwise SVM [22], [49], [50], triplet-loss-based, and pAUCMetric back-ends whose loss functions are denoted as the classification-loss, triplet-loss, and pAUC-loss (9), respectively.

The classification-loss [22], triplet-loss [21], and pAUC-loss all use the hinge loss function to relax the $0/1$-loss. The only difference between them is how the errors are accumulated. The classification-loss accumulates the classification error, which suffers from the class-imbalance problem of speaker verification. In contrast, the pAUC-loss focuses on the ranking of the similarity scores; So, it does not suffer from the class-imbalance problem.

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

$$S(\mathbf{x}^a, \mathbf{x}^n; \mathbf{M}) - S(\mathbf{x}^a, \mathbf{x}^p; \mathbf{M}) > \delta \tag{18}$$

where $\mathbf{x}^a$, $\mathbf{x}^p$, and $\mathbf{x}^n$ represent, respectively, the anchor, positive, and negative utterances of a trial. For clarity, we denote the speaker features in a constraint as a *relative constraint*. For example, we call $\{\mathbf{x}^a, \mathbf{x}^p, \mathbf{x}^n\}$ in (18) as a relative constraint of the triplet-loss. The difference between the triplet-loss (18) and pAUC-loss (9) lies in the following three aspects.

First, the relative constraints of the triplet-loss (18) are triplet, which cannot deal with the situation where the training data contains only positive or negative trials. While the relative

constraints of the pAUC-loss (9) are tetrad, which matches the pipeline of speaker verification. Therefore, (9) does not have the same limitation as (18). Second, the pAUC-loss is intrinsically able to pick difficult training trials from the exponentially large number of training trials, while the triplet-loss lacks such an ability (that is why it has to use additional training trial selection methods). Third, as proven in Appendix B, the relative constraints of the triplet-loss are a subset of the relative constraints of the AUC-loss. As a result, the triplet-loss is a specifical case of the AUC-loss. Moreover, because the AUC-loss is a specifical case of the pAUC-loss with $\alpha = 0$ and $\beta = 1$, we conclude that the triplet-loss is a specifical case of the pAUC-loss.

To validate the above analysis, we propose a new triplet-loss based algorithm for experimental comparison, which differs from pAUCMetric only in the loss function. The new algorithm, named TripletMetric, replaces the tetrad constraints (9) with the triplet constraints (18). Its training data are constructed in a similar way as $\mathcal{T}$ in Section III-C, which randomly selects s speakers with each speaker selecting two embedding vectors. The number of the training triplet trials is $2s(2s-2)$. Note that, because the number of the tetrad constraints in (9) is $s[(s(2s-1)-s)(\beta-\alpha)]$, the ratio of the number of the training trials of pAUCMetric to that of TripletMetric is $\frac{2}{s(\beta-\alpha)}$.

IV. Complexity Analysis

Theorem 1: The computational complexity of pAUCMetric is:

$$O = \mathcal{O}(d^2(I+J+R)) + \mathcal{O}(JR) + \mathcal{O}(d^3) + \mathcal{O}(Klog_2K) \tag{19}$$

where I, J, R, and K are the size of $\mathcal{T}$, $\mathcal{P}$, $\mathcal{N}_0$, and $\mathcal{N}$, respectively, and d is the dimension of the input feature.

Proof: According to Algorithm 1, the computational complexity of pAUCMetric is composed of three parts:

The first part is the computation of $\mathcal{P}$ and $\mathcal{N}_0$. We first need $\mathcal{O}(I)$ operations to separate the positive and negative trials in $\mathcal{T}$. Then, computing the squared Mahalanobis distances between all training pairs according to (1) consumes $\mathcal{O}(d^2I)$ multiplications. Finally, we need $\mathcal{O}(Klog_2K)$ operations to sort all scores of $\mathcal{N}$ for $\mathcal{N}_0$. Thus, the total computational complexity of the first part is:

$$O_1 = \mathcal{O}(I) + \mathcal{O}(d^2I) + \mathcal{O}(Klog_2K). \tag{20}$$

The second part is the computation of $\mathbf{P}^t$ and $\mathbf{P}^t_{\mathcal{P}}$. First, computing $\mathbf{\Pi}(j,r)$ according to (12) needs $\mathcal{O}(JR)$ operations. Then, computation of $\mathbf{P}^t_{\mathcal{P}}$ and $\mathbf{P}^t$ needs $\mathcal{O}(d^2J)$ and $\mathcal{O}(d^2J + d^2R)$ multiplications respectively. The total computational complexity of the second part is therefore:

$$O_2 = \mathcal{O}(JR) + \mathcal{O}(d^2J) + \mathcal{O}(d^2J + d^2R). \tag{21}$$

The third part is the computation of $\mathbf{M}_t$. Both of the eigenvalue decomposition and the updating procedure consume $\mathcal{O}(d^3)$ multiplications. Therefore, the third part has a complexity of:

$$O_3 = \mathcal{O}(d^3). \tag{22}$$

Summing the above three parts gives (19), which completes the proof. ■

Because the value of d is relatively small, the overall computational complexity depends mainly on the complexity of computing the $\mathbf{\Pi}(j,r)$ matrix, which is quadratic with respect to $\mathcal{P}$, $\mathcal{N}_0$. The complexity of computing $\mathbf{\Pi}(j,r)$ is reduced by the random sampling strategy described in Section III-C, which leads the following corollary.

Corollary 1: Given the batch size s, the computational complexity of pAUCMetric is reduced to

$$\mathcal{O}(2cs^3) \tag{23}$$

where c is a coefficient related to the FPR range $[\alpha, \beta]$.

Proof: According to Section III-C, we have $I = 2s^2 - s$, $J = s$, $K = 2s^2 - 2s$, and $R = c(2s^2 - 2s)$. Therefore, the computational complexity is reduced to $\mathcal{O}(2cs^3) + \mathcal{O}(d^3)$. Because the dimension d is small, the computational complexity depends mainly on s only. ■

Corollary 1 shows that the computational complexity of pAUCMetric is cubic with respect to s. As will be shown in Section VI-F3, pAUCMetric can achieve good performance with a small value of s.

V. The Input Features of pAUCMetric

After the feature extraction by a front-end, one needs to preprocess the features for boosting the performance of pAUCMetric as shown in Fig. 3. This section presents two preprocessing techniques.

A. Length-Normalization

Given a speaker feature $\mathbf{y}$ from a front-end, we use the length-normalized feature [25] $\mathbf{x}$ as the input to pAUCMetric:

$$\mathbf{x} = \frac{\mathbf{y}}{\|\mathbf{y}\|_2} \tag{24}$$

The underlying reason for this normalization is as follows. Learning a transform matrix in the cosine similarity scoring framework, i.e.,

$$S_{\cos}(\mathbf{y}_1, \mathbf{y}_2; \mathbf{M}) = \frac{\langle \mathbf{A}\mathbf{y}_1, \mathbf{A}\mathbf{y}_2 \rangle}{\|\mathbf{A}\mathbf{y}_1\|_2 \|\mathbf{A}\mathbf{y}_2\|_2} \tag{25}$$

has been studied extensively, e.g. [6]. However, the learning problem is nonlinear and non-convex. Existing methods either learn $\mathbf{A}$ independently by, e.g., LDA, WCCN [6], or learn $\mathbf{A}$ in the above framework with a good initialization [37]. Both ways are suboptimal.

The Euclidean distance scoring is empirically inferior to the cosine similarity scoring when given the same input $\mathbf{y}$. But it is equivalent to the cosine similarity scoring if its input is the length-normalized feature $\mathbf{x}$ since

$$\begin{aligned} S_{\text{Euc}}(\mathbf{x}_1, \mathbf{x}_2) &= \|\mathbf{x}_1\|_2^2 + \|\mathbf{x}_2\|_2^2 - 2\langle \mathbf{x}_1, \mathbf{x}_2 \rangle \\ &= 2 - 2S_{\cos}(\mathbf{y}_1, \mathbf{y}_2). \end{aligned} \tag{26}$$

More importantly, learning $\mathbf{A}$ in the following Euclidean distance scoring framework does not suffer from the nonlinear and

non-convex issues:

$$\begin{aligned} S_{\mathrm{Euc}}(\mathbf{x}_1, \mathbf{x}_2; \mathbf{A}) &= \|\mathbf{A}\mathbf{x}_1 - \mathbf{A}\mathbf{x}_2\|_2^2 \\ &= (\mathbf{x}_1 - \mathbf{x}_2)^T \mathbf{M} (\mathbf{x}_1 - \mathbf{x}_2) \\ &= S(\mathbf{x}_1, \mathbf{x}_2; \mathbf{M}) \end{aligned} \tag{27}$$

where $\mathbf{M} = \mathbf{A}^T\mathbf{A}$ and $S(\cdot)$ is the scoring function of our pAUCMetric.

B. PLDA-Based Preprocessing

Two kinds of PLDA algorithms have been widely adopted in speaker verification, i.e., the simplified PLDA [23], [25] and the two-covariance based PLDA [51]. We adopt the latent variables of the simplified PLDA [23] as the input features of pAUCMetric. It generates a centralized feature $\mathbf{x}$ by first generating a speaker center $\mathbf{h}$ according to:

$$\mathbf{h} \sim \mathcal{N}(0, \mathbf{\Phi}_b) \tag{28}$$

and then generating the observation data according to:

$$\mathbf{x} \sim \mathcal{N}(\mathbf{h}, \mathbf{\Phi}_w), \tag{29}$$

where $\mathbf{\Phi}_b$ is required to be positive semi-definite, and $\mathbf{\Phi}_w$ is required to be positive definite. The expectation maximization algorithm is employed to estimate the parameters. $\mathbf{\Phi}_b$ and $\mathbf{\Phi}_w$ can be simultaneously diagonalized by solving the following generalized eigenvalue problem:

$$\mathbf{\Phi}_b \mathbf{w} = \psi \mathbf{\Phi}_w \mathbf{w}, \tag{30}$$

which leads to

$$\mathbf{W}\mathbf{\Phi}_b\mathbf{W}^T = \mathbf{\Psi} \tag{31}$$

$$\mathbf{W}\mathbf{\Phi}_w\mathbf{W}^T = \mathbf{I}_0 \tag{32}$$

where $\mathbf{W}$ is a square matrix whose columns are the generalized eigenvectors of (30), $\mathbf{\Psi}$ is a diagonal matrix whose diagonal elements are the generalized eigenvalues of (30), and $\mathbf{I}_0$ is the identity matrix.

Finally, the centralized feature is calculated as:

$$\mathbf{x} = \mathbf{W}^{-1}\mathbf{u}, \tag{33}$$

where $\mathbf{u} \sim \mathcal{N}(\mathbf{v}, \mathbf{I}_0)$, and $\mathbf{v} \sim \mathcal{N}(\mathbf{v}, \mathbf{\Psi})$, $\mathbf{v}$ represents the speaker, and $\mathbf{u}$ represents an example of that speaker in the latent space. Therefore, the example $\mathbf{x}$ in the original space is related to its latent representation $\mathbf{u}$ via an invertible transformation $\mathbf{W}$. We take the latent variable $\mathbf{u}$ as input features of pAUCMetric.[1] This preprocessing method adopts the advantage of the PLDA adaptation into the input features, which improves the overall performance of the speaker verification system.

VI. Experiments

In this section, we first present the datasets and experimental settings and then the main results as well as analysis on the effects of the hyperparameters of pAUCMetric.

[1]Similar to the implementation of the PLDA in Kaldi, we normalize $\mathbf{u}$ to $\mathbf{u} \times \sqrt{\frac{d}{\mathbf{u}^T(\mathbf{\Psi}+\mathbf{I}_0)^{-1}\mathbf{u}}}$, where d is the dimension of $\mathbf{u}$.

A. Datasets

1) Training Datasets: The training data consists of Switchboard (SWBD), NIST speaker recognition evaluation (SRE), and VoxCeleb database. SWBD consists of Switchboard Cellular 1 and 2 as well as Switchboard 2 Phase 1, 2, and 3. It contains 28,181 English utterances from 2,594 speakers. The SRE database consists of NIST SREs from 2004 to 2010 along with Mixer 6. It contains 64,388 telephone and microphone recordings from 4,392 speakers. Most of the utterances are in English, while some utterances are in Chinese, Russian, Arabic etc. VoxCeleb consists of VoxCeleb1 [52] and VoxCeleb2 [53], which contains over 1 million recordings from 7,363 celebrities. It is collected from real world noisy environments, therefore it contains background chatter, laughter and overlapping speech etc. In addition, we adopt the same data augmentation scheme as in [15] to further increase the amount and diversity of the training data. See Table II for summarization of the training data.

2) Evaluation Datasets: The evaluation data include NIST SRE 2016 (SRE16) [54] and the Speakers in the Wild (SITW) [55] datasets. Specifically, SRE16 contains two major languages—Cantonese and Tagalog. They are recorded in real-world noisy environments. The Cantonese language contains 965,393 trials. The Tagalog language contains 1,021,332 trials. The enrollment segments vary from 60 to 180 seconds, and the test utterances are about 10 to 180 seconds long. The SITW is collected from open-source media, which contains real-world noise, reverberation, intra-speaker variability and compression artifacts. It contains 299 speakers. Each recording varies from 6 to 180 seconds. It has two evaluation tasks—Dev.Core which consists of 338,226 trials, and Eval.Core, which consists of 721,788 trials. See Table III for summarization of the evaluation data.

B. Experimental Settings

1) Training Schemes: Due to different collection methods and sampling rates of the training data, we define two kinds of systems:

- **8 kHZ system:** We adopt the augmented SWBD and SRE data, which include 220,569 recordings in total, to train front-end feature extractors. The back-ends are trained on the augmented SRE data. The signals originally sampled at 16 kHz are downsampled to 8 kHz.
- **16 kHZ system:** We use the VoxCeleb data to train an i-vector feature extractor, and use the augmented VoxCeleb data to train an x-vector feature extractor. We randomly selected 200,000 recordings from the augmented VoxCeleb data to train the back-ends.

2) Front-ends: We use the GMM-UBM/i-vector and x-vector front-ends to extract speaker features. The front-ends are implemented using Kaldi [56]. Their parameter settings are also the same as in Kaldi, which are summarized in Table I.

Specifically, for the i-vector extractor, the frame length is 25 ms, and the frame shift is 10 ms. The frame-level acoustic features of the 8 kHZ and 16 kHZ systems are 20- and 24-dimensional MFCCs respectively, which are further

TABLE I
PARAMETER SETTINGS OF FRONT-ENDS. THE TERMS "DIM" AND "MIX" IS SHORT FOR *DIMENSIONS* AND *MIXTURES* RESPECTIVELY. THE TERMS Δ AND $\Delta\Delta$ DENOTE THE DELTA AND DELTA-DELTA COEFFICIENTS OF MFCCS RESPECTIVELY

Systems	i-vector	x-vector
8 kHZ system	20-dim MFCCs +Δ + $\Delta\Delta$ / 2048-mix GMM / 600-dim i-vector	23-dim MFCCs / 512-dim x-vector
16 kHZ system	24-dim MFCCs +Δ + $\Delta\Delta$ / 2048-mix GMM / 400-dim i-vector	30-dim MFCCs / 512-dim x-vector

TABLE II
DESCRIPTIONS OF TRAINING DATASETS

	SWBD	SRE	VoxCeleb
Languages	English	English (most) , others	Multilingual
#Speakers	2,594	4,392	7,363
#Recordings	28,181	64,388	1,281,762
Data sources	Telephone	Telephone, microphone	Multi-media
Environments	Clean	Clean	Real world noise

TABLE III
DESCRIPTIONS OF EVALUATION DATASETS

	SRE16	SITW
Enrollment durations	60 $\sim$ 180 secs	6 $\sim$ 180 secs
Test durations	10 $\sim$ 60 secs	6 $\sim$ 180 secs
Data sources	Telephone	Multi-media
Evaluation kinds	Cantonese / Tagalog	Dev.Core / Eval.Core
#Evaluation trials	965,396 / 1,021,332	338,226 / 721,788

mean-normalized over a sliding window of 3 s. The final acoustic features are a concatenation of the MFCCs and their delta and delta-delta coefficients, which produces a total of 60-dimensional acoustic feature vector for the 8 kHZ system and 72-dimensional acoustic feature vector for the 16 kHZ system. An energy-based voice activity detector (VAD) is employed to remove non-speech frames. The number of Gaussian mixtures is set to 2048 for both the 8 kHZ and 16 kHZ systems. The dimension of the i-vectors is set to 600 for the 8 kHZ system, and 400 for the 16 kHZ system.

For the x-vector extractor, we used the standard Kaldi SRE16 and SITW recipes. Specifically, the frame-length is 25 ms, and the frame shift is 10 ms. The acoustic features of the 8 kHZ and 16 kHZ systems are 24- and 30-dimensional MFCCs, respectively, which are further mean-normalized over a sliding window of 3 s. The energy-based VAD is the same as that in the i-vector extractor. The 8 kHZ x-vector extractor is a pre-trained system provided at *http://kaldi-asr.org/models/m3*. The 16 kHZ x-vector extractor is a newly trained system by Kaldi. The dimensions of the x-vectors in both the systems are set to 512.

3) Back-ends: We compare pAUCMetric with the state-of-art PLDA back-end and a commonly used cosine similarity scoring back-end. The parameter settings of the compared back-ends are summarized as following.

- **PLDA:** We first reduce the speaker features into a low dimensional vector by linear discriminant analysis (LDA). Specifically, if the i-vector front-end is used, the LDA dimension is set to 200 for the 8 kHZ system and 150 for the 16 kHZ system. If the x-vector front-end is used, the LDA dimension is set to 150 and 128 in the 8 kHZ system and 16 kHZ system, respectively. The dimensions of LDA are summarized in Table IV. We use the output of LDA as the input of PLDA to compute the similarity scores.
- **Cosine similarity scoring (Cosine):** We adopt the same dimension reduction as that in Table IV by LDA, and then use the dimension-reduced feature as the input of the cosine similarity scoring to make decisions.
- **PLDA-adp:** We conduct domain adaptation to the PLDA back-end of the 8 kHZ system by using an unlabeled major dataset in NIST 2016 SRE, which consists of 2,272 utterances. The adaptation technique is implemented in kaldi-master/egs/sre16 of Kaldi.
- **pAUCMetric:** We adopt the same dimension reduction as that in Table IV by LDA. Then, the speaker features are pre-processed according to Section V. At last, the preprocessed features are used as the input of pAUCMetric. The default hyperparameters of pAUCMetric are as follows. $\alpha = 0$, $\beta = 0.01$, $\mu = 10^{-3}$, $\eta = 10$, and $s = 500$. γ is set to 0.5 for the x-vector front-end, and set to $\gamma = 0.1$ for the i-vector front-end. As will be shown in Section VI-F2, pAUCMetric performs robustly with a wide range of hyperparameter settings.
- **TripletMetric:** The algorithm was proposed in the last paragraph of Section III-D. Its hyperparameter setting is the same as that of pAUCMetric.

TABLE IV
OUTPUT DIMENSIONS OF THE LDA IN THE BACK-ENDS, WHICH ARE THE DEFAULT VALUES OF KALDI

Systems	i-vector	x-vector
8 kHZ system	200 dim	150 dim
16 kHZ system	150 dim	128 dim

We evaluated the studied methods using the calibration-insensitive metrics, including the EER, minDCF with $P_{tar} = 0.01$ and equal costs of misses and false alarms, pAUC with $\alpha = 0$, $\beta = 0.01$ ($\mathrm{pAUC}_{[0,0.01]}$), AUC, and average precision (AP).

We also conducted an experiment in Section VI-E, where we evaluated the performance by the calibration-sensitive metrics, including the actDCF with $P_{tar} = 0.01$ and equal costs of misses and false alarms, and $\mathrm{C}_{\mathrm{llr}}$.

C. Results Based on PLDA-Based Preprocessing

This section presents the main experimental results of the pAUCMetric with the PLDA-based preprocessing technique. We evaluate both the 8 kHZ and 16 kHZ systems on the SRE16 and SITW datasets, which contains the following four evaluation schemes:

TABLE V
COMPARISON RESULTS OF PAUCMETRIC AND PLDA IN THE E1 EVALUATION SCHEME

		i-vector					x-vector				
	Back-ends	EER(%)	minDCF	$\text{pAUC}_{[0,0.01]}$	AUC	AP(%)	EER(%)	minDCF	$\text{pAUC}_{[0,0.01]}$	ACU	AP(%)
Cantonese	PLDA	10.29	0.654	0.570	0.964	69.38	6.78	0.531	0.689	0.982	80.24
	TripletMetric	10.22	0.667	0.559	0.965	68.62	6.42	0.527	0.695	0.983	80.93
	pAUCMetric	**9.52**	**0.649**	**0.578**	**0.969**	**70.58**	**6.00**	**0.503**	**0.717**	**0.986**	**82.60**
Tagalog	PLDA	**21.39**	**0.985**	**0.178**	**0.864**	**23.03**	**18.34**	**0.977**	**0.218**	**0.894**	**28.90**
	TripletMetric	22.05	0.985	0.175	0.859	22.36	18.42	0.980	0.216	0.894	28.21
	pAUCMetric	21.85	0.985	0.175	0.860	22.45	18.52	0.980	0.218	0.894	28.39

TABLE VI
COMPARISON RESULTS OF PAUCMETRIC AND PLDA-ADP IN THE E2 EVALUATION SCHEME

		i-vector					x-vector				
	Back-ends	EER(%)	minDCF	$\text{pAUC}_{[0,0.01]}$	AUC	AP(%)	EER(%)	minDCF	$\text{pAUC}_{[0,0.01]}$	AUC	AP(%)
Cantonese	PLDA-adp	8.91	0.597	0.625	0.970	74.35	4.80	0.400	0.800	0.990	88.26
	TripletMetric	8.21	0.586	0.641	0.976	76.00	4.34	0.391	0.810	0.992	88.98
	pAUCMetric	**7.93**	**0.577**	**0.646**	**0.977**	**76.60**	**4.19**	**0.379**	**0.818**	**0.993**	**89.58**
Tagalog	PLDA-adp	19.85	0.892	0.313	0.885	39.00	12.27	0.753	0.499	0.948	60.21
	TripletMetric	**18.95**	**0.892**	**0.322**	**0.894**	**40.23**	12.04	**0.750**	**0.506**	0.950	**60.80**
	pAUCMetric	19.11	0.896	0.315	0.892	39.48	**11.97**	0.754	0.503	**0.951**	60.59

- **E1:** This scheme conducts the comparison on language mismatched conditions. The evaluation is carried out with the 8 kHZ system on the SRE16 dataset. Most training data of the 8 kHZ system are in English, while the SRE16 test data are in Cantonese and Tagalog languages.
- **E2:** Contrary to E1, this scheme conducts the comparison on language matched conditions. The evaluation is carried out with the 8 kHZ system on the SRE16 dataset as well, and furthermore, the domain adaptation technique is adopted. The input features of pAUCMetric are the latent variables of PLDA-adp.
- **E3:** This scheme makes an evaluation on channel and noise mismatched conditions. We conducted the evaluation with the 8 kHZ system on the SITW data, where we downsampled the SITW from 16 KHZ to 8 KHZ. The mismatched problem is caused by the fact that SITW is collected from multi-media videos, while the training data, i.e., SWBD and SRE, are collected from telephone or meeting conditions.
- **E4:** Contrary to E3, this scheme makes an evaluation on channel and noise matched conditions. Specifically, we make the evaluation with the 16 kHZ system on the SITW dataset. Both the SITW and VoxCeleb datasets are collected from multi-media videos.

The experimental results of E1 are presented in Table V. As seen, pAUCMetric achieves obvious performance improvement over PLDA on the Cantonese language. Specifically, when the x-vector front-end is used, it obtains 11% relative EER reduction and 5% relative minDCF reduction; it also achieves 9% relative $\text{pAUC}_{[0,0.01]}$ improvement, 22% relative AUC improvement, and 12% relative AP improvement. When the i-vector front-end is used, it obtains 7% relative EER reduction and 14% relative AUC improvement. However, the experimental results of PLDA and pAUCMetric on the Tagalog language are not good, which may be due to the large mismatch between the Tagalog and the languages of the training data, as well as the fact that the Tagalog data is quite noisy.

The experimental results of E2 are presented in Table VI. It is seen that pAUCMetric yields better performance than PLDA-adp, for both the i-vector and x-vector front-ends. Specifically, when the x-vector front-end is applied to the Cantonese language of SRE16, pAUCMetric obtains 13% relative EER reduction and 5% relative minDCF reduction respectively; it also achieves 9% relative improvement in terms of $\text{pAUC}_{[0,0.01]}$ and 30% relative improvement in terms of AUC. When the i-vector front-end is applied to the Cantonese language, pAUCMetric also obtains 11% relative EER reduction and 23% relative AUC improvement, respectively. pAUCMetric also achieves better performance than PLDA-adp on the Tagalog language.

The experimental results of E3 are presented in Table VII. One can see that pAUCMetric achieves better performance than PLDA. Specifically, when the x-vector front-end is used, pAUCMetric achieves 10% relative EER reduction, 3% relative minDCF reduction, and more than 8% relative $\text{pAUC}_{[0,0.01]}$ improvement on both the Dev.Core and Eval.Core tasks; it also obtains more than 20% and 12% relative AUC improvement on the Dev.Core and Eval.Core tasks, respectively. When the i-vector front-end is used, it also achieves better performance than PLDA.

The experimental results of E4 are presented in Table VIII. From this table, one can see that pAUCMetric also yields better performance than PLDA. Specifically, when the x-vector front-end is used, it obtains approximately 10% relative EER reduction; it also obtains about 9% relative $\text{pAUC}_{[0,0.01]}$ improvement, and more than 20% relative AUC improvement. When the i-vector front-end is used, a similar experimental phenomenon is observed as well.

To summarize, when the x-vector front-end is used, pAUCMetric obtains about 10% relative EER reduction, 9% relative $\text{pAUC}_{[0,0.01]}$ improvement, and more than 20% relative

TABLE VII
COMPARISON RESULTS OF PAUCMETRIC AND PLDA IN THE E3 EVALUATION SCHEME

		i-vector					x-vector				
	Back-ends	EER(%)	minDCF	$\text{pAUC}_{[0,0.01]}$	AUC	AP(%)	EER(%)	minDCF	$\text{pAUC}_{[0,0.01]}$	AUC	AP(%)
Dev.Core	PLDA	9.20	0.619	0.590	0.972	62.01	6.85	0.513	0.697	0.985	72.31
	TripletMetric	9.70	0.624	0.583	0.970	61.05	6.43	0.515	0.697	0.986	72.39
	pAUCMetric	**8.73**	**0.605**	**0.600**	**0.974**	**62.99**	**5.91**	**0.500**	**0.724**	**0.988**	**74.87**
Eval.Core	PLDA	10.03	0.646	0.563	0.969	54.70	6.75	0.546	0.674	0.984	66.13
	TripletMetric	10.14	0.654	0.552	0.968	53.77	6.86	0.562	0.669	0.984	65.38
	pAUCMetric	**9.65**	**0.639**	**0.571**	**0.970**	**55.73**	**6.10**	**0.528**	**0.703**	**0.986**	**68.71**

TABLE VIII
COMPARISON RESULTS OF PAUCMETRIC AND PLDA IN THE E4 EVALUATION SCHEME

		i-vector					x-vector				
	Back-ends	EER(%)	minDCF	$\text{pAUC}_{[0,0.01]}$	AUC	AP(%)	EER(%)	minDCF	$\text{pAUC}_{[0,0.01]}$	AUC	AP(%)
Dev.Core	PLDA	5.30	**0.418**	0.772	0.989	79.90	2.96	0.301	0.868	0.996	88.69
	TripletMetric	5.27	0.429	0.765	0.989	79.32	2.77	0.309	0.862	0.996	88.22
	pAUCMetric	**4.93**	0.420	**0.776**	**0.990**	**80.3**	**2.58**	**0.289**	**0.880**	**0.997**	**89.71**
Eval.Core	PLDA	5.72	**0.453**	0.746	0.988	73.91	3.58	0.333	0.847	0.995	84.23
	TripletMetric	6.04	0.464	0.738	0.987	73.10	3.68	0.341	0.842	0.995	83.59
	pAUCMetric	**5.49**	0.456	**0.748**	**0.988**	**74.32**	**3.23**	**0.316**	**0.861**	**0.996**	**85.43**

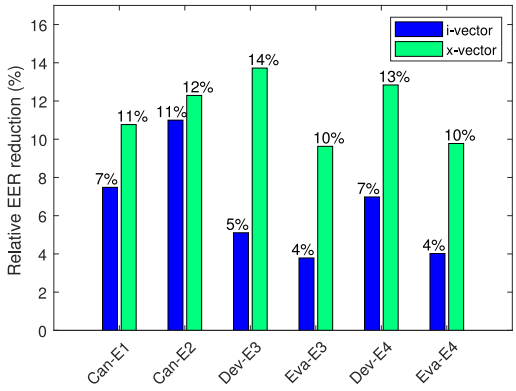

Fig. 4. Relative EER reduction of pAUCMetric over PLDA. The terms "Can," "Dev," and "Eva" denote the Cantonese data of SRE16, the Dev.Core and Eval.Core tasks of SITW, respectively. The terms "E1," "E2," "E3," and "E4" denote the four evaluation schemes.

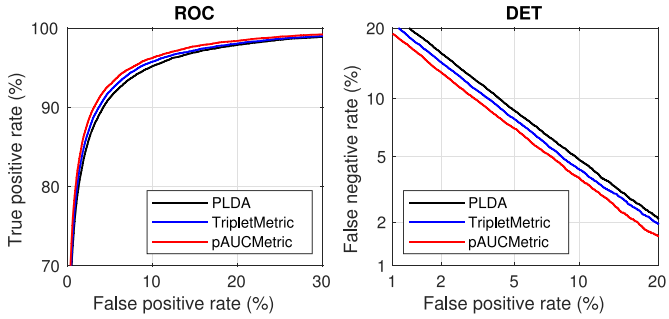

Fig. 5. ROC and DET curves of the comparison methods with the x-vector front-end on the Cantonese data of SRE16 in the E1 evaluation scheme.

AUC improvement over the state-of-the-art PLDA, except the Eval.Core task of the SITW dataset in the E3 evaluation scheme. Although the performance improvement with the i-vector front-end is not so significant as that with the x-vector front-end, the trends are consistent. For clarity, the relative EER improvement of pAUCMetric over PLDA in different evaluation schemes is summarized in Fig. 4. pAUCMetric also achieves better performance than TripletMetric in all of the above four conditions.

Figure 5 plots the ROC and DET curves of the comparison methods with the x-vector front-end in the SRE16 Cantonese of the E1 evaluation scheme. It is seen from the figure that pAUCMetric yields better ROC and DET curves than PLDA. We further draw the DET curves of the $\text{E2} \sim \text{E4}$ schemes in Appendix C, where we also see the effectiveness of pAUCMetric.

D. Results Based on Length-Normalization Preprocessing

This section presents the main experimental results of the pAUCMetric with the length-normalization preprocessing technique. We compare it with the Cosine back-end.

Specifically, we first evaluate the 8 kHZ system on the Cantonese data of SRE16 and the Dev.Core and Eval.Core tasks of SITW. The experimental results are summarized in Table IX. As shown in the table, pAUCMetric achieves significant performance improvement over the Cosine back-end. When the i-vector front-end is used, it obtains about 16% to 23% relative EER reduction, and approximately 2% to 7% minDCF reduction respectively; it also obtains about 7% to 13% relative improvement in terms of $\text{pAUC}_{[0,0.01]}$, and about 23% to 37% relative improvement in terms of AUC. When the x-vector front-end is used, pAUCMetric obtains more than 25% relative EER reduction; moreover, it obtains about 20% relative $\text{pAUC}_{[0,0.01]}$ improvement and more than 40% relative AUC improvement.

Then, we evaluate the 16 kHZ system on the Dev.Core and Eval.Core tasks of SITW. The experimental results are summarized in Table X. One can see that pAUCMetric also yields significant performance improvement over the Cosine back-end. For example, when the x-vector front-end is used, it obtains 27% and 30% relative EER reduction on the Dev.Core task and Eval.Core task respectively. It also obtains more than 40% relative AUC improvement on both of the tasks. The performance

TABLE IX
COMPARISON RESULTS OF PAUCMETRIC AND COSINE WITH THE MODELS OF THE 8 KHZ SYSTEM

		i-vector					x-vector				
	Back-ends	EER(%)	minDCF	$\text{pAUC}_{[0,0.01]}$	AUC	AP(%)	EER(%)	minDCF	$\text{pAUC}_{[0,0.01]}$	AUC	AP(%)
Cantonese	Cosine	13.68	0.744	0.467	0.940	59.10	9.25	0.606	0.613	0.968	73.23
	TripletMetric	11.78	0.715	0.507	0.954	63.60	6.89	0.551	0.677	0.981	79.34
	pAUCMetric	**10.57**	**0.689**	**0.537**	**0.962**	**66.75**	**6.35**	**0.523**	**0.700**	**0.984**	**81.20**
Dev.Core	Cosine	11.09	0.650	0.552	0.960	57.77	9.01	0.573	0.643	0.974	66.53
	TripletMetric	10.70	0.638	0.568	0.966	59.37	6.89	0.526	0.684	0.984	71.02
	pAUCMetric	**9.07**	**0.616**	**0.593**	**0.971**	**62.29**	**5.94**	**0.497**	**0.719**	**0.987**	**74.41**
Eval.Core	Cosine	11.86	0.684	0.519	0.956	49.13	8.53	0.600	0.617	0.974	59.92
	TripletMetric	10.83	0.692	0.527	0.962	50.07	7.20	0.576	0.648	0.982	62.82
	pAUCMetric	**10.00**	**0.668**	**0.553**	**0.966**	**52.88**	**6.40**	**0.539**	**0.689**	**0.985**	**67.09**

TABLE X
COMPARISON RESULTS OF PAUCMETRIC AND COSINE WITH THE MODELS OF THE 16 KHZ SYSTEM

		i-vector					x-vector				
	Back-ends	EER(%)	minDCF	$\text{pAUC}_{[0,0.01]}$	AUC	AP(%)	EER(%)	minDCF	$\text{pAUC}_{[0,0.01]}$	AUC	AP(%)
Dev.Core	Cosine	7.30	0.569	0.661	0.981	68.20	4.62	0.472	0.758	0.991	77.68
	TripletMetric	6.12	0.532	0.691	0.987	71.74	3.85	0.442	0.785	0.994	80.34
	pAUCMetric	**5.61**	**0.496**	**0.722**	**0.988**	**74.89**	**3.35**	**0.352**	**0.834**	**0.995**	**85.35**
Eval.Core	Cosine	7.45	0.606	0.616	0.979	60.60	5.41	0.465	0.744	0.989	73.15
	TripletMetric	6.40	0.581	0.644	0.984	62.94	4.54	0.444	0.769	0.992	75.69
	pAUCMetric	**5.96**	**0.547**	**0.679**	**0.986**	**66.51**	**3.80**	**0.374**	**0.825**	**0.994**	**81.48**

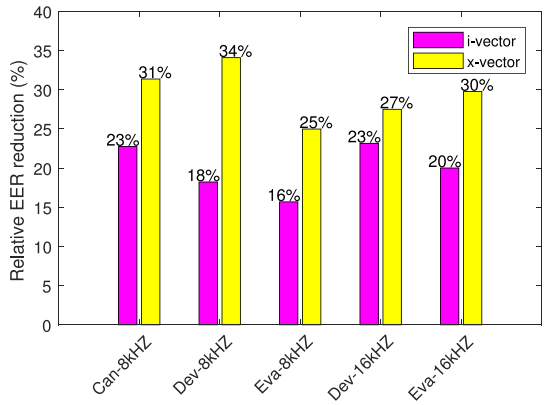

Fig. 6. Relative EER reduction of pAUCMetric over Cosine. The terms "Can," "Dev," and "Eva" denote the Cantonese data of SRE16, the Dev.Core and Eval.Core. tasks of SITW, respectively. The terms "8kHZ" and "16kHZ" denote the 8 kHZ system and 16 kHZ system respectively.

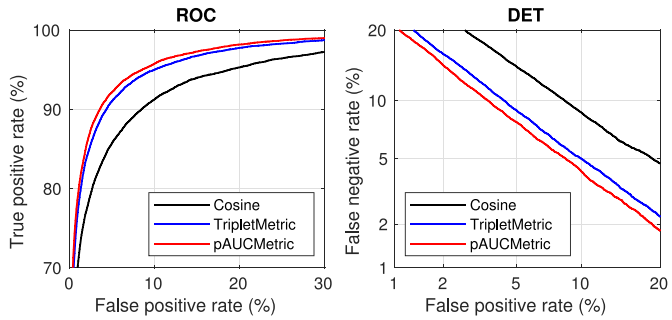

Fig. 7. ROC and DET curves of the comparison methods with the x-vector front-end of the 8 kHZ system on the Cantonese data of SRE16.

trend with the i-vector front-end is consistent with the trend with the x-vector front-end.

To summarize, when the length-normalization is adopted to preprocess the speaker features, pAUCMetric achieves significant performance improvement over the Cosine back-end. For clarity, the relative EER improvement on different evaluation dataset is summarized in Fig. 6. Moreover, the relative improvement of the pAUCMetric over PLDA with the x-vector front-end behaves better than that with the i-vector front-end. pAUCMetric also achieves better performance than TripletMetric, when the length-normalization preprocessing is adopted.

Figure 7 plots the ROC and DET curves of the comparison methods with the x-vector front-end on the SRE16 Cantonese data. It is seen from the figure that pAUCMetric yields better ROC and DET curves than Cosine.

E. Calibration

In real applications, it is needed to present the verification result in terms of calibrated LLR [57]. So, we applied calibration to the all the studied back-ends, i.e., Cosine, PLDA, TripletMetric, and pAUCMetric, with the linear logistic regression method of BOSARIS Toolkit,[2] where the calibration model was trained on the Dev.Core dataset of SITW and evaluated on its Eval.Core dataset.

We conducted experiments on the conditions of the 8 kHZ and 16 kHZ systems respectively. From the experimental results inTable XII, one can see that pAUCMetric achieves better performance than the compared methods on all the four conditions in terms of actDCF and C_{llr}.

F. Discussion

In this section, we first discuss the effect of the input feature dimension of pAUCMetric on performance, then analyze the

[2][Online]. Available: *https://sites.google.com/site/bosaristoolkit/*.

TABLE XI
EER RESULTS OF THE COMPARISON BACK-ENDS WITH DIFFERENT INPUT FEATURE DIMENSIONS. THE TERM "LENGTH-NORMALIZATION" DENOTES THE LENGTH NORMALIZATION PREPROCESSING. THE TERM "PLDA-BASED" DENOTES THE PLDA-BASED PREPROCESSING

	Back-ends	50 dim	100 dim	150 dim	200 dim	250 dim	300 dim	350 dim	400 dim
Length-normalization	Cosine	9.14	8.51	9.25	9.93	10.91	11.78	12.55	13.23
	pAUCMetric	8.80	6.85	6.35	6.50	6.62	6.90	7.17	7.43
PLDA-based	PLDA	8.36	6.72	6.82	7.50	8.13	8.77	9.26	9.67
	pAUCMetric	8.02	6.19	6.05	6.54	7.08	7.67	7.89	8.18

TABLE XII
SCORE CALIBRATION RESULTS ON THE X-VECTOR OF EVAL.CORE TASK OF THE SITW DATASET

		8 KHZ System		16 KHZ System	
Preprocessing	Back-ends	actDCF	C_{llr}	actDCF	C_{llr}
Length-normalization	Cosine	0.6029	0.3012	0.4708	0.1961
	TripletMetric	0.5775	0.2590	0.4494	0.1670
	pAUCMetric	**0.5404**	**0.2351**	**0.3762**	**0.1461**
PLDA-based	PLDA	0.5502	0.2364	0.3354	0.1300
	TripletMetric	0.5640	0.2404	0.3446	0.1353
	pAUCMetric	**0.5324**	**0.2209**	**0.3221**	**0.1199**

effects of its hyperparameters, and at last discuss the computational complexity and performance with respect to the batch size s.

All discussions use the x-vector front-end of the 8 kHZ system to extract speaker features, and compare PLDA with the pAUCMetric that adopts the PLDA-based preprocessing on the Cantonese data of SRE16. No domain adaptation is adopted in the discussions.

1) Effect of the Input Feature Dimension on Performance: We set the dimensions of the input features of the comparison back-ends from 50 to 400 with a step size of 50, where the features are produced from LDA. The experimental results are summarized in Table XI. From this table, one can see that pAUCMetric obtains lower EER scores and smaller performance variances than the comparison back-ends in all cases. It reaches the lowest EER when the input feature dimension is set to 150.

2) Effects of the Hyperparameters of pAUCMetric: pAUCMetric has five hyperparameters α, β, δ, γ, and μ. The reason why we set $\alpha = 0$ and $\beta = 0.01$ is that the number of the imposter trials is much larger than the number of the true trials, hence restricting the working area $[\alpha, \beta]$ to a FPR range of close to zero makes the algorithm focus on discriminating the difficult trials.

We study the effects of δ, γ, and μ by grid search. We first search δ in [0,10] with the other hyperparameters set to their default values. Figure 8 shows the relative performance improvement of pAUCMetric over PLDA. From the figure, we find that pAUCMetric is robust in a wide range of δ with the best δ being around 1.5. We search γ and μ in grid jointly as listed in Tables XIII and XIV with the other hyperparameters set to their default values. It is observed that the stable working region is $\mu \in [0, 10^{-3}] \cap \gamma \in [0, 1.5]$. Interestingly, pAUCMetric still works well even without regularization, i.e., $\mu = 0$ and $\gamma = 0$. The above observation is consistent across all training scenarios of this paper. To summarize, pAUCMetric is insensitive to the 3 hyperparameters.

3) Complexity Analysis: In Section VI-F3, we have proven that the computational complexity is cubic with respect to the batch size s. This section further discusses the effect of s on the computational complexity and performance of pAUCMetric.

Figure 9 shows the training time of *the pAUCMetric at each iteration* with respect to the batch size s. One can see that the training time increases sharply with the value of s, which is consistent with the theoretical analysis in Section VI-F3. Note

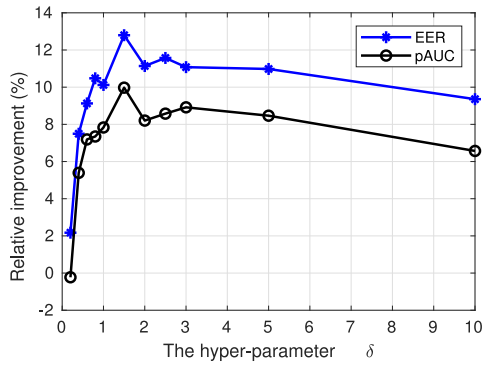

Fig. 8. Relative performance improvement of pAUCMetric over PLDA with respect to hyperparameter δ.

TABLE XIII
RELATIVE EER REDUCTION OF PAUCMETRIC OVER PLDA WITH RESPECT TO γ AND μ

γ \ μ	0	10^{-7}	10^{-6}	10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}
0	7.5	7.4	7.8	6.8	6.5	7.6	1.2	-5.9
0.01	7.8	8.8	7.8	7.5	7.6	7.6	1.7	-6.2
0.05	8.4	9.2	8.4	9.5	9.0	10.3	2.0	-6.1
0.10	9.4	7.4	9.4	9.1	9.3	10.2	1.7	-6.0
0.50	9.6	9.5	9.6	10.8	10.5	11.0	1.9	-6.2
1.00	8.0	10.6	8.0	9.5	8.5	10.9	2.1	-6.0
1.50	6.9	5.5	6.8	8.7	8.4	8.6	2.7	-5.8
3.00	-1.4	-3.3	-1.3	-3.6	-0.2	1.0	1.1	-5.6

TABLE XIV
RELATIVE $\text{pAUC}_{[0,0.01]}$ REDUCTION OF PAUCMETRIC OVER PLDA WITH RESPECT TO γ AND μ

γ \ μ	0	10^{-7}	10^{-6}	10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}
0	5.0	4.6	4.9	4.2	4.1	4.3	-0.6	-6.2
0.01	5.2	5.4	5.2	5.4	4.9	4.8	-0.6	-6.3
0.05	5.6	6.1	5.6	6.2	5.8	6.6	-0.1	-6.1
0.10	6.9	5.4	6.9	6.6	6.6	7.1	-0.1	-6.1
0.50	7.3	7.4	7.3	7.3	7.4	8.0	-0.2	-6.1
1.00	6.1	7.3	6.1	6.6	4.9	7.5	0.4	-6.1
1.50	4.6	3.8	4.6	2.9	6.0	5.6	1.5	-5.9
3.00	-3.5	-4.4	-1.8	-2.6	-3.2	-0.3	-0.7	-6.1

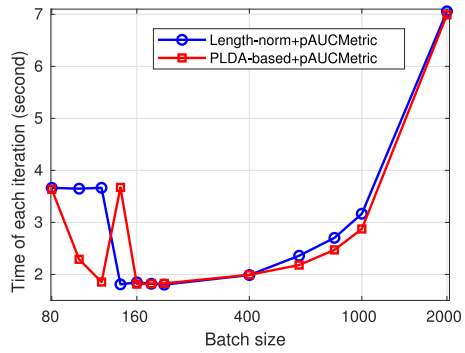

Fig. 9. Training time of pAUCMetric at each iteration with different batch sizes.

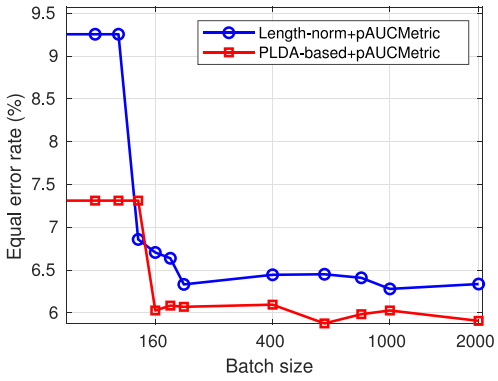

Fig. 10. EER of pAUCMetric with different batch sizes.

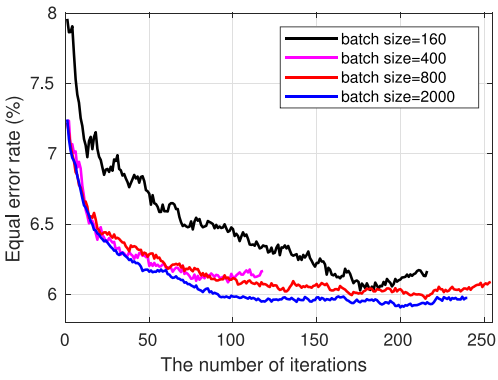

Fig. 11. Convergence analysis of pAUCMetric with different batch sizes.

that, when the value of s is less than 160, the fluctuation of the training time is caused by some random factors.

Figure 10 shows the EER results of pAUCMetric with different values of s. One can see that, on the one hand, the value of s cannot be too small, e.g. smaller than 160, and on the other hand, increasing the batch size does not always improve the performance. In practice, we only need a small suitable batch size, such as our default $s = 500$.

Figure 11 plots the convergence rate with respect to s. We see that, when s is larger than a reasonable small value, the convergence rate of pAUCMetric does not improve anymore. In other words, although the computational complexity of pAUCMetric is theoretically cubic with respect to s, setting s to a small reasonable value not only guarantees good performance but also is efficient.

Finally, we evaluated the proposed pAUCMetric in other test scenarios beyond the scenarios in this subsection and with the length-normalization preprocessing technique as well. The experimental conclusions are consistent with those in this subsection. But we will not report the tedious results to make the paper concise.

VII. Conclusion

In this paper, we presented a speaker verification back-end based on the squared Mahalanobis distance, i.e., pAUCMetric, to maximize pAUC. Because directly optimizing pAUC is an NP-hard problem, we first relaxed the optimization problem to a polynomial-time solvable one, and then adopted a random sampling strategy to reduce the computational complexity. The pAUC optimization was proven to be a problem of enlarging the weighted margin between the positive and negative trials, where the information of pAUC is encoded in the weights of the trials. In order to boost the performance of pAUCMetric, we further proposed to use the length-normalization and the PLDA-based preprocessing techniques. Experimental results on the NIST 2016 SRE and SITW data demonstrated the effectiveness of pAUCMetric and showed that pAUCMetric is insensitive to the hyperparameter settings in all the studied evaluation scenarios.

The proposed method can be further improved in many aspects. Work is in progress to study automatic hyperparameter tuning algorithms via auto machine learning and investigate new methods that do not need feature preprocessing. Since pAUCMetric does not need a decision threshold, it is interesting to explore whether the pAUC optimization can be integrated with score calibration, which will be carried out in the near future. A speaker verification system consists of both front-end and back-end. So, only developing a good back-end may not give the best performance. Consequently, it is legitimate to extend pAUCMetric to end-to-end training, which is also on our roadmap. Furthermore, as suggested by one anonymous reviewer, it is interesting to separate the effects of back-ends and loss functions, and evaluate how well the hinge loss can approximate the indicator function in pAUCMetric.

Appendix A

A probabilistic explanation of the Mahalanobis distance is given as follows. Let $\mathbf{z} = \mathbf{x}_1 - \mathbf{x}_2$ be the difference between two embedding vectors. We further assume that $p(\mathbf{z}|tar) = N(0, \boldsymbol{\Sigma}_0)$ and $p(\mathbf{z}|non) = N(0, \boldsymbol{\Sigma}_1)$, where "$tar$" and "$non$" denote target and non-target respectively. The LLR test is:

$$LLR(\mathbf{z}) = \log(p(\mathbf{z}|tar)) - \log(p(\mathbf{z}|non)), \tag{34}$$

which can be transformed to:

$$2LLR(\mathbf{z}) = -\mathbf{z}^T(\boldsymbol{\Sigma}_0^{-1} - \boldsymbol{\Sigma}_1^{-1})\mathbf{z} + \log|\boldsymbol{\Sigma}_1| - \log|\boldsymbol{\Sigma}_0|. \tag{35}$$

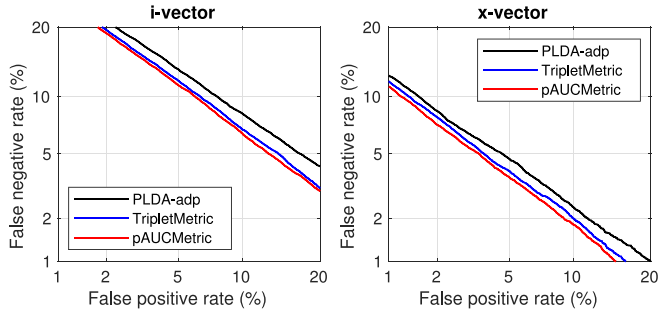

Fig. 12. DET curves of the studied methods on the Cantonese data of the E2 scheme.

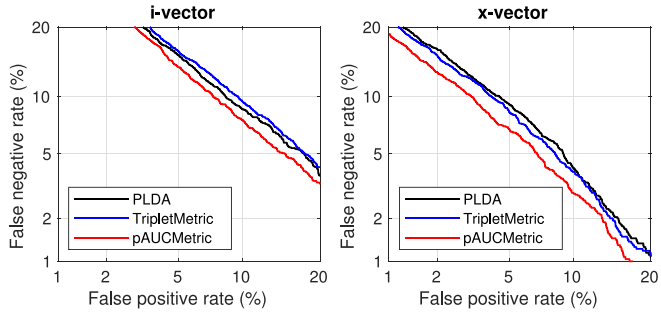

Fig. 13. DET curves of the studied methods on the Dev.Core task of the E3 scheme.

Neglecting the constant terms of (35) gives:

$$\widetilde{LLR}(\mathbf{z}) = -\mathbf{z}^T\mathbf{M}\mathbf{z} \tag{36}$$

where $\mathbf{M} = \mathbf{\Sigma}_0^{-1} - \mathbf{\Sigma}_1^{-1}$ is the parameters of the Mahalanobis distance [58].

APPENDIX B

Theorem 2: The relative constraints of the triplet-loss are a subset of the relative constraints of the AUC-loss.

Proof: Let $\mathbf{x}_i^m$ be the ith embedding vector of the mth speaker. The relative constraints of the triplet-loss Tri is:

$$Tri = \{(\mathbf{x}_i^m, \mathbf{x}_j^m; \mathbf{x}_k^n) | i \neq j, m \neq n\} \tag{37}$$

The relative constraints of the AUC-loss Tet can be divided into the following four sets:

$$Tet_1 = \{(\mathbf{x}_i^m, \mathbf{x}_j^m; \mathbf{x}_i^m, \mathbf{x}_k^n) | i \neq j, m \neq n\} \tag{38}$$

$$Tet_2 = \{(\mathbf{x}_i^m, \mathbf{x}_j^m; \mathbf{x}_j^m, \mathbf{x}_k^n) | i \neq j, m \neq n\} \tag{39}$$

$$Tet_3 = \{(\mathbf{x}_i^m, \mathbf{x}_j^m; \mathbf{x}_l^m, \mathbf{x}_k^n) | i \neq j \neq l, m \neq n\} \tag{40}$$

$$Tet_4 = \{(\mathbf{x}_i^m, \mathbf{x}_j^m; \mathbf{x}_l^t, \mathbf{x}_k^n) | i \neq j, m \neq t \neq n\} \tag{41}$$

with $Tet = Tet_1 \cup Tet_2 \cup Tet_3 \cup Tet_4$. Obviously, $Tet_1 \cup Tet_2 = Tri$, which derives $Tri \subset Tet$. ■

APPENDIX C

The DET curves of the studied methods on the E2 $\sim$ E4 schemes are plotted in Figs. 12 to 14.

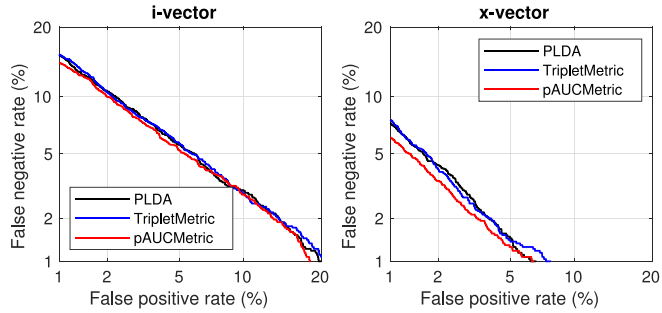

Fig. 14. DET curves of the studied methods on the Dev.Core task of the E4 scheme.

ACKNOWLEDGMENT

The authors are grateful to Dr. Kong Aik Lee, the Associate Editor, and the anonymous reviewers for their valuable comments, which helped greatly improve the quality of the paper.

REFERENCES

[1] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer, "End-to-end text-dependent speaker verification," in *Acoust., Speech and Signal Process., IEEE Int. Conf*, 2016, pp. 5115–5119.
[2] S.-X. Zhang, Z. Chen, Y. Zhao, J. Li, and Y. Gong, "End-to-end attention based text-dependent speaker verification," in *Proc. IEEE Spoken Lang. Technol. Workshop (SLT).*, 2016, pp. 171–178.
[3] D. Snyder, P. Ghahremani, D. Povey, D. G.-Romero, Y. Carmiel, and S. Khudanpur, "Deep neural network-based speaker embeddings for end-to-end speaker verification," in *Spoken Lang. Technol. Workshop , IEEE.* 2016, pp. 165–170.
[4] 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 Int. Conf. Acoust., Speech and Signal Process., IEEE*, 2018, pp. 5349–5353.
[5] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted Gaussian mixture models," *Digit. signal process.*, vol. 10, no. 1-3, pp. 19–41, 2000.
[6] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 19, no. 4, pp. 788–798, 2011.
[7] S. Cumani and P. Laface, "Speaker recognition using e–vectors," *IEEE/ACM Trans. Audio, Speech Lang. Process.*, vol. 26, no. 4, pp. 736–748, 2018.
[8] Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," in *Acoust., Speech and Signal Process., 2014 IEEE Int. Conf.*, 2014, pp. 1695–1699.
[9] P. Kenny, V. Gupta, T. Stafylakis, P. Ouellet, and J. Alam, "Deep neural networks for extracting baum-welch statistics for speaker recognition," in *Proc. Odyssey*, 2014, pp. 293–298.
[10] F. Richardson, D. Reynolds, and N. Dehak, "Deep neural network approaches to speaker and language recognition," *IEEE Signal Process. Lett.*, vol. 22, no. 10, pp. 1671–1675, 2015.
[11] Z. Tan, M.-W. Mak, B. K.-W. Mak, and Y. Zhu, "Denoised senone i-vectors for robust speaker verification," *IEEE/ACM Trans. Audio, Speech, and Lang. Process.*, vol. 26, no. 4, pp. 820–830, 2018.
[12] E. Variani, X. Lei, E. McDermott, I. L.-Moreno, and J. G.-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification." in *Int. Conf. on Acoust., Speech, and Signal Processing*, vol. 14. Citeseer, 2014, pp. 4052–4056.
[13] L. Li, Y. Chen, Y. Shi, Z. Tang, and D. Wang, "Deep speaker feature learning for text-independent speaker verification," in *Proc. Interspeech*, 2017, pp. 1542–1546.
[14] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep neural network embeddings for text-independent speaker verification," in *Proc. Interspeech*, 2017, pp. 999–1003.

[15] D. Snyder, D. G.-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *2018 IEEE Int. Conf. Acoust., Speech and Signal Process*. 2018.
[16] K. Okabe, T. Koshinaka, and K. Shinoda, "Attentive statistics pooling for deep speaker embedding," *arXiv:1803.10963*, 2018.
[17] 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.
[18] Z. Gao, Y. Song, I. McLoughlin, W. Guo, and L. Dai, "An improved deep embedding learning method for short duration speaker verification," *Proc. Interspeech 2018*, pp. 3578–3582.
[19] S. Yadav and A. Rai, "Learning discriminative features for speaker identification and verification," *Proc. Interspeech 2018*, pp. 2237–2241.
[20] N. Li, D. Tuo, D. Su, Z. Li, D. Yu, and A. Tencent, "Deep discriminative embeddings for duration robust speaker verification," *Proc. Interspeech 2018*, pp. 2262–2266.
[21] C. Zhang, K. Koishida, and J. H. Hansen, "Text-independent speaker verification based on triplet convolutional neural network embeddings," *IEEE/ACM Trans. Audio, Speech Lang. Process.*, vol. 26, no. 9, pp. 1633–1644, 2018.
[22] S. Cumani and P. Laface, "Large-scale training of pairwise support vector machines for speaker recognition," *IEEE/ACM Trans. Audio, Speech Lang. Process.*, vol. 22, no. 11, pp. 1590–1600, 2014.
[23] S. Ioffe, "Probabilistic linear discriminant analysis," in *Eur. Conf. Comput. Vision*, Springer, 2006, pp. 531–542.
[24] P. Kenny, "Bayesian speaker verification with heavy-tailed priors.," in *Odyssey*, 2010, p. 14.
[25] D. G.-Romero and C. Y. E.-Wilson, "Analysis of i-vector length normalization in speaker recognition systems," in *Twelfth Annu. Conf. the Int. Speech Commun. Assoc.*, 2011.
[26] O. Ghahabi and J. Hernando, "Deep belief networks for i-vector based speaker recognition," in *Acous., Speech and Signal Process., 2014 IEEE Int. Conf.* , 2014, pp. 1700–1704.
[27] O. Ghahabi and J. Hernando, "Deep learning backend for single and multisession i-vector speaker recognition," *IEEE/ACM Trans. Audio, Speech, and Lang. Process.*, vol. 25, no. 4, pp. 807–817, 2017.
[28] A. O. Hatch, S. Kajarekar, and A. Stolcke, "Within-class covariance normalization for SVM-based speaker recognition," in *Ninth int. conf. spoken lang. process.*, 2006.
[29] S. Cumani, P. Laface, S. Cumani, and P. Laface, "Nonlinear i-vector transformations for plda-based speaker recognition," *IEEE/ACM Trans. Audio, Speech Lang. Process.*, vol. 25, no. 4, pp. 908–919, 2017.
[30] S. Cumani and P. Laface, "Joint estimation of plda and nonlinear transformations of speaker vectors," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 25, no. 10, pp. 1890–1900, 2017.
[31] S. Cumani and P. Laface, "Scoring heterogeneous speaker vectors using nonlinear transformations and tied PLDA models," *IEEE/ACM Trans. Audio, Speech Lang. Process.*, vol. 26, no. 5, pp. 995–1009, 2018.
[32] Z. Tieran, H. Jiqing, and Z. Guibin, "Deep neural network based discriminative training for i-vector/PLDA speaker verification," in *2018 IEEE Int. Conf. Acoust., Speech Signal Process.*, *IEEE*, 2018, pp. 5354–5358.
[33] B. Kulis, "Metric learning: A survey," *Foundations and Trends in Mach. Learn.*, vol. 5, no. 4, pp. 287–364, 2013.
[34] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep metric learning via lifted structured feature embedding," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2016, pp. 4004–4012.
[35] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *Proc. Int. Workshop Similarity-Based Pattern Recognit*. Springer, 2015, pp. 84–92.
[36] F. Cakir, K. He, X. Xia, B. Kulis, and S. Sclaroff, "Deep metric learning to rank," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2019, pp. 1861–1870.
[37] Z. Bai, X.-L. Zhang, and J. Chen, "Cosine metric learning for speaker verification in the i-vector space," *Proc. Interspeech 2018*, pp. 1126–1130, 2018.
[38] 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.
[39] L. P. G.-Perera, J. A. N.-Flores, B. Raj, and R. Stern, "Optimization of the det curve in speaker verification," in *2012 IEEE Spoken Lang. Technol. Workshop*. *IEEE*, 2012, pp. 318–323.
[40] 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," 2019, *arXiv:1901.11332*.
[41] N. Brümmer and E. De Villiers, "The Bosaris toolkit: Theory, algorithms and code for surviving the new dcf," 2013, *arXiv:1304.2865*.
[42] N. Brümmer and J. Du Preez, "Application-independent evaluation of speaker detection," *Comput. Speech & Lang.*, vol. 20, no. 2-3, pp. 230–275, 2006.
[43] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki, "The DET curve in assessment of detection task performance," Nat. Inst of Standards and Technol. Gaithersburg MD, Tech. Rep., 1997.
[44] N. Brummer, "Measuring, refining and calibrating speaker and language information extracted from speech," Ph.D. dissertation, Stellenbosch: University of Stellenbosch, 2010.
[45] N. Brümmer and G. Doddington, "Likelihood-ratio calibration using prior-weighted proper scoring rules," 2013, *arXiv:1307.7981*.
[46] J. Huo, Y. Gao, Y. Shi, and H. Yin, "Cross-modal metric learning for AUC optimization," *IEEE Trans. Neural Netw. Learn. Syst., PP (99)*, pp. 1–13, 2018.
[47] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *J. Mach. Learn. Res.*, vol. 10, no. Feb. pp. 207–244, 2009.
[48] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *Proc. 24th int. conf. Mach. learn*. ACM, 2007, pp. 209–216.
[49] L. Burget, O. Plchot, S. Cumani, O. Glembek, P. Matějka, and N. Brümmer, "Discriminatively trained probabilistic linear discriminant analysis for speaker verification," in *2011 IEEE int. conf. acoust., speech and signal process*, 2011, pp. 4832–4835.
[50] S. Cumani, N. Brümmer, L. Burget, and P. Laface, "Fast discriminative speaker verification in the i-vector space," in *2011 IEEE Int. Conf. Acoust., Speech and Signal Process.* 2011, pp. 4852–4855.
[51] N. Brümmer and E. De Villiers, "The speaker partitioning problem.," in *Odyssey*, 2010, p. 34.
[52] A. N. snd Joon Son Chung and A. Zisserman, "VoxCeleb: a large-scale speaker identification dataset," in *Proc. Interspeech 2017*, 2017, pp. 1487–1491.
[53] 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
[54] "Nist 2016 speaker recognition evaluation plan," https://www.nist.gov/itl/iad/mig/speaker-recognition-evaluation-2016, 2016.
[55] M. McLaren, L. Ferrer, D. Castan, and A. Lawson, "The speakers in the wild (sitw) speaker recognition database," in *Interspeech*, 2016, pp. 818–822.
[56] D. Povey *et al.*, "The Kaldi speech recognition toolkit," in *IEEE 2011 workshop on autom. speech recognit. understanding*, no. EPFL-CONF-192584. IEEE Signal Process. Soc., 2011.
[57] A. Khosravani and M. M. Homayounpour, "AUT System for SITW Speaker Recognition Challenge." in *INTERSPEECH*, 2016, pp. 843–847.
[58] M. Koestinger, M. Hirzer, P. Wohlhart, P. M. Roth, and H. Bischof, "Large scale metric learning from equivalence constraints," in *2012 IEEE Conf. Comput. Vision Pattern Recognit*. IEEE, 2012, pp. 2288–2295.

Zhongxin Bai received the bachelor's degree in electronics and information engineering and the master's degree in information and communication engineering from the Northwestern Polytechnical University(NPU), Xi'an, China, in 2015 and 2017, respectively, where he is currently working toward the Ph.D. degree in information and communication engineering. His research interests include speech enhancement, speaker recognition and machine learning.

Xiao-Lei Zhang received the Ph.D. degree in information and communication engineering from Tsinghua University, Beijing, China, in 2012. He is currently a Full Professor with the Center for Intelligent Acoustics and Immersive Communications, and the School of Marine Science and Technology, Northwestern Polytechnical University, Xian, China. He was a Postdoctoral Researcher with the Perception and Neurodynamics Laboratory, The Ohio State University.

His research interests include audio and speech signal processing, machine learning, statistical signal processing, and artificial intelligence. He has published over 40 journal articles and conference papers in Neural Networks, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, IEEE/ACM TRANSACTION ON AUDIO, SPEECH, AND LANGUAGE PROCESSING, IEEE TRANSACTIONS ON CYBERNETICS, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, Part B: Cybernetics, ICASSP, Interspeech, etc. and co-edited a text book in statistics. He received the first-class Beijing Science and Technology Award. He serves as an Associate Editor of *Neural Networks* and *EURASIP Journal on Audio, Speech, and Music Processing*. He is a member of ISCA.

Jingdong Chen (Senior Member, IEEE) received the Ph.D. degree in pattern recognition and intelligence control from the Chinese Academy of Sciences in 1998.

From 1998 to 1999, he was with ATR Interpreting Telecommunications Research Laboratories, Kyoto, Japan, where he conducted research on speech synthesis, speech analysis, as well as objective measurements for evaluating speech synthesis. He then joined the Griffith University, Brisbane, Australia, where he engaged in research on robust speech recognition and signal processing. From 2000 to 2001, he worked at ATR Spoken Language Translation Research Laboratories on robust speech recognition and speech enhancement. From 2001 to 2009, he was a member of Technical Staff at Bell Laboratories, Murray Hill, New Jersey, working on acoustic signal processing for telecommunications. He subsequently joined WeVoice Inc. in New Jersey, serving as the Chief Scientist. He is currently a professor at the Northwestern Polytechnical University in Xi'an, China. His research interests include array signal processing, adaptive signal processing, speech enhancement, adaptive noise/echo control, signal separation, speech communication, and artificial intelligence.

Dr. Chen served as an Associate Editor of the IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING from 2008 to 2014 and as a Technical Committee (TC) Member of the IEEE Signal Processing Society (SPS) TC on Audio and Electroacoustics from 2007 to 2009. He is currently a member of the IEEE SPS TC on Audio and Acoustic Signal Processing, and a Series Editor of Springer Series on Signals and Communication Technology. He was the General Co-Chair of ACM WUWNET 2018 and IWAENC 2016, the Technical Program Chair of IEEE TENCON 2013, a Technical Program Co-Chair of IEEE WASPAA 2009, IEEE ChinaSIP 2014, IEEE ICSPCC 2014, and IEEE ICSPCC 2015, and helped organize many other conferences. He co-authored 12 monograph books including *Array Processing–Kronecker Product Beamforming*, (Springer, 2019), *Fundamentals of Signal Enhancement and Array Signal Processing*, (Wiley, 2018), *Fundamentals of Differential Beamforming*, (Springer, 2016), *Design of Circular Differential Microphone Arrays* (Springer, 2015), *Noise Reduction in Speech Processing* (Springer, 2009), *Microphone Array Signal Processing* (Springer, 2008), and *Acoustic MIMO Signal Processing* (Springer, 2006), etc.

Dr. Chen received the 2008 Best Paper Award from the IEEE Signal Processing Society (with Benesty, Huang, and Doclo), the Best Paper Award from the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics in 2011 (with Benesty), the Bell Labs Role Model Teamwork Award twice, respectively, in 2009 and 2007, the NASA Tech Brief Award twice, respectively, in 2010 and 2009, and the Young Author Best Paper Award from the 5th National Conference on Man-Machine Speech Communications in 1998. He is a Co-Author of a paper for which C. Pan received the IEEE R10 (Asia-Pacific Region) Distinguished Student Paper Award (First Prize) in 2016. He was also a recipient of the Japan Trust International Research Grant from the Japan Key Technology Center in 1998 and the "Distinguished Young Scientists Fund" from the National Natural Science Foundation of China (NSFC) in 2014.