When Speaker Recognition Meets Noisy Labels: Optimizations for Front-ends and Back-ends

Lin Li [1], Fuchuan Tong [2,2], and Qingyang Hong [1]

[1]Affiliation not available
[2]School of Information Science and Engineering

October 31, 2023

Abstract

A typical speaker recognition system often involves two modules: a feature extractor front-end and a speaker identity back-end. Despite the superior performance that deep neural networks have achieved for the front-end, their success benefits from the availability of large-scale and correctly labeled datasets. While label noise is unavoidable in speaker recognition datasets, both the front-end and back-end are affected by label noise, which degrades the speaker recognition performance. In this paper, we first conduct comprehensive experiments to help improve the understanding of the effects of label noise on both the front-end and back-end. Then, we propose a simple yet effective training paradigm and loss correction method to handle label noise for the front-end. We combine our proposed method with the recently proposed Bayesian estimation of PLDA for noisy labels, and the whole system shows strong robustness to label noise. Furthermore, we show two practical applications of the improved system: one application corrects noisy labels based on an utterance's chunk-level predictions, and the other algorithmically filters out high-confidence noisy samples within a dataset. By applying the second application to the NIST SRE0410 dataset and verifying filtered utterances by human validation, we identify that approximately 1% of the SRE04-10 dataset is made up of label errors.

When Speaker Recognition Meets Noisy Labels: Optimizations for Front-ends and Back-ends

Lin Li, *Member, IEEE,* Fuchuan Tong, and Qingyang Hong, *Member, IEEE*

***Abstract*—A typical speaker recognition system often involves two modules: a feature extractor front-end and a speaker identity back-end. Despite the superior performance that deep neural networks have achieved for the front-end, their success benefits from the availability of large-scale and correctly labeled datasets. While label noise is unavoidable in speaker recognition datasets, both the front-end and back-end are affected by label noise, which degrades the speaker recognition performance. In this paper, we first conduct comprehensive experiments to help improve the understanding of the effects of label noise on both the front-end and back-end. Then, we propose a simple yet effective training paradigm and loss correction method to handle label noise for the front-end. We combine our proposed method with the recently proposed Bayesian estimation of PLDA for noisy labels, and the whole system shows strong robustness to label noise. Furthermore, we show two practical applications of the improved system: one application corrects noisy labels based on an utterance's chunk-level predictions, and the other algorithmically filters out high-confidence noisy samples within a dataset. By applying the second application to the NIST SRE04-10 dataset and verifying filtered utterances by human validation, we identify that approximately 1% of the SRE04-10 dataset is made up of label errors.**

***Index Terms*—speaker recognition, noisy labels, x-vector, Probabilistic Linear Discriminant Analysis.**

I. Introduction

SPEAKER recognition is a typical biometric authentication technology that verifies the identities of speakers from their voices. A typical speaker recognition system often involves two modules: a feature extractor front-end and a speaker identity back-end. The front-end extracts low-dimensional discriminative speaker representations (embeddings) from length-variable utterances, whereas the back-end determines whether two embeddings are from the same speaker [1].

Conventional speaker recognition front-ends are based on Baum–Welch statistics, and the i-vector [2] is one of the typical front-ends, which is trained in an unsupervised manner. Probabilistic Linear Discriminant Analysis (PLDA) [3]–[6] is commonly used as a back-end scoring model. To satisfy the PLDA Gaussian assumptions for training data [7], extracted features generally require pre-processing, including Linear Discriminant Analysis (LDA) and length regularization [8], before being used to train a PLDA model. Both LDA and PLDA are trained in a supervised manner that requires training data with corresponding speaker labels.

Along with the increasing amount of training data and the development of neural networks, state-of-art performances for speaker recognition have been achieved by deep neural networks [9]. Among these networks, the x-vector [10] is perhaps the most popular deep speaker embedding architecture. The x-vector directly replaces the i-vector to extract discriminative speaker representations using time delay neural network (TDNN) layers [11] and a statistical pooling layer. Based on the x-vector architecture, multiple deep speaker embedding network variants [12], [13] have been proposed to boost recognition performance. In addition, margin-based objective functions [14], [15] have been widely used to learn more discriminative speaker representations. Although these methods have achieved remarkable success, the supervised training for deep embedding models requires large-scale datasets that are correctly labeled.

Unfortunately, erroneously labeled samples are unavoidable during speaker utterance collections by web spider or crowdsourcing. This phenomenon is denoted as label noise, the incorrectly labeled utterances are denoted as noisy samples and the corresponding labels are noisy labels. For instance, the NIST SRE18 [16] development set does not provide speaker labels but instead only provides a phone number corresponding to each utterance [17]. The VoxCeleb dataset [18] are collected from YouTube, and the speaker identities are confirmed through facial recognition based on convolutional neural networks (CNN). Data collections such as these often lead to label noise. Typically, these noisy labels can be categorized into two categories: *closed-set*, i. e., noisy samples whose true labels are contained within the training classes; and *open-set*, i. e., noisy samples whose true labels are outside the training set. Both types of label noise would impair both the front-end and back-end model training, thereby degrading the speaker recognition performance. A recent study [19] shows that network learning with closed-set noise is more challenging, so in this paper, we focus more on this type of noise. Although mislabeled samples can be manually eliminated by human validation, this would be extremely time-consuming and costly. Making models robust to label noise is a more practical solution.

For a training dataset with an unknown number of noisy samples, the front-end goal is to learn discriminative feature spaces where different speaker embeddings are well separated, this is so-called *learning with label noise*. However, such a goal is practically challenging, as the high capacity of deep

This work was partially supported by the National Natural Science Foundation of China under Grants No.61876160 and No.62001405. *(Corresponding authors: Qingyang Hong; Lin Li.)*

Lin Li and Fuchuan Tong are with the XMU Speech Lab and the School of Electronic Science and Engineering, Xiamen University, Xiamen 361005, China (e-mail: lilin@xmu.edu.cn; tongfuchuan@stu.xmu.edu.cn).

Qingyang Hong is with the XMU Speech Lab and the School of Informatics, Xiamen University, Xiamen 361005, China (e-mail: qyhong@xmu.edu.cn).

networks makes them capable of memorizing noisy labels even if the labels are completely random [20]. Nonetheless, recent studies have shown that deep neural networks would first learn clean samples and general patterns from a dataset, and then they would be forced to memorize noisy labels [20], [21].

Recently, several approaches for learning with label noise have been proposed in the computer vision community [22]–[27]. Paper [28] provides a comprehensive overview of recently proposed approaches. Although the literature on label noise for speaker recognition is relatively small, this topic is beginning to receive attention from researchers. For instance, in the x-vector front-end, the detrimental effects of label noise for speaker recognition are confirmed in [29], and the author proposed modifying the entropy loss to relax the constraints of the speaker identity function to avoid fitting noisy samples. Pham *et al.* [30] conducted extensive experiments based on VoxCeleb2 to investigate the effects of different types of label noise on the x-vector extractor. In addition, for the back-end, Borgström *et al.* [17] proposed a novel method for Bayesian estimation of PLDA when training labels are noisy (we refer to this method as NL-PLDA). However, the existing literature either focuses only on the front-end or the back-end, and does not comprehensively analyze the impact of label noise on different components for speaker recognition systems.

In this paper, we jointly optimize the front-end and back-end for speaker recognition systems when training datasets are noisy. For the network front-end, we propose a simple yet effective training paradigm to prevent networks from fitting noisy samples. The proposed framework consists of three major components: 1) A label confidence training scheme that incorporates network predicted pseudo-labels into the loss function; this method is similar to Bootstrapping [22], but we use a well-designed dynamically increasing confidence weight; 2) a re-scaling strategy that reduces the posterior probability of clean labels to emphasize them more in the loss function; 3) an improved AM-Softmax loss function that relaxes the intra-class constraint. For the back-end, we treat the true labels as multinomial random variables and train an NL-PLDA model to perform speaker identification scoring.

This paper extends the study of our previous work on combating noisy labels [31]. Instead of conducting experiments on the VoxCeleb dataset in [31], the experiments in this paper are conducted on Switchboard and NIST04-10 datasets. Besides, more comprehensive experiments are conducted to analyze the effects of label noise on the x-vector, LDA, PLDA, and NL-PLDA models by setting different label error rates under both the close-set and open-set label noise scenarios. The contributions of this paper are multi-fold, which can be summarized as follows:

1) In the front-end, a label confidence training paradigm with a dynamic confidence policy, a re-scaling strategy, and an improved AM-Softmax are proposed for front-end learning when label noise is present. In combination with these components, the network shows consistent improvements in the robustness of label noise. Furthermore, a label correction method based on chunk-level label predictions is proposed that significantly reduces the number of noisy samples in a dataset.

2) In the back-end, we show how to apply NL-PLDA to filter out noisy labels. For further practical contributions, we provide an optimized expectation-maximization (EM) algorithm and pseudo-code for the NL-PLDA training process.

3) To verify whether there are noisy samples in the SRE04-10 dataset, we utilize the network prediction and NL-PLDA estimation to filter out high-confidence noisy samples and then verify them by human validation. As a result, we find that approximately 1% of the samples in the SRE04-10 dataset are mislabeled. After removing these samples, a relatively correct *spk2utt* mapping is released for this dataset.

This paper is organized as follows. Section II reviews the three most popular models used in speaker recognition systems. Section III introduces our proposed method for front-end learning with label noise. A detailed EM algorithm description for NL-PLDA is presented in Section IV and Appendix A. Section V shows comprehensive experiments, results, and analyses. Section VI shows applications of the proposed method. Conclusions are given in Section VII.

II. Embedding-based Speaker Recognition

A. X-vector

Nowadays, deep learning-based speaker recognition is of increasing interest to researchers. The x-vector is a typical architecture that extracts discriminative low-dimensional vectors for speakers through a neural network. Benefiting from a large amount of data, the x-vector shows superior performance over the i-vector, and it is the main focus of this paper. The x-vector typically consists of frame-level layers, a pooling layer, segment-level layers, and a softmax layer. The frame-level layers process length-variable speech acoustic frames using TDNN. The pooling layer aggregates length-variable frames into a fixed-dimensional vector. The segment-level layers are generally composed of two fully connected layers, and the outputs of the two layers are so-called x-vectors.

To deal with long-duration and length-variable training data, acoustic sequences are usually cut into multiple small chunks, and then they are used as inputs to train a network. During training, an objective function computes cross-entropy between given speaker labels and corresponding output probabilities. In addition to the standard softmax objective function, the additive margin softmax (AM-Softmax or CosFace) [14], [32] and the additive angular margin softmax (AAM-Softmax or ArcFace) [15] are two commonly used loss functions. The back-propagation-based optimization algorithm updates the parameters of a network by minimizing the loss function. However, in the presence of noisy labels, this loss function might drive a speaker network to learn the opposite. Therefore, improving the loss function to prevent a network from fitting noisy samples is the key for learning with noisy labels.

B. LDA

LDA is a supervised method to reduce feature dimensions, which is useful for classification tasks, therefore, it is widely used in image recognition and speaker recognition. LDA maximizes the Fisher criterion [33] for subspace embeddings by projecting high-dimensional features into low-dimensional features through a projection matrix $\mathbf{P}$, i. e., LDA maximizes the between-class variance and minimizes the within-class variance. LDA is trained by optimizing the following Fisher criterion function:

$$\hat{\mathbf{P}} = \arg\max_{\mathbf{P}} \frac{\operatorname{tr}\left(\mathbf{P}^{\mathrm{T}}\mathbf{S}_b\mathbf{P}\right)}{\operatorname{tr}\left(\mathbf{P}^{\mathrm{T}}\mathbf{S}_w\mathbf{P}\right)}, \tag{1}$$

where $\mathbf{S}_b$ and $\mathbf{S}_w$ denote between-class and within-class variance, respectively. They are calculated as

$$\mathbf{S}_b = \frac{1}{N}\sum_{m=1}^{M} N_m \left(\boldsymbol{\mu}_m - \boldsymbol{\mu}\right)\left(\boldsymbol{\mu}_m - \boldsymbol{\mu}\right)^{\mathrm{T}} \tag{2}$$

$$\mathbf{S}_w = \frac{1}{N}\sum_{m=1}^{M}\sum_{n=1}^{N_m} \left(\mathbf{x}_n^m - \boldsymbol{\mu}_m\right)\left(\mathbf{x}_n^m - \boldsymbol{\mu}_m\right)^{\mathrm{T}}, \tag{3}$$

where N is the total number of embeddings from M speakers, N_m is the number of samples of the m-th speaker, $\boldsymbol{\mu}_m$ denotes the mean of the m-th speaker, $\boldsymbol{\mu}$ denotes the global mean, and $\mathbf{x}_n^m$ represents the n-th embedding of the m-th speaker.

LDA is typically performed as a channel compensation technology for both the i-vector and x-vector [2] [7]. However, since LDA is a supervised model, we explore how label noise affects LDA in Section V-E.

C. PLDA

PLDA is a probabilistic generative model typically used to make probabilistic inferences about the class of data. It is a probabilistic version of LDA. Compared to LDA, PLDA adds a continuous Gaussian Prior to class centers, which enables it to generate new unseen class centers given even a single example. Besides the standard PLDA formulation [4], there are several variants of PLDA [34], such as simplified variant [5], two-covariance variant [3], [5], and heavy-tailed PLDA [6]. In this paper, we adopt two-covariance PLDA, which is assumed to generate the class center and the observed data in a two-stage process:

$$\mathbf{y}_m \sim \mathcal{N}\left(\mathbf{y}_m | \boldsymbol{\mu}, \boldsymbol{\Sigma}_b^{-1}\right) \tag{4}$$

$$\mathbf{x}_n^m \sim \mathcal{N}\left(\mathbf{x}_n^m | \mathbf{y}_m, \boldsymbol{\Sigma}_w^{-1}\right), \tag{5}$$

where $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}_b^{-1}$ and $\boldsymbol{\Sigma}_w^{-1}$ are the parameters estimated by PLDA, namely the global mean, between-speaker, and within-speaker covariance matrices, respectively.

In the hypothesis-testing stage, given a pair of individual embeddings, we can decide whether or not two given embeddings belong to the same speaker by computing a likelihood ratio. Although making such a decision based on cosine distance is a simpler way, but its performance might be suboptimal under more challenging situations (e. g., cross-channel, language mismatch, and noisy environments). While thanks to the multiple PLDA domain adaptation technologies [35]–[37], and data augmentation methods [10], [38], PLDA shows its superior advantages. Besides, PLDA proved to be the theoretically optimal scoring method for speaker recognition [39]. Therefore, it is currently the dominant back-end algorithm for speaker recognition. In addition, for the problem of noisy PLDA training labels, a novel Bayesian estimation method has been proposed in [17], and we detail this method in Section IV.

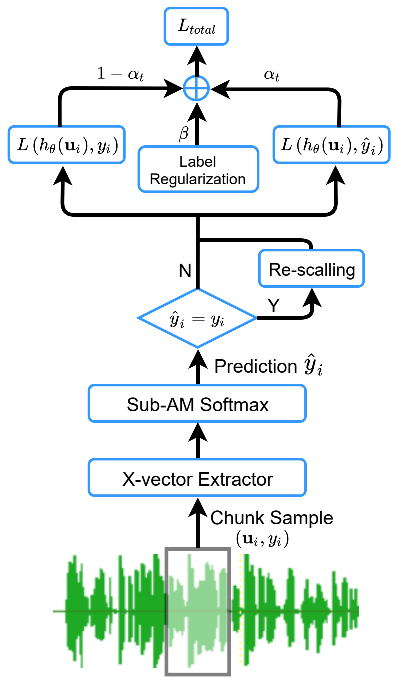


Fig. 1. An overview of the proposed method. During training, the SubAM-Softmax will output the predicted label of each chunk, and the total loss is formed by a confidence weighted combination of predicted and original labels.

III. Front-End Learning with Noisy Labels

We propose a proper training scheme and loss correction method to improve the noise-tolerant capacity of the front-end. As illustrated in Fig. 1, our framework consists of three major components: 1) incorporating pseudo-labels into a loss function with a well-designed dynamic confidence policy that weighs the combination of pseudo-labels and original labels; 2) re-scaling clean-label posterior probability; 3) introducing a sub-center layer into the AM-Softmax to separate noisy samples from training data.

A. Label Confidence Training

To put this formally, let us first rethink the deep embedding-based speaker recognition systems from a classification perspective. The x-vector network training process is formulated as a problem of learning a model $h_\theta(\mathbf{u})$ from a set of batch training samples $\mathcal{D} = \{(\mathbf{u}_i, y_i)\}_{i=1}^{B}$, where B is the mini-batch size, $y_i \in \{0,1\}^M$ denotes the ground-truth label corresponding to $\mathbf{u}_i$, and M is the total number of classes. For classification issues concerning label noise, label y_i might be noisy (i. e., $\mathbf{u}_i$ is a noisy sample). Supposing the extracted embedding of $\mathbf{u}_i$ is $\mathbf{x}_i$, the parameters of the network would be updated by optimizing the following loss function:

$$L = -\frac{1}{B}\sum_{i=1}^{B}\log\left(P_{i,y_i}\right), \tag{6}$$

where P_{i,y_i} denotes the posterior probability that $\mathbf{x}_i$ is classified as the ground-truth label y_i. In this paper, we term P_{i,y_i}

as the *ground-truth label posterior probability*; if adopting the AM-Softmax, P_{i,y_i} is formulated as:

$$P_{i,y_i} = \frac{e^{s\left(\cos\left(\theta_{y_i,i}\right)-m\right)}}{e^{s\left(\cos\left(\theta_{y_i,i}\right)-m\right)} + \sum_{j\neq y_i}^{M} e^{s\left(\cos\left(\theta_{j,i}\right)\right)}}, \tag{7}$$

where $\theta_{j,i}$ is the angle between $\mathbf{W}_j$ and $\mathbf{x}_i$. $\cos\left(\theta_j, i\right) = \mathbf{W}_j^{\mathrm{T}}\mathbf{x}_i$ represents the similarity score, and $\mathbf{W}_j$ is the j-th class center vector of the fully connected layer matrix $\mathbf{W}$. It is noted that here $\mathbf{W} \in \mathbb{R}^{M\times d}$, whereas in Section III-C, the dimension is extended to $\mathbb{R}^{M\times K\times d}$ based on a sub-center layer.

However, a neural network trained directly on this objective function will overfit noisy samples, as the loss contains noisy labels. Nonetheless, we may observe that a network maintains highly generative performance without memorizing noisy labels at the beginning of the training process; an example is shown in Fig. 2, Section V-C, where the prediction accuracy for clean data is higher than for noisy data. This situation indicates that the network clusters noisy samples into correct classes; therefore, to leverage this ability, we incorporate a *predicted label posterior probability* $P_{i,\hat{y}_i}$ into the objective function to prevent fitting incorrect samples as training iterations become larger. The subscript $\hat{y}_i \in \{0, 1, ..., M-1\}$ denotes the predicted label of $\mathbf{x}_i$, which is categorized as class j with the max-activated output. Then, the loss function in Eq. (6) is extended as follows:

$$L' = -\frac{1}{B}\sum_{i=1}^{B}\left\{(1-\alpha_t)\log\left(P_{i,y_i}\right) + \alpha_t \log\left(P_{i,\hat{y}_i}\right)\right\}, \tag{8}$$

where $\alpha_t \in [0, 1]$ is the t-th training iteration confidence weight between P_{i,y_i} and $P_{i,\hat{y}_i}$, and it determines whether the loss function relies more on the ground-truth label or the predicted label. This method is similar to Bootstrap [22]. Bootstrap sets α_t as a fixed small value for all iterations, and it maintains the effects of noisy labels during the whole training process; thus, the performance is suboptimal since the noisy label correction is limited. Conversely, we adopt a dynamic weight for α_t. Since the network parameters are initialized randomly, and the predictions are likely to be incorrect, it is not a practical idea to set α_t to be too large at the beginning of training process. Also, it should not be set too small in an advanced stage of the training; otherwise, there will be too many adverse effects from noisy labels. Thus, we set α_t as the exponentially increasing function of training iterations, formulated as:

$$\alpha_t = \alpha_T \cdot (t/T)^{\lambda}, \tag{9}$$

where α_T represents the confidence weight at the final iteration T, t denotes the number of iterations of the current training, and λ is the exponent that controls the rate of increase. With this confidence policy, α_t would dynamically increase from 0.0 to α_T as iterations increase. The basic assumption is that predictions become more and more accurate during training. Thus the loss function should accordingly put more reliability on the predictions. Therefore, we refer to this method as label confidence training.

However, during the last few optimization processes, there is the risk that the network may simply predict all samples as belonging to one same class to minimize the loss. To avoid this issue, inspired by [23], [40], we further incorporate a label regularization term into the objective function for back-propagation, and the total loss L_{total} is written as:

$$L_{total} = L' + \beta\frac{1}{M}\sum_{j=0}^{M-1}\log\left(\frac{1}{M\bar{P}_j}\right), \tag{10}$$

where β is the regularization coefficient, and $\bar{P}_j = \frac{1}{B}\sum_{i=1}^{B} P_{i,j}$ is the mean softmax probability for class j. The label regularization term enables the classifier to allocate each sample with a probability of belonging to every class, thereby preventing all samples from being assigned to a single class.

B. Clean Label Probability Re-scaling

If the predicted label of a sample is the same as its ground-truth label, we can generally believe that this sample is correctly labeled since the label confidence training criterion prevents the network from fitting noisy labels. Then, Eq. (8) is equivalent to Eq. (6). Nonetheless, we emphasize these clean samples to utilize them to learn discriminative speaker embeddings. Intuitively, the posterior probability is larger for clean samples and smaller for noisy samples, while the loss function is a monotonically decreasing function of the posterior probability. Therefore, we increase the contributions of clean samples to the parameter optimization by re-scaling its probability in the loss function. Specifically, the probability for clean samples is reduced to

$$P'_{i,y_i} = \frac{e^{s\left(\cos\left(\theta_{y_i,i}\right)-m\right)}}{e^{s\left(\cos\left(\theta_{y_i,i}\right)-m\right)} + \sum_{j\neq y_i}^{M} g\left(u\right) e^{s\left(\cos\left(\theta_{j,i}\right)\right)}}, \tag{11}$$

where $g\left(u\right)$ is a re-scaling function, and is formulated as:

$$g\left(u\right) = e^{u\cdot s\left(\cos\left(\theta_{j,i}\right)+1\right)}, \tag{12}$$

where $u \geq 0$, so that $g\left(u\right) \geq 1$, thereby $P'_{i,y_i} \leq P_{i,y_i}$ for clean samples. This approach follows [25]. By re-scaling, in the last training process, most of the updates would be attributed to clean examples, thereby weakening the incorrectly labeled samples.

C. Sub-center AM-Softmax

AM-Softmax and AAM-Softmax are two angular margin-based objective functions that are commonly used in deep speaker recognition. The x-vectors learned by such a function are angularly distributed and naturally match the back-end scoring based on cosine similarity [41]. Nevertheless, they also perform better than Softmax when adopting the PLDA back-end for scoring since they explicitly minimize the within-class covariance [9]. However, they are susceptible to label noise as the inter-class speakers contain incorrect samples. To address this problem, sub-center ArcFace has recently been proposed for face recognition [42], it relaxes the intra-class constraint of ArcFace [15]. While for speaker recognition, the AM-Softmax performs comparably to AAM-Softmax [43]–[45], and it converges faster [46]. So, in this work, we adopt the AM-Softmax as an objective function, and also manage to relax its intra-class constraint to further improve the robustness to label noise.

The concept of "sub-classes" has been employed in face recognition for some time. Research shows that it separates different patterns more clearly, thus improves recognition accuracy [47], [48]. Following the set in [42], we introduce sub-classes into AM-Softmax and refer to the improved loss function as sub-center AM-Softmax (SubAM-Softmax for short). Specifically, K sub-centers are introduced for each class to relax the intra-class constraint; this is carried out where there is one dominant sub-class, containing the majority of clean samples, and $(K-1)$ non-dominant sub-classes contain noisy samples. To form K sub-centers, the dimension of matrix $\mathbf{W}$ in SubAM-Softmax is extended to $\mathbb{R}^{M\times K\times d}$, and then the similarity score is formulated as $\cos\left(\theta_{j,i}\right) = \max_k\left(\mathbf{W}_{j_k}^{\mathrm{T}}\mathbf{x}_i\right), k \in \{1,\cdots,K\}$, where $\max_k$ denotes a max-pooling step. The sub-classes are able to capture the complex distribution for the training data and separate noisy samples from clean samples. Therefore, sub-classes enable the loss function to be more robust to label noise [42].

D. Discussions

1) Handling Hard Samples: In this paper, we refer to hard samples as clean samples that require more time for the network to learn. In the proposed method, we re-label samples according to the network's predictions. Although the predictions are more accurate than the original noisy labels, they may also misclassify some hard samples as noisy. To reduce this mis-labeling, we keep the original labels in the loss function as a part of supervised training. Moreover, in the well-designed confidence policy, as shown in Eq. (9), α_T controls the maximum confidence degree, and λ controls the confidence rate for the network. Thus the two parameters can be empirically set to trade-off ground-truth and pseudo labels.

2) How This Method Works: The framework leverages both the generalization ability of a network and speech single features to learn from label noise. Specifically, for generalization, a network first learns the patterns of a dataset from the correct samples and maintains highly generative performances at the beginning of training processes. This learning characteristic enables a learned model to generate correct patterns for noisy samples and classify them into correct classes before memorizing them. Therefore, the proposed method prevents a network from fitting incorrect samples by incorporating predicted labels to correct noisy samples on the fly. While for speech samples, an utterance is split into multiple chunks during training, and the network learns chunk-level speaker representations. Since speech is a non-stationary time series single, there are contrasts across chunks [49], so it is almost impossible for the network to predict all the chunks in an utterance to a mislabeled class. However, by adopting label-confidence learning, the majority of chunks for an utterance are more likely to be correctly predicted.

IV. Back-end PLDA estimation with noisy labels

To address the problem of label errors in the back-end, a method for Bayesian estimation of PLDA with noisy labels is proposed in [17]. Though the theoretical analysis of NL-PLDA has been covered extensively in [17], it does not illustrate a concrete implementation of the algorithm. In this section, for further practical contributions, we show a detailed algorithmic presentation of NL-PLDA and its utilization of automatic filtering to weed out high-confidence noisy samples.

Algorithm 1 Bayesian estimation of PLDA with noisy labels

```
Input  : Traing set $\mathcal{X} = \{\mathbf{x}_1,\ldots,\mathbf{x}_N\}$, and corresponding labels $\mathcal{L} = \{l_1,\ldots,l_N\}$, where contains $M$ individuals and $N$ samples.
Initialize $\mu$, $\mathbf{\Sigma}_b$, $\mathbf{\Sigma}_w$; $\epsilon \leftarrow 0$; $z_{n,m} \leftarrow 1$ if $l_n = m$ else $0$;
repeat
    E-step:
        Set $\mathbf{r}_y \leftarrow 0$; $\left\{\mathbf{R}_y^o, \mathbf{R}_y, \mathbf{R}_{xy}\right\} \leftarrow \mathbf{0}$;
        Update $N_e$ based on Eq. (14);
        Update $P\left(l_n|z_{n,m}=1,\epsilon\right)$ based on Eq. (27);
        for $m = 1$ to $M$ do
            Update $N_m$, $\mathbf{r}_{x,m}$ based on Eq. (15), (16);
            Update $\mathbf{\Phi}_m$ based on Eq. (20);
            Update $\langle\mathbf{y}_m\rangle$, $\langle\mathbf{y}_m\mathbf{y}_m^T\rangle$ based on Eq. (18), (19);
        Update $\mathbf{r}_y$, $\mathbf{R}_y^o$, $\mathbf{R}_y$, and $\mathbf{R}_{xy}$ based on Eq. (21)–(24);
    M-step:
        Update $z_{n,m}$ based on Eq. (25);
        Update $\epsilon$ based on Eq. (28);
        Update $\mu$, $\mathbf{\Sigma}_b$, and $\mathbf{\Sigma}_w$ based on Eq. (29)–(31);
until Convergence;
Output: PLDA model parameters $\{\mu, \mathbf{\Sigma}_b, \mathbf{\Sigma}_w\}$, and $\epsilon$
```

To combat label noise, NL-PLDA treats true labels as multinomial random variables and estimates a model's parameters based on maximum-likelihood estimation in the context of Variational Bayes. Specifically, we suppose that the training samples and corresponding labels denoted as $\mathcal{X} = \{\mathbf{x}_1,\ldots,\mathbf{x}_N\}$, $\mathcal{L} = \{l_1,\ldots,l_N\}$, respectively, where N is the sample size from M individuals. However, since the data is noisy, $\mathcal{L}$ is not the correct identity. To tackle this problem, the true label for each sample $\mathbf{x}_n$ is modeled as a latent identity $\mathbf{z}_n \in \mathbb{R}^M$, and we let $\mathcal{Z} = \{\mathbf{z}_1,\ldots,\mathbf{z}_N\}$ denote the set of true identities corresponding to $\mathcal{X}$. Each element $z_{n,m}$ ($\sum_{m=1}^{M} z_{n,m} \equiv 1$) in $\mathcal{Z}$ indicates the confidence (probability) that $\mathbf{x}_n$ belongs to individual m.

The EM algorithm for the NL-PLDA is summarized in short form in Algorithm 1, and the details are available in Appendix A. In the E-step, it estimates both the posterior of true identity and the individual feature distribution simultaneously, whereas the M-step updates the label error rate ϵ, true latent identities $\mathcal{Z}$, and the parameters of NL-PLDA, respectively.

Moreover, since $\mathcal{Z}$ explicitly models the latent identity distribution, it can be utilized to filter out high-confidence noisy labels. Before training, no a priori information about the error rate is available; therefore, it is assumed that there are no label errors ($\epsilon = 0$). So, the initialization for $\mathcal{Z}$ can be shown as the left matrix of Eq. (13), where $z_{n,m}$ is initialized as $z_{n,l_n} = 1$, implying that $\mathbf{x}_n$ belongs to the original corresponding individual l_n. During the EM iteration steps, $\mathcal{Z}$ is determined by the maximum posterior estimation.

We assume that the final updated $\mathcal{Z}$ is shown as the right matrix of Eq. (13). When the value of z_{n,l_n} becomes relatively small, this indicates that l_n might have a high probability of mislabeling. So, a threshold (e. g., $z_{n,l_n} \leq 0.1$) is empirically set to filter out these samples.

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}, \begin{bmatrix} 0.9 & 0 & 0 & \cdots & 0.01 \\ \boxed{0.1} & 0.7 & 0.1 & \cdots & 0 \\ 0 & 0.8 & 0 & \cdots & 0.1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0.4 & 0.2 & 0.2 & \cdots & \boxed{0.05} \end{bmatrix} \tag{13}$$

V. Experiments

A. Datasets

1) Training Datasets: The training datasets are prepared following the SRE16 Kaldi recipe[1], including the Switchboard Phase2-3, Cellular1-2 (SWBD), and the NIST SRE04-10 datasets. After filtering out non-speech frames by energy-based voice activity detection (VAD), the recordings shorter than four seconds and the speakers with less than eight recordings are discarded. Finally, the SWBD portion contains 18,407 English recordings from 1,318 speakers, and the SRE04-10 includes 2,682 speakers with 48,022 utterances. Most of these recordings are in English, while some are in Chinese, Russian, Arabic, etc. We use the two pooled datasets to train the front-end extractions, while only using the SRE04-10 portion for the LDA and PLDA back-end training.

2) Evaluation Datasets: The evaluation datasets consist of NIST SRE 2016 (SRE16) and NIST SRE 2018 CMN2 (SRE18). Specifically, SRE16 is composed of Cantonese and Tagalog telephone conversations; the Cantonese dataset contains 965,395 trials and the Tagalog dataset contains 1,021,332 trials. For SRE18, the CMN2 collection is mainly spoken in Tunisian Arabic, and contains 108,095 trials in the development set and 2,063,007 trials in the evaluation set.

B. Experimental Settings

1) Data Preparation: In this section, we describe the way we prepare the simulated closed-set label noise. In our experiments, we first assume that the original training datasets are clean (without error labels), denoted as $\epsilon = 0\%$. Although we later confirmed that there are indeed a few mislabeled recordings in the training datasets. To better monitor the network prediction accuracy on a clean data set, we divide the training datasets into a *training set* and a *validation set*. Specifically, the validation set is composed of one utterance randomly selected from each speaker, and the remaining recordings are used to compose the training set. It is noteworthy that there is no overlap between the two subsets. To simulate different label error rates, we perform label disruption on the SWBD and NIST SRE04-10 training sets, respectively. Specifically, we randomly select $\epsilon \in \{5\%, 10\%, 20\%, 30\%, 50\%\}$ percent of each speaker's utterances and then randomly relabel them as other speaker identities presented in the training set. For instance, if $\epsilon = 50\%$, then half of each speaker's utterances are randomly relabeled as belonging to other speakers in the training set; in this way, we can obtain a 50% label error rate in the SWBD and NIST SRE04-10 training sets, respectively. Meanwhile, the validation set is kept clean and is used to monitor the actual prediction accuracy. It is noted that the validation set does not involve network parameter updates.

All of the raw audio files are converted to 40-dimensional Mel-frequency Cepstral Coefficients (MFCCs) with a 25 ms window and a 10 ms frame shift. Cepstral Mean Normalization over a three-second sliding window is applied to the MFCCs. After removing non-speech frames by VAD, the average duration of utterances in SWBD is 171 seconds and 160 seconds in SRE04-10. For the x-vector based front-end training, speech utterances are uniformly cut into chunks without overlaps, where the chunk length is set to 400 ms. These chunks are randomly formed into mini-batches as the network inputs.

TABLE I
Configurations of The X-vector Based Front-ends

	Dynamic Confidence	Fixed Confidence	Label Regularization	Probability Re-scaling	SubAM-Softmax
Front1	✓		✓		
Front2		✓			
Front3	✓				
Front4	✓		✓	✓	
Front5	✓		✓		✓
Front6	✓		✓	✓	✓

2) Front-end Configurations: We conduct experiments using x-vector based front-ends, which are implemented in ASV-Subtools [50]. For the x-vector baseline system, we apply the extended-TDNN (E-TDNN) structure [12] with AM-Softmax loss to train a 512-dimensional x-vector extractor. The detailed implementations of the E-TDNN source codes are released on GitHub[2]. Besides the x-vector baseline, we also train six other x-vector based front-ends with different configurations as shown in Table I. These front-ends are roughly trained by adding more components progressively, which enables us to observe the contributions of individual components. It is noteworthy that the Front2 is set to adopt fixed confidence weights, which is characteristic of the Bootstrap method [22]. All of these networks are trained on the GeForce RTX 2080 Ti GPUs with a mini-batch size of 256. AdamW is chosen as the optimizer, the weight decay is set to $1e-1$, and the learning rate is initially set to $1e-3$ and gradually reduced to $1e-6$. The networks are trained with 21 epochs, in which there are about two hundred thousand (200K) iterations in total.

3) Back-end Training & Evaluation: Once trained, we choose the epoch that gives the highest validation set accuracy as the final model. The activations for the model's penultimate fully connected layer are extracted as speaker embeddings (x-vectors). For the evaluation process, we first project the x-vectors to a lower 256-dimensional space using LDA and then adopt centering and length normalization. We trained LDA, PLDA, and NL-PLDA only on the SRE04-10 dataset.

Since the evaluation datasets are non-English, and PLDA is mainly trained on English utterances, domain mismatch

[1]https://github.com/kaldi-asr/kaldi/tree/master/egs/sre16/v1

[2]https://github.com/Snowdar/asv-subtools/tree/master/pytorch/model

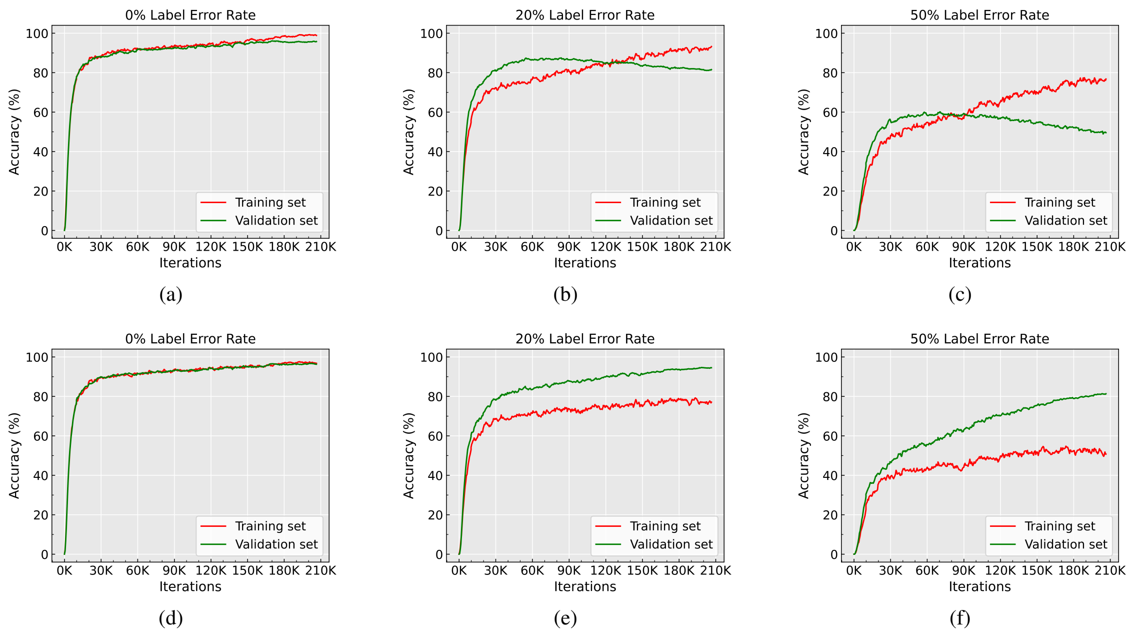


Fig. 2. Comparisons of the x-vector (Row 1) and Front1 (Row 2) accuracy under 0%, 20%, and 50% label error rates.

is possible between the training and test. To handle this problem, we adopt the unsupervised PLDA adaptation method implemented in Kaldi. For the SRE16 evaluation, the SRE16 unlabeled development set is used for PLDA adaptation, while for the SRE18 CMN2 evaluation, the PLDA adaptation is trained on the SRE18 development set. The PLDA scoring results are reported in terms of Equal Error Rate (EER) and minimum detection cost function (minDCF) with p-target set to 0.01. Experiments based on closed-set label noise are shown in Section V-C–V-E, and open-set label noise scenarios are shown in Section V-F.

C. Results for The X-vector and Label Confidence Training

In this section, we compare the effects of label noise on the x-vector baseline and the Front1–3 extractors based on label confidence training. The exponent value for label confidence training is set as $\lambda = 2.0$, and α_T is set to 1.0.

1) Training Processes & Results: Before presenting the final results, we first show an explicit comparison between the x-vector baseline and the Front1 prediction accuracy on the training set and evaluation set, respectively. Note that the prediction accuracy is computed as the fraction of *chunk samples* in the training set or validation set that are classified correctly with respect to the corresponding labeled classes. As depicted in Fig. 2, the representative training evolutions with label error rates of $\epsilon \in \{0\%, 20\%, 50\%\}$ are presented in order from left to right. Compared with Fig. 2(a), (b), and (c), one can clearly observe that the clean dataset converges faster than the mislabeled dataset, indicating that the network takes longer to learn mislabeled samples. It also shows that the network learns reasonable representations in the first few iterations, as shown by the fact that the validation set gives higher prediction accuracy than the training set with label error rates of 20% and 50%. Unfortunately, the increasing number of training iterations does not further motivate the model to learn as expected; instead, it leads to the model overfitting incorrect samples, which subsequently degrades the prediction accuracy on the validation set. On the contrary, the label confidence training scheme suffers from fewer adverse effects due to mislabeled samples. As shown in Fig. 2(e) and (f), the model produces higher validation set accuracy during the entire training evolution. Besides, it is quite remarkable that a final validation accuracy rate of around 85% can be achieved even when the label error rate increases to 50%. We would like to emphasize that the final training accuracy of the models trained by this approach is close to the expected true label error rate of the dataset, indicating that this approach can separate erroneous samples from correct samples within the whole training dataset.

The results of the x-vector baseline and Front1 extractor used in conjunction with PLDA and NL-PLDA are summarized in Table II. Let us first focus on the PLDA results, as shown in the left part of Table II. One can clearly observe that the x-vector's performance breaks down rapidly as the label error rate increases, while the Front1 significantly outperforms the baseline in situations with label noise.

Results for NL-PLDA are shown on the right side of Table II. In general, these results are consistent with the trend of PLDA, while NL-PLDA achieves better performance in terms of EER and minDCF under most situations. This implies NL-PLDA is capable of handling label noise. However, the capacity of NL-PLDA degrades in the presence of strong label noise, especially for the x-vector. One explanation is that an insufficient number of correct labels poses a challenge to the NL-PLDA training. While another important reason is that the speaker embeddings learned by the x-vector contain less discriminative information, which confuses the NL-PLDA label noise estimation. This implies that a front-end that is

TABLE II
RESULTS OF FRONT1 USED IN CONJUNCTION WITH PLDA AND NL-PLDA WITH DIFFERENT LABEL ERROR RATES (ϵ)

		PLDA								**NL-PLDA**							
	ϵ(%)	SRE16 Cantonese		SRE16 Tagalog		SRE18 CMN2 Dev		SRE18 CMN2 Eval		SRE16 Cantonese		SRE16 Tagalog		SRE18 CMN2 Dev		SRE18 CMN2 Eval	
		EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF
x-vector	0	4.27	0.396	11.83	0.815	8.86	0.578	10.83	0.626	4.28	0.396	11.80	0.812	8.74	0.576	10.78	0.625
	5	6.41	0.510	14.15	0.830	11.84	0.648	13.66	0.696	5.72	0.486	13.49	0.819	11.20	0.629	13.05	0.682
	10	7.20	0.543	15.11	0.844	13.12	0.692	15.05	0.716	6.10	0.513	14.07	0.823	11.69	0.656	14.20	0.698
	20	9.57	0.620	17.23	0.868	14.36	0.761	17.14	0.768	7.83	0.585	15.75	0.846	12.16	0.714	15.79	0.747
	30	10.92	0.658	18.33	0.894	16.82	0.795	18.45	0.800	8.88	0.623	16.87	0.866	14.78	0.764	17.13	0.785
	50	13.70	0.783	20.65	0.966	20.65	0.884	21.41	0.851	11.99	0.729	19.70	0.925	18.94	0.860	20.53	0.838
Front1	0	4.29	0.402	11.19	0.789	8.88	0.583	10.76	0.623	4.25	0.400	11.13	0.754	8.79	0.608	10.71	0.632
	5	4.92	0.448	12.28	0.813	9.40	0.629	11.52	0.655	4.28	0.417	11.45	0.790	8.66	0.589	10.81	0.632
	10	5.60	0.490	13.08	0.817	9.85	0.648	12.07	0.674	4.43	0.440	11.74	0.786	8.57	0.595	11.00	0.641
	20	7.21	0.553	14.55	0.849	12.34	0.667	13.65	0.710	5.06	0.483	12.65	0.810	9.78	0.615	12.00	0.669
	30	8.25	0.611	15.11	0.874	12.55	0.686	14.73	0.742	6.03	0.544	13.09	0.826	10.11	0.643	12.86	0.702
	50	10.90	0.739	18.75	0.940	16.00	0.838	17.75	0.810	8.50	0.644	16.01	0.887	12.85	0.770	15.89	0.780

TABLE III
PERFORMANCE OF FRONT2 IN CONJUNCTION WITH PLDA

		PLDA							
	ϵ(%)	SRE16 Cantonese		SRE16 Tagalog		SRE18 CMN2 Dev		SRE18 CMN2 Eval	
		EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF
Front2	0	4.33	0.395	11.46	0.797	8.90	0.574	10.88	0.631
	5	5.38	0.485	12.67	0.836	9.50	0.649	11.82	0.665
	10	6.16	0.542	13.76	0.845	11.51	0.688	13.13	0.702
	20	8.30	0.599	15.85	0.892	12.99	0.742	15.10	0.748
	30	9.58	0.666	16.85	0.915	14.73	0.782	16.27	0.792
	50	12.76	0.769	19.67	0.944	18.07	0.870	19.51	0.832

TABLE IV
PERFORMANCE OF FRONT3 IN CONJUNCTION WITH PLDA

		PLDA							
	ϵ(%)	SRE16 Cantonese		SRE16 Tagalog		SRE18 CMN2 Dev		SRE18 CMN2 Eval	
		EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF
Front3	0	4.07	0.387	11.18	0.803	8.43	0.573	10.76	0.622
	5	5.03	0.455	12.41	0.824	9.72	0.676	11.55	0.660
	10	5.65	0.507	13.19	0.831	10.59	0.666	12.71	0.671
	20	7.26	0.552	14.51	0.866	12.42	0.709	13.84	0.712
	30	8.57	0.623	15.97	0.892	12.98	0.716	14.95	0.741
	50	11.33	0.762	19.04	0.941	16.64	0.818	17.89	0.809

robust to label noise would facilitate the whole system's performance improvement. Besides, it is worth pointing out that label confidence training does not cause performance degradation when a training dataset is clean. As shown, the Front1 extractor achieves almost the same results as the x-vector on SRE16 Cantonese and SRE18 CMN2 test sets, and even better results on the SRE16 Tagalog test set.

Another interesting observation is that the EERs of NL-PLDA on the SRE16 Tagalog and SRE18 CMN2 sets are slightly lower than those of PLDA. Since the only difference between the training of NL-PLDA and PLDA lies in how the speaker labels are treated, this phenomenon motivates us to investigate whether there are noisy samples in the original training datasets, which is explored in Section VI-B.

2) Comparisons with Fixed Confidence: In this section, we consider the comparisons between dynamic and fixed confidence weights. We empirically fix the confidence weight in Front2 as $\alpha_t = 0.3$ during the whole training process. Since the results of NL-PLDA are consistent with the trend of PLDA, only the PLDA results are reported in Table III. Compared with the results of Front1 (in Table II) and Front2, one can clearly observe that the Front1 outperforms Front2 in most situations, indicating that the gradually increasing confidence weights are more effective in reducing the impact of label noise.

3) Effects of Label Regularization: To examine the effects of label regularization term, the Front3 without label regularization, is trained as a comparison to Front1. The results of Front3 with a PLDA back-end are shown in Table IV. One can observe that Front3 results lower EERs than Front1 on clean label training dataset, indicating that adding label regularization causes slight performance degradation when training label is clean. However, incorporating label regularization achieves consistent improvements on the noisy dataset. One explanation is that label regularization results in redundant information within the clean training dataset's loss values, while this information is useful for learning with label noise.

D. Effectiveness of Re-scaling and SubAM-Softmax

We further examine the effectiveness of the re-scaling strategy and SubAM-Softmax for handling label noise. Table V provides the results of adding a re-scaling, SubAM-Softmax, and the two combined front-ends, respectively. Compared to the results of Front1 in Table II, one can observe from the first part of Table V that the re-scaling helps boost the performance. These favorable results indicate that focusing more on clean labels is helpful when training with noisy labels. From the second part of Table V, one can observe that the SubAM-Softmax is more robust than the standard AM-Softmax in conditions with massive noise. Besides, the performance of SubAM-Softmax is even slightly better than AM-Softmax when trained on a clean dataset.

Finally, we obtain the best results on the Front6 extractor, which adopts label confidence training and combines the re-scaling and Sub-AM softmax. The results are shown in the third part of Table V. The Front6 shows its superior performance over the baseline and effectively enhances the robustness of the speaker recognition system for dealing with noisy labels. More specifically, compared to the x-vector baseline in Table II, when scoring with PLDA, the Front6 extractor trained with a 50% label error rate performs comparably compared to the x-vector trained with a 20% label error rate. While scoring with NL-PLDA, the performance of Front6 even approaches the x-vector baseline trained with a 5% label error rate. To better illustrate the performance comparisons with different front-ends and back-ends, the DET curves on

TABLE V
PERFORMANCE COMPARISONS OF FRONT4–6 WITH DIFFERENT LABEL ERROR RATES (ϵ)

		PLDA								NL-PLDA							
	ϵ(%)	SRE16 Cantonese		SRE16 Tagalog		SRE18 CMN2 Dev		SRE18 CMN2 Eval		SRE16 Cantonese		SRE16 Tagalog		SRE18 CMN2 Dev		SRE18 CMN2 Eval	
		EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF
Front4	0	4.33	0.392	11.52	0.801	8.46	0.592	11.00	0.627	4.30	0.392	11.44	0.799	8.30	0.590	10.96	0.625
	5	5.04	0.435	12.40	0.827	9.37	0.617	11.56	0.650	4.33	0.394	11.46	0.802	8.40	0.583	10.79	0.626
	10	5.42	0.451	13.11	0.824	9.73	0.684	11.91	0.663	4.16	0.394	11.66	0.786	8.20	0.621	10.63	0.626
	20	6.62	0.510	14.21	0.881	11.55	0.678	12.62	0.695	4.48	0.425	11.85	0.832	8.81	0.610	10.85	0.647
	30	7.66	0.574	15.22	0.900	11.89	0.733	13.65	0.715	4.85	0.465	12.37	0.838	9.04	0.656	11.34	0.658
	50	9.32	0.679	17.36	0.950	14.43	0.771	15.30	0.770	5.64	0.518	13.11	0.863	10.20	0.659	11.73	0.688
Front5	0	4.06	0.390	11.59	0.811	8.31	0.596	10.60	0.616	4.02	0.388	11.57	0.810	8.21	0.596	10.58	0.616
	5	4.89	0.444	12.75	0.826	9.23	0.645	11.40	0.644	4.11	0.406	11.83	0.803	8.52	0.614	10.64	0.620
	10	5.73	0.485	13.09	0.845	10.00	0.649	11.86	0.664	4.36	0.415	11.45	0.810	8.53	0.591	10.51	0.623
	20	6.17	0.501	14.06	0.870	11.64	0.665	12.48	0.682	4.21	0.416	11.64	0.817	8.94	0.590	10.49	0.631
	30	7.51	0.582	15.01	0.900	11.90	0.697	13.42	0.716	4.92	0.469	12.00	0.844	8.91	0.636	10.99	0.657
	50	9.64	0.678	17.92	0.953	14.64	0.780	15.38	0.775	6.05	0.520	13.67	0.873	10.34	0.680	12.09	0.695
Front6	0	4.01	0.395	11.57	0.804	8.44	0.599	10.59	0.616	3.99	0.393	11.53	0.803	8.31	0.594	10.55	0.616
	5	4.87	0.431	12.64	0.837	9.06	0.631	11.22	0.641	4.08	0.391	11.71	0.813	8.23	0.597	10.45	0.617
	10	5.49	0.470	13.43	0.838	9.69	0.670	11.86	0.658	4.20	0.408	11.90	0.801	8.53	0.624	10.55	0.621
	20	6.57	0.525	14.02	0.865	10.82	0.693	12.50	0.683	4.38	0.419	11.55	0.816	8.58	0.601	10.54	0.627
	30	6.84	0.532	14.67	0.880	12.31	0.686	12.94	0.699	4.36	0.428	11.70	0.814	8.90	0.598	10.54	0.632
	50	8.85	0.673	16.99	0.951	14.39	0.764	14.96	0.757	5.01	0.478	12.17	0.856	9.00	0.644	11.87	0.665

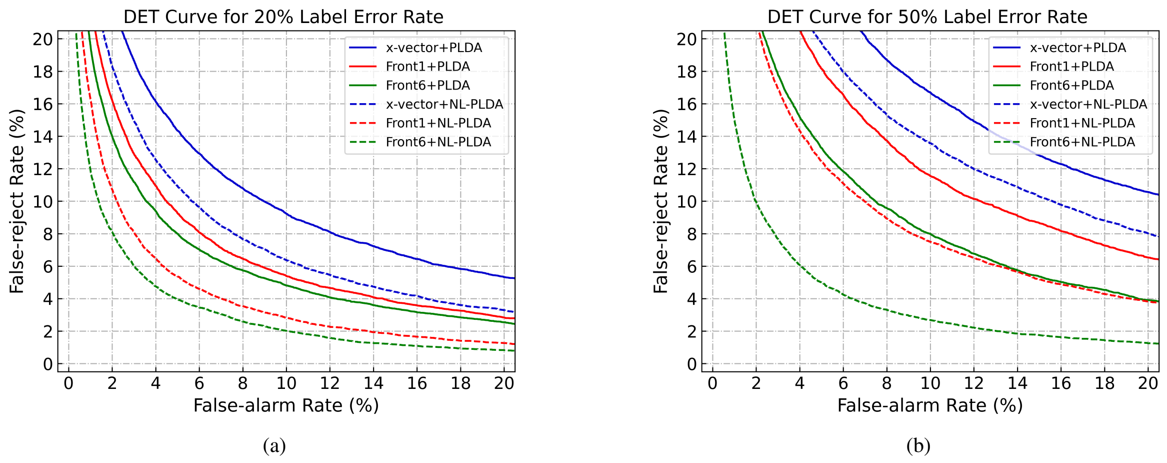


Fig. 3. DET curves on SRE16 Cantonese for different front-ends and back-ends with 20% and 50% label error rates.

the SRE16 Cantonese evaluation set are presented in Fig. 3. All the systems selected are trained with 20% and 50% label error rates. From Fig. 3, one can make the following observations: 1) Substantial improvements are obtained by label confidence training. 2) Re-scaling and SubAM-Softmax have complementary properties, and the systems yield further improvements when used in tandem. 3) NL-PLDA always performs better than PLDA when training labels are noisy. 4) The superiority of NL-PLDA benefits from a more robust front-end, especially in the presence of strong label error rates.

E. Effects of LDA Configurations

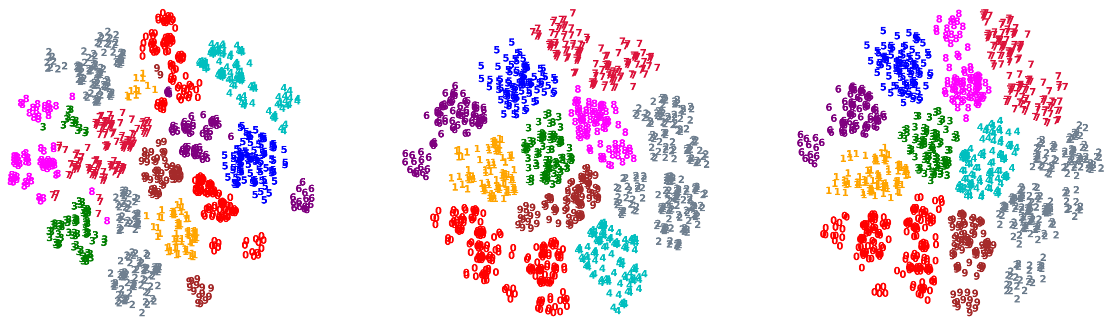
Since LDA training also requires speaker labels, in this set of experiments, we investigate the effects of different LDA configurations on PLDA and NL-PLDA. Three distinct back-end configurations are compared concretely: without LDA, LDA trained on noisy labels, and LDA trained on clean labels, respectively. The x-vector baseline is used as the front-end, and the experimental results are shown in Table VI. For convenient comparisons, we re-present the results of x-vector baseline (in the middle part of Table VI). From Table VI, it is clear that the back-end without LDA yields the worst results. Moreover, NL-PLDA loses its ability to combat label noise and even achieves worse results than PLDA. However, the performance of NL-PLDA can be improved by using LDA projections, even if LDA is trained on incorrect labels. LDA trained on clean labels achieves optimal results, but the performance gain is not very large, especially when the label error rate is small. It seems that LDA is robust to noisy labels.

To further observe the effects of LDA, we visualize the speaker embeddings by plotting the t-SNE embeddings. The embeddings from ten distinct clusters without distinct LDA configurations are shown in Fig. 4. Each embedding is represented by its corresponding true label. From Fig 4 (a), embeddings without LDA projections are more isolated within their classes, and they do not have clear separated boundaries between classes. While in Fig 4 (b) and (c), the embeddings with LDA are more effectively separated into clusters. The embeddings presented in Fig 4 (b) are very similar to those in Fig 4 (c), demonstrating that LDA shows robustness to label

TABLE VI
PERFORMANCE OF THE X-VECTOR BASELINE WITH DIFFERENT LDA CONFIGURATIONS

		PLDA								**NL-PLDA**							
	ϵ(%)	SRE16 Cantonese		SRE16 Tagalog		SRE18 CMN2 Dev		SRE18 CMN2 Eval		SRE16 Cantonese		SRE16 Tagalog		SRE18 CMN2 Dev		SRE18 CMN2 Eval	
		EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF
w/o LDA	0	5.41	0.429	14.33	0.865	10.34	0.644	16.11	0.730	6.22	0.518	15.05	0.899	11.38	0.685	17.10	0.748
	5	8.43	0.548	17.21	0.915	13.21	0.747	19.48	0.819	8.78	0.598	18.32	0.919	14.15	0.781	20.40	0.831
	10	9.13	0.586	17.52	0.910	14.74	0.784	20.27	0.824	9.75	0.616	18.76	0.959	15.39	0.810	21.30	0.838
	20	10.82	0.648	18.66	0.940	15.84	0.783	21.18	0.828	11.05	0.671	19.28	0.962	16.12	0.797	22.37	0.845
	30	11.86	0.678	19.46	0.938	17.57	0.807	21.75	0.843	11.99	0.725	19.61	0.978	17.75	0.833	22.14	0.860
	50	14.00	0.774	21.05	0.970	21.93	0.878	24.68	0.868	14.24	0.795	21.23	0.998	21.70	0.884	24.48	0.884
LDA w/ NL	0	4.27	0.396	11.83	0.815	8.86	0.578	10.83	0.626	4.28	0.396	11.80	0.812	8.74	0.576	10.78	0.625
	5	6.41	0.510	14.15	0.830	11.84	0.648	13.66	0.696	5.72	0.486	13.49	0.819	11.20	0.629	13.05	0.682
	10	7.20	0.543	15.11	0.844	13.12	0.692	15.05	0.716	6.10	0.513	14.07	0.823	11.69	0.656	14.20	0.698
	20	9.57	0.620	17.23	0.868	14.36	0.761	17.14	0.768	7.83	0.585	15.75	0.846	12.16	0.714	15.79	0.747
	30	10.92	0.658	18.33	0.894	16.82	0.795	18.45	0.800	8.88	0.623	16.87	0.866	14.78	0.764	17.13	0.785
	50	13.70	0.783	20.65	0.966	20.65	0.884	21.41	0.851	11.99	0.729	19.70	0.925	18.94	0.860	20.53	0.838
LDA w/ CL	0	4.27	0.396	11.83	0.815	8.86	0.578	10.83	0.626	4.28	0.396	11.80	0.812	8.74	0.576	10.78	0.625
	5	6.35	0.504	14.13	0.825	11.72	0.641	13.64	0.693	5.61	0.479	13.48	0.813	11.15	0.622	13.03	0.679
	10	7.11	0.542	14.96	0.832	12.72	0.695	15.04	0.713	5.90	0.504	13.81	0.811	11.49	0.662	14.15	0.694
	20	9.25	0.611	16.75	0.855	13.86	0.740	16.72	0.759	7.39	0.577	15.26	0.828	11.79	0.693	15.46	0.740
	30	10.17	0.646	17.85	0.893	16.26	0.764	17.93	0.788	8.07	0.605	16.47	0.862	14.04	0.745	16.60	0.770
	50	12.84	0.776	19.81	0.950	20.05	0.872	20.64	0.839	11.09	0.711	18.75	0.898	18.20	0.850	19.59	0.824

where "NL" denotes noisy labels and "CL" denotes clean labels.



(a) (b) (c)

Fig. 4. t-SNE visualization of speaker embeddings for w/o LDA, LDA trained on noisy labels, and LDA trained on clean labels (in order from left to right, respectively). The speaker embeddings are extracted by the x-vector baseline trained with 50% label error rate. Each number or color represents a class.

noise. It also suggests that LDA removes non-discriminant dimensions, which are potentially caused by the label noise, and hence facilitates PLDA and NL-PLDA training.

F. Validations on Open-set Label Noise

In this section, we conduct experiments to validate the performance of the proposed front-end and NL-PLDA back-end on an open-set noisy datasets. The open-set noisy datasets are simulated by randomly selecting $p \in \{5\%, 10\%, 20\%, 30\%, 50\%\}$ training utterances per speaker in the original SWBD and SRE04-10 datasets and replacing them with utterances randomly selected from the concatenated Vox-Celeb2 datasets (subsegments belonging to the same video are concatenated together to form a unique utterance, and then it is down-sampled to 8 kHz). It is noteworthy that the labels and the number of utterances per speaker remain unchanged. These open-set noisy datasets are used to train an x-vector base-line and Front6 extractors, respectively. Experimental results for four validation sets using PLDA and NL-PLDA scoring are shown in Table VII. As shown, Front6 achieves better performance compared to the x-vector baseline; this indicates that our method robustly trains front-ends from open-set noisy datasets. Although these noisy samples cannot be localized to the corresponding correct labels in the training set, our method reduces their detrimental effects by clustering them separately into similar classes in the training set. Compared with the x-vector baseline in Table II, we observe that the impact of closed-set label noise is more harmful than that of open-set for the front-end. However, this phenomenon is opposite for the NL-PLDA back-end, as shown by the comparison of Front6 in Table V. NL-PLDA results in higher EER and minDCF under the same label error rates for the open-set label noise. Nonetheless, it still outperforms PLDA in all scenarios.

VI. APPLICATIONS

A. Label Correction on Synthetic Datasets

As shown above, when a model is trained to be robust to noisy labels, the prediction accuracy of the model is greater than the label error rate itself. In addition, this prediction accuracy is based on chunk-level samples, and since an utterance is composed of multiple chunks, it can be expected

TABLE VII
PERFORMANCE COMPARISONS OF THE X-VECTOR BASELINE AND FRONT6 WITH OPEN-SET LABEL ERROR RATES (p)

		PLDA								**NL-PLDA**							
	p(%)	SRE16 Cantonese		SRE16 Tagalog		SRE18 CMN2 Dev		SRE18 CMN2 Eval		SRE16 Cantonese		SRE16 Tagalog		SRE18 CMN2 Dev		SRE18 CMN2 Eval	
		EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF
x-vector	5	4.91	0.440	12.27	0.833	9.16	0.632	11.24	0.655	4.53	0.424	11.89	0.824	8.67	0.617	10.82	0.644
	10	5.67	0.476	12.51	0.834	10.11	0.675	11.92	0.673	5.10	0.453	11.98	0.821	9.31	0.654	11.30	0.658
	20	6.01	0.488	13.41	0.820	10.51	0.676	12.23	0.685	5.30	0.462	12.48	0.802	9.59	0.659	11.49	0.675
	30	7.06	0.539	14.60	0.861	11.25	0.698	13.27	0.707	6.61	0.523	13.36	0.836	10.41	0.705	12.44	0.704
	50	8.75	0.637	16.49	0.914	13.30	0.719	14.52	0.730	7.86	0.569	14.07	0.852	12.15	0.699	13.70	0.718
Front6	5	4.66	0.425	11.74	0.795	8.54	0.599	10.67	0.638	4.29	0.408	11.27	0.786	7.93	0.576	10.24	0.626
	10	5.39	0.454	12.16	0.817	9.02	0.627	11.19	0.646	4.72	0.437	11.52	0.801	8.39	0.616	10.61	0.632
	20	5.95	0.493	13.08	0.825	9.81	0.657	12.03	0.671	5.13	0.474	12.01	0.811	8.70	0.654	11.31	0.660
	30	6.80	0.535	14.02	0.864	11.10	0.680	12.67	0.687	6.07	0.524	13.03	0.850	9.92	0.680	11.91	0.682
	50	8.08	0.594	15.95	0.894	11.71	0.756	14.14	0.746	7.32	0.561	14.94	0.859	10.34	0.742	13.03	0.726

TABLE VIII
PERFORMANCE OF FRONT6 TRAINED ON LABEL CORRECTED DATASETS

		PLDA								**NL-PLDA**							
	ϵ(%)	SRE16 Cantonese		SRE16 Tagalog		SRE18 CMN2 Dev		SRE18 CMN2 Eval		SRE16 Cantonese		SRE16 Tagalog		SRE18 CMN2 Dev		SRE18 CMN2 Eval	
		EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF
Front6	5	4.22	0.401	11.70	0.782	8.77	0.599	10.97	0.627	4.22	0.401	11.69	0.782	8.77	0.598	10.97	0.627
	10	4.32	0.411	11.59	0.814	8.42	0.605	10.88	0.624	4.30	0.410	11.59	0.814	8.35	0.603	10.88	0.623
	20	4.10	0.404	11.32	0.795	8.34	0.602	11.17	0.623	4.10	0.403	11.33	0.794	8.28	0.600	11.12	0.622
	30	4.47	0.429	11.80	0.814	8.49	0.615	11.26	0.637	4.34	0.424	11.68	0.812	8.48	0.613	11.14	0.634
	50	5.96	0.523	13.23	0.844	10.33	0.675	12.79	0.694	4.95	0.478	12.04	0.834	8.94	0.614	11.78	0.681

where "ϵ" denotes the label error rate before label correction.

that the accuracy will improve if it is converted to utterance-level training datasets. So, a straightforward application is to use chunk-level predictions from a well-trained model to re-label the training datasets, and we call this application *label correction*. This is of practical interest as label-corrected datasets can then be beneficially used to retrain front-end networks and back-end models.

To verify this method, we apply label correction on synthetic close-set noisy datasets. The Front6 extractors trained on different label error rates are utilized to predict the chunk-level labels for the corresponding dataset. Specifically, in the label-prediction process, the inputs for the network are the chunk-level samples from each utterance, while the output is the speaker label corresponding to each chunk sample. Then, each utterance is relabeled with the prediction that occurs the most frequently in its multiple chunks. As expected, the utterance-level prediction accuracy of the dataset with label noise is further improved through label correction, and the updated label error rate (one minus the corresponding prediction accuracy) is significantly reduced compared to the original error rate. This is summarized as follows: 5% $\rightarrow$ 1.2%, 10% $\rightarrow$ 1.4%, 20% $\rightarrow$ 1.9%, 30% $\rightarrow$ 2.6%, and 50% $\rightarrow$ 8.6%. Then, we use the label-corrected datasets to retrain the speaker recognition systems. The final results are shown in Table VIII. One can clearly observe that this method corrects erroneous labels even in the case of high error rates, thereby significantly alleviating adverse effects from mislabeling.

B. *Label Denoising on Real-word NIST SRE04-10 Dataset*

We further investigate whether there are mislabels in the original "clean" NIST SRE04-10 dataset. This experiment adopts more rigorous methods, including network prediction, NL-PLDA estimation, and human validation, and we call this process *label denoising*. Specifically, the Front6 and NL-PLDA trained on the original dataset are used to filter out high-confidence noisy samples. Two types of samples are filtered out: those with predicted labels that are inconsistent with the ground-truth labels and those with low latent identity ($z_{n,m} < 0.1$). We find that most of the samples are overlapping. We obtain a sub-dataset that is 1.2% the size of the original SRE04-10 dataset. Then, we identify these erroneous labels by human validation and find that more than half of these samples are indeed listen-clearly mislabeled. Common types of corrupted labels include *gender error* (sentences with female sentences mixed in the category male or vice versa), *language error* (multiple languages mixed in the speaker sample and not sounding like the same person), and *non-speech* (barely audible human voice). Examples of mislabeled audio files are publicly available[3]. Finally, we retrain the x-vector baseline on the original dataset with these samples removed, and the results are shown in Table IX. Compared with the x-vector in Table II, slight improvements can be observed on all four evaluation sets. In addition, a relatively clean version of the SRE04-10 *spk2utt* file that contains speaker-to-utterance mappings is also available on the website[3].

TABLE IX
RESULTS OF X-VECTOR TRAINED ON CLEAN DATASET

	SRE16 Tagalog		SRE16 Cantonese		SRE18 CMN2 Dev		SRE18 CMN2 Eval	
	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF
PLDA	4.13	0.394	11.55	0.770	8.59	0.576	10.55	0.622
NL-PLDA	4.13	0.394	11.56	0.797	8.56	0.576	10.56	0.621

VII. CONCLUSIONS

In this paper, we demonstrate that label noise leads to significant performance degradation for both the x-vector front-end and PLDA back-end. Then, we propose a simple yet effective approach to combat label noise in the front-end training. Our

[3]http://dwz.date/fbC5

proposed framework contains three strategies, including a label confidence training scheme, a posterior probability re-scaling strategy, and an improved AM-Softmax loss function. When progressively combining these three strategies, experiments conducted on the pooled SWBD and SRE04-10 datasets show consistent improvements in robustness against label noise. Since a speaker recognition system consists of both a front-end and a back-end, it is necessary to optimize both to achieve the best performance. Consequently, we also optimize the back-end PLDA when the training labels are noisy. When combining the optimized front-end and back-end, the whole speaker recognition system demonstrates strong resistance to noisy labels.

In addition, we show two practical applications of this improved system, including label correction and label denoising. Label correction is used to correct noisy labels that occur in a dataset. We propose correcting noisy samples based on utterance chunk-level predictions from a well-trained network. Experimental results show that label correction greatly reduces the number of noisy samples within a dataset. Therefore, models retrained on a label corrected dataset perform similarly to those trained on a clean dataset. Besides, we apply label denoising to the real-work NIST SRE04-10 dataset to weed out the original error labels, where both the front-end and back-end are used to algorithmically filter out high-confidence noisy samples, and then we verify them by human validation. As a result, we verify that approximately 1% of the samples are noisy in the original SRE04-10 dataset. Experimental results show that models trained on the label denoised datasets achieve slight improvements compared to the baseline system.

In the future, we are interested in validating our method on other front-end networks and conducting experiments on real-work label noise datasets. In addition, we plan to apply this approach to semi-supervised learning and self-supervised learning networks.

Appendix A
PLDA Learning Algorithm with Noisy Labels

This Appendix presents the EM algorithm for the parameter $\{\mu, \mathbf{\Sigma}_b, \mathbf{\Sigma}_w\}$ learning of NL-PLDA [17]. For convenience, let $\langle\cdot\rangle$ denote the expectation of a given random variable.

In the E-step, let us first pre-compute the number of error samples as:

$$N_e = \sum_{n=1}^{N} \left(1 - \langle z_{n,l_n}\rangle\right), \tag{14}$$

the number of samples for the m-th individual:

$$N_m = \sum_{n=1}^{N} \langle z_{n,m}\rangle, \quad 1 \le m \le M \tag{15}$$

the first-order moment for the m-th individual:

$$\mathbf{r}_{x,m} = \sum_{n=1}^{N} \langle z_{n,m}\rangle \mathbf{x}_n, \quad 1 \le m \le M \tag{16}$$

and the global second-order moment:

$$\mathbf{R}_x = \sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^{\mathrm{T}}. \tag{17}$$

Then, we compute the first and second moments of the latent variables:

$$\langle \mathbf{y}_m\rangle = \mathbf{\Phi}_m^{-1}\left(\mathbf{\Sigma}_b \boldsymbol{\mu} + \mathbf{\Sigma}_w \mathbf{r}_{x,m}\right), \tag{18}$$

$$\left\langle \mathbf{y}_m \mathbf{y}_m^{\mathrm{T}}\right\rangle = \mathbf{\Phi}_m^{-1} + \langle \mathbf{y}_m\rangle \langle \mathbf{y}_m\rangle^{\mathrm{T}}, \tag{19}$$

where

$$\mathbf{\Phi}_m = \mathbf{\Sigma}_b + N_m \mathbf{\Sigma}_w. \tag{20}$$

Next, we need to compute the following auxiliary matrices:

$$\mathbf{r}_y = \sum_{m=1}^{M} \langle \mathbf{y}_m\rangle, \tag{21}$$

$$\mathbf{R}_y^o = \sum_{m=1}^{M} \left\langle \mathbf{y}_m \mathbf{y}_m^{\mathrm{T}}\right\rangle, \tag{22}$$

$$\mathbf{R}_y = \sum_{m=1}^{M}\sum_{n=1}^{N} \langle z_{n,m}\rangle \left\langle \mathbf{y}_m \mathbf{y}_m^{\mathrm{T}}\right\rangle, \tag{23}$$

$$\mathbf{R}_{xy} = \sum_{m=1}^{M} \mathbf{r}_{x,m} \langle \mathbf{y}_m\rangle^{\mathrm{T}}. \tag{24}$$

For the M-step, we update the matrix of label latent identity $\mathcal{Z}$ by:

$$\langle z_{n,m}\rangle = \frac{i_{n,m}}{\sum_{j=1}^{M} i_{n,j}}, \tag{25}$$

where

$$\begin{aligned} i_{n,m} = {} & P\left(l_n \mid z_{n,m}=1, \epsilon\right) \mathcal{N}\left(\mathbf{x}_n \mid \langle \mathbf{y}_m\rangle, \mathbf{\Sigma}_w^{-1}\right) \\ & \times \exp\left(-\frac{1}{2}\operatorname{tr}\left\{\mathbf{\Sigma}_w \mathbf{\Phi}_m^{-1}\right\}\right) \end{aligned} \tag{26}$$

$$P\left(l_n \mid z_{n,m}=1, \epsilon\right) = \begin{cases} 1-\epsilon, & \text{if } l_n = m \\ \frac{\epsilon}{M-1}, & \text{else} \end{cases} \tag{27}$$

$$\epsilon = \frac{N_e}{N}. \tag{28}$$

After that we update the NL-PLDA parameters as follows:

$$\mu = \frac{\mathbf{r}_y}{M}, \tag{29}$$

$$\mathbf{\Sigma}_b = M\left(\mathbf{R}_y^o - \frac{\mathbf{r}_y \mathbf{r}_y^{\mathrm{T}}}{M}\right)^{-1}, \tag{30}$$

$$\mathbf{\Sigma}_w = N\left(\mathbf{R}_x - \mathbf{R}_{xy} - \mathbf{R}_{xy}^{\mathrm{T}} + \mathbf{R}_y\right)^{-1}. \tag{31}$$

REFERENCES

[1] K. A. Lee, H. Yamamoto, K. Okabe, Q. Wang, L. Guo, T. Koshinaka, J. Zhang, and K. Shinoda, "NEC-TT system for mixed-bandwidth and multi-domain speaker recognition," *Comput. Speech Lang.*, vol. 61, p. 101033, 2020.

[2] 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.

[3] S. Ioffe, "Probabilistic linear discriminant analysis," in *Proc. ECCV*, 2006, pp. 531–542.

[4] S. J. Prince and J. H. Elder, "Probabilistic linear discriminant analysis for inferences about identity," in *Proc. ICCV*, 2007, pp. 1–8.

[5] N. Brümmer and E. De Villiers, "The speaker partitioning problem," in *Proc. Odyssey*, 2010, p. 34.

[6] P. Kenny, "Bayesian speaker verification with heavy-tailed priors," in *Proc. Odyssey*, vol. 14, 2010.

[7] Y. Cai, L. Li, A. Abel, X. Zhu, and D. Wang, "Deep normalization for speaker vectors," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 29, pp. 733–744, 2020.

[8] D. Garcia-Romero and C. Y. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems," in *Proc. Interspeech*, 2011, pp. 249–252.

[9] J. Villalba, N. Chen, D. Snyder, D. Garcia-Romero, A. McCree, G. Sell, J. Borgstrom, L. P. García-Perera, F. Richardson, R. Dehak *et al.*, "State-of-the-art speaker recognition with neural network embeddings in NIST SRE18 and speakers in the wild evaluations," *Comput. Speech Lang.*, vol. 60, p. 101026, 2020.

[10] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *Proc. ICASSP*, 2018, pp. 5329–5333.

[11] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Proc. Interspeech*, 2015, pp. 3214–3218.

[12] D. Snyder, D. Garcia-Romero, G. Sell, A. McCree, D. Povey, and S. Khudanpur, "Speaker recognition for multi-speaker conversations using x-vectors," in *Proc. ICASSP*, 2019, pp. 5796–5800.

[13] B. Desplanques, J. Thienpondt, and K. Demuynck, "ECAPA-TDNN: Emphasized channel attention, propagation and aggregation in TDNN based speaker verification," in *Proc. Interspeech*, 2020, pp. 1–5.

[14] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, "CosFace: Large margin cosine loss for deep face recognition," in *Proc. CVPR*, 2018, pp. 5265–5274.

[15] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "ArcFace: Additive angular margin loss for deep face recognition," in *Proceedings CVPR*, 2019, pp. 4690–4699.

[16] NIST, "NIST 2018 speaker recognition evaluation plan," 2018. [Online]. Available: https://www.nist.gov/system/files/documents/2018/08/17/sre18_eval_plan_2018-05-31_v6.pdf

[17] B. J. Borgström and P. Torres-Carrasquillo, "Bayesian estimation of PLDA with noisy training labels, with applications to speaker verification," in *Proc. ICASSP*, 2020, pp. 7594–7598.

[18] A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman, "Voxceleb: Large-scale speaker verification in the wild," *Comput. Speech Lang.*, vol. 60, p. 101027, 2020.

[19] R. Sachdeva, F. R. Cordeiro, V. Belagiannis, I. Reid, and G. Carneiro, "Evidentialmix: Learning with combined open-set and closed-set noisy labels," in *Proc. WACV*, 2021, pp. 3607–3615.

[20] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *arXiv preprint arXiv:1611.03530*, 2016.

[21] D. Arpit, S. K. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. C. Courville, Y. Bengio *et al.*, "A closer look at memorization in deep networks," in *Proc. ICML*, 2017, pp. 233–242.

[22] S. E. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, "Training deep neural networks on noisy labels with bootstrapping," in *Proc. ICLR (Workshop)*, 2015.

[23] D. Tanaka, D. Ikami, T. Yamasaki, and K. Aizawa, "Joint optimization framework for learning with noisy labels," in *Proc. CVPR*, 2018, pp. 5552–5560.

[24] E. Arazo, D. Ortego, P. Albert, N. O'Connor, and K. McGuinness, "Unsupervised label noise modeling and loss correction," in *Proc. ICML*, 2019, pp. 312–321.

[25] X. Wang, S. Wang, J. Wang, H. Shi, and T. Mei, "Co-mining: Deep face recognition with noisy labels," in *Proc. ICCV*, 2019, pp. 9358–9367.

[26] X. Xia, T. Liu, B. Han, N. Wang, M. Gong, H. Liu, G. Niu, D. Tao, and M. Sugiyama, "Part-dependent label noise: Towards instance-dependent label noise," in *Proc. NIPS*, vol. 33, 2020, pp. 7597–7610.

[27] D. Ortego, E. Arazo, P. Albert, N. E. O'Connor, and K. McGuinness, "Towards robust learning with different label noise distributions," in *Proc. ICPR*, 2021, pp. 7020–7027.

[28] H. Song, M. Kim, D. Park, Y. Shin, and J.-G. Lee, "Learning from noisy labels with deep neural networks: A survey," *arXiv preprint arXiv:2007.08199*, 2020.

[29] S. Zheng, G. Liu, H. Suo, and Y. Lei, "Towards a fault-tolerant speaker verification system: A regularization approach to reduce the condition number," in *Proc. Interspeech*, 2019, pp. 4065–4069.

[30] M. Pham, Z. Li, and J. Whitehill, "How does label noise affect the quality of speaker embeddings?" in *Proc. Interspeech*, 2020, pp. 3216–3220.

[31] F. Tong, Y. Liu, J. Wang, L. Li, and Q. Hong, "Automatic error correction for speaker embedding learning with noisy labels," in *Accepted at Proc. Interspeech*, 2021.

[32] F. Wang, J. Cheng, W. Liu, and H. Liu, "Additive margin softmax for face verification," *IEEE Signal Process. Lett.*, vol. 25, no. 7, pp. 926–930, 2018.

[33] R. A. Fisher, "The use of multiple measurements in taxonomic problems," in *Ann. Eugenics*, vol. 7, no. 2, 1936, p. 179–188.

[34] A. Sizov, K. A. Lee, and T. Kinnunen, "Unifying probabilistic linear discriminant analysis variants in biometric authentication," in *Proc. Joint IAPR Int. Workshops Statistical Techn. Pattern Recognit. (SPR) Struct. Syntactic Pattern Recognit. (SSPR)*, 2014, pp. 464–475.

[35] D. Garcia-Romero, A. McCree, S. Shum, N. Brummer, and C. Vaquero, "Unsupervised domain adaptation for i-vector speaker recognition," in *Proc. Odyssey: Speaker Lang. Recognit. Workshop*, vol. 8, 2014.

[36] Y. Tu, M.-W. Mak, and J.-T. Chien, "Variational domain adversarial learning with mutual information maximization for speaker verification," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 28, pp. 2013–2024, 2020.

[37] K. A. Lee, Q. Wang, and T. Koshinaka, "The CORAL+ algorithm for unsupervised domain adaptation of PLDA," in *Proc. ICASSP*, 2019, pp. 5821–5825.

[38] S. Wang, Y. Yang, Z. Wu, Y. Qian, and K. Yu, "Data augmentation using deep generative models for embedding based speaker recognition," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 28, pp. 2598–2609, 2020.

[39] D. Wang, "Remarks on optimal scores for speaker recognition," *arXiv preprint arXiv:2010.04862*, 2020.

[40] W. Hu, T. Miyato, S. Tokui, E. Matsumoto, and M. Sugiyama, "Learning discrete representations via information maximizing self-augmented training," in *Proc. ICML*, 2017, pp. 1558–1567.

[41] Z. Bai and X. Zhang, "Speaker recognition based on deep learning: An overview," *Neural Netw.*, 2021.

[42] J. Deng, J. Guo, T. Liu, M. Gong, and S. Zafeiriou, "Sub-center ArcFace: Boosting face recognition by large-scale noisy web faces," in *Proc. ECCV*, 2020, pp. 741–757.

[43] X. Xiang, S. Wang, H. Huang, Y. Qian, and K. Yu, "Margin matters: Towards more discriminative deep neural network embeddings for speaker recognition," *Proc. APSIPA ASC*, pp. 1652–1656, 2019.

[44] Y. Liu, L. He, and J. Liu, "Large margin softmax loss for speaker verification," in *Proc. Interspeech*, 2019, pp. 2873–2877.

[45] Z. Chen, Z. Ren, and S. Xu, "A study on angular based embedding learning for text-independent speaker verification," in *Proc. APSIPA ASC*, 2019, pp. 445–449.

[46] M. Rybicka and K. Kowalczyk, "On parameter adaptation in softmax-based cross-entropy loss for improved convergence speed and accuracy in DNN-based speaker recognition," in *Proc. Interspeech*, 2020, pp. 3805–3809.

[47] M. Zhu and A. M. Martínez, "Optimal subclass discovery for discriminant analysis," in *Proc. CVPR (Workshop)*, 2004, pp. 97–97.

[48] H. Wan, H. Wang, G. Guo, and X. Wei, "Separability-oriented subclass discriminant analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 2, pp. 409–422, 2017.

[49] A. K. Sarkar, Z.-H. Tan, H. Tang, S. Shon, and J. Glass, "Time-contrastive learning based deep bottleneck features for text-dependent speaker verification," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 27, no. 8, pp. 1267–1279, 2019.

[50] F. Tong, M. Zhao, J. Zhou, H. Lu, Z. Li, L. Li, and Q. Hong, "ASV-Subtools: Open source toolkit for automatic speaker verification," in *Proc. ICASSP*, 2021, pp. 6184–6188.