End-to-end Speaker Verification Via Curriculum Bipartite Ranking Weighted Binary Cross-entropy

Zhongxin Bai, Jianyu Wang, Xiao-Lei Zhang, and Jingdong Chen

***Abstract*—End-to-end speaker verification achieves the verification through estimating directly the similarity score between a pair of utterances, which is formulated as a binary (i.e., target versus non-target) classification problem. Unlike the stage-wise method, an end-to-end verification approach optimizes the evaluation metrics directly and its output layer is parameter-free, which can save great computing and memory resources. However, there are two important issues that need to be meticulously handled in training an end-to-end speaker verification model. The first one is how to deal with severely imbalanced trials, i.e., the number of target trials is much smaller than that of nontarget trials, and the other is about how to handle easy trials that do not help improve the model in training. To circumvent these two issues, we propose in this paper a binary cross-entropy (BCE) type of loss function and present a method to train the deep neural network (DNN) models based on the proposed loss function for end-to-end speaker verification. The training process employs a bipartite ranking method to deal with the trial imbalance problem and a curriculum learning method to help improve both the training stability and performance of the model by selecting non-target trials from easy to hard ones gradually along the convergence process. Since the training process employs bipartite ranking and curriculum learning and the loss function is of the generalized BCE form, we name the new approach *curriculum bipartite ranking weighted binary cross-entropy* (CBRW-BCE). Experimental results show that the model trained with CBRW-BCE not only achieves the state-of-the-art performance but is also well calibrated.**

***Index Terms*—End-to-end, metric learning, bipartite ranking, curriculum learning, calibration.**

I. Introduction

SPEAKER verification aims to verify whether a given utterance is pronounced by a hypothesized speaker based on some utterances pre-recorded from that speaker [1], [2]. State-of-the-art speaker verification systems can be broadly categorized into two classes: *stage-wise* and *end-to-end* ones. A stage-wise system consists of a front-end speaker feature extractor and a back-end similarity score calculator. The former extracts speaker features by projecting utterances of different length into a fixed, low-dimensional feature space. The representative extractors include the d-vector [3], the x-vector [4] and many other deep embedding methods [5], [6], [7], [8], [9], [10], [11]. The latter calculates similarity scores between the test and enrollment embeddings, and decides whether the utterances in the trial are from a same speaker by comparing the similarity score with a predefined threshold. The representative scoring methods for back-end include probabilistic linear discriminant analysis [12], cosine similarity, metric leaning based ones [13], [14], and others [15], [16]. The threshold can be either an empirical one tuned from a development set or one calculated according to the Bayes decision theory if the scores are calibrated [19], [20]. In contrast, an end-to-end system takes a pair of utterances (called a trial) as input and outputs the similarity score directly [17], [18]. Again, calibrated score is preferred from the decision making perspective.

One major difference between the stage-wise and end-to-end speaker verification lies in the loss function, as summarized in Table I. The earliest loss function developed for the stage-wise speaker verification is the so-called Softmax loss function[1], which attempts to maximize the classification accuracy for speakers in the training data [3], [4]. It is known that within-speaker variances play an important role on the generalization ability of speaker verification systems. Generally, the smaller the within-speaker variances, the better is the generalization ability. However, Softmax does not explicitly consider to reduce the within-speaker variances. To deal with this issue, several variants of Softmax, e.g., the ASoftmax [21], [22], [23], the AMSoftmax [24], and the ArcSoftmax [25], were developed. These variants introduce a margin to explicitly constrain the embedding space to have small within-speaker variances, thereby improving the generalization ability of the verification system. Besides, they are optimized to match the cosine similarity scoring back-end directly, and are therefore more suited to many cosine-similarity-based verification systems than their original counterpart. What in common with the aforementioned loss functions is that they all require a parametric classification layer (so, we call them the classification-based loss functions). In practice, the number of parameters of this layer increases dramatically with the increase of the number of training speakers, which can be problematic to the network training if the number of speakers is extremely large. To mitigate this issue, methods such as low-rank approximation to the weight matrix of the classification layer [26] and computation acceleration for the Softmax function [27] are developed.

In contrast, the end-to-end speaker verification systems generally use metric-learning-based loss functions, which learn

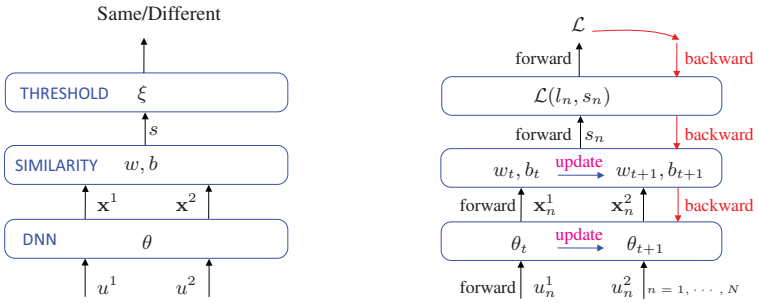
Z. Bai, J. Wang, and X.-L. Zhang are with the Center of Intelligent Acoustics and Immersive Communications (CIAIC), Shaanxi Provincial Key Laboratory of Artificial Intelligence, and the School of Marine Science and Technology, Northwestern Polytechnical University, 127 Youyi West Road, Xi'an, Shaanxi 710072, China (e-mail: zxbai@mail.nwpu.edu.cn, alexwang96@mail.nwpu.edu.cn, xiaolei.zhang@nwpu.edu.cn).
J. Chen is with CIAIC and Shaanxi Provincial Key Laboratory of Artificial Intelligence, Northwestern Polytechnical University, 127 Youyi West Road, Xi'an, Shaanxi 710072, China (e-mail: jingdongchen@ieee.org).

[1]This paper defines the Softmax loss function as a combination of the last fully connected layer, the Softmax function, and the cross-entropy.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TASLP.2022.3161155, IEEE/ACM Transactions on Audio, Speech, and Language Processing

TABLE I: Comparison of the loss functions for speaker verification where "classification" and "metric" denote, respectively, the classification- and metric-leaning-based loss functions, and "optimizing on all speakers" means that the training is conducted with respect to all the training speakers at each training iteration.

Loss types	Loss functions	Label types	Parametric classifier	Optimizing on all speakers	Calibrated scores	Multi-label classification	Hard negative mining requirement	Trial imbalance problem
Classification	Softmax [3], [4]	One-hot	✓	✓	✗	✗	✗	✗
	A/AM/Arc-Softmax [22], [24], [25]	One-hot	✓	✓	✗	✗	✗	✗
Metric	GE2E [28]	One-hot	✗	✗	✗	✗	✗	✗
	Prototype [29], [30]	One-hot	✗	✗	✗	✗	✗	✗
	pAUC [31]	Binary	✓/✗	✓/✗	✗	✓	✗	✗
	Triplet [32], [33]	Triple	✗	✗	✗	✗	✓	✓
	Contrastive [34]	Binary	✗	✗	✗	✓	✓	✓
	BCE [17], [35], [18]	Binary	✗	✗	✓	✓	✓	✓
	CBRW-BCE (ours)	Binary	✗	✗	✓	✓	✗	✗



(a) Test stage. (b) Training stage.

Fig. 1: Workflows of the training and test stages of an end-to-end speaker verification system. In the test stage, an enrollment utterance u^1 and a test utterance u^2 are first fed into a DNN (parameterized by θ), which produces two embeddings $\mathbf{x}^1$ and $\mathbf{x}^2$. Then, a similarity score s between $\mathbf{x}^1$ and $\mathbf{x}^2$ is computed by a scoring function (parameterized by w, b). Finally, a hard decision on whether the two inputs are from the same speaker are made by comparing s with a threshold ξ. In the training stage, a manually constructed pairwise training set $\{(u_n^1, u_n^2)\}_{n=1}^N$ is first fed forward into the model, which calculates a training loss $\mathcal{L}(l_n, s_n)$, where l_n denotes the ground truth similarity of the input utterances. Then, the loss is minimized by back-propagation (as shown by the downward red arrows), which updates the model parameters iteratively (as shown by the blue left arrows) where the subscript $(\cdot)_t$ denotes the tth training iteration.

a similarity function from the training utterances directly. As the classification layer of this category of loss functions is nonparametric, handling large number of speakers in training is not a issue.

According to the number of utterances used to construct an independent element, the metric-learning-based loss functions can be classified into prototype based [28], [29], [30], quadruplet [31], triplet [32], [33], and pairwise [34], [17], [35], [18] ones. Among those, only the pairwise loss functions, e.g., the binary cross-entropy (BCE), which work with binary labels, are able to realize rigorous end-to-end training since the binary labels based on pairs of training utterances do not need accurate speaker identities. Furthermore, these loss functions can be generalized to work with multi-speaker applications [36], [37] by simply formulating the problem as one of multi-label classification [38]. Another property of the pairwise loss functions that has been overlooked is that they have the potential of implementing end-to-end calibration, which trains the model to directly output well calibrated scores. From the aforementioned perspectives, the pairwise loss functions should in principle be best suited to end-to-end speaker verification. However, in reality, pairwise loss functions are often founded inferior to others in terms of performance. The underlying reasons are multiple, which motivate us to investigate, thereby unlocking the full potential of the pairwise loss functions.

To properly train a deep neural network (DNN) with pairwise loss functions, some challenging issues have to be carefully handled. The first one is about the severely imbalanced positive and negative trials. Generally, positive trials are far fewer than negative trials in the training data regardless of whether the training trials are constructed from well labeled corpora or collected from real applications. To circumvent this issue, under-sampling of the negative trials or over-sampling of the positive trials is usually adopted. But the over-sampling strategy may easily lead to model overfitting and/or introduces additional noise while the down-sampling strategy may lose informative data. The second issue is that there are generally many easy-to-classify negative training trials in the training data, which are non-informative and may result in an inferior DNN model after training. Therefore, it is important to select informative negative trials, which is a process known as "hard negative mining". A common approach to this is to apply a fixed mining strategy during the entire training process, which attempts to remove those easy to classify negative trials while selecting the hard negative trials as they

may contribute significantly the performance improvement. However, this fixed mining strategy is in general not optimal. On the one hand, at the beginning of the training process in which the model classifies training trials arbitrarily, using only hard negative trials may cause the training converge to some local minima [40] and the resulting model is biased toward discriminating the hard negative trials while failing to classify many easy negative trials that have not yet been correctly classified. Even worse, if the selected trials are too difficult, the optimization process may become unstable [31] and even divergent [29]. On the other hand, at the late stage of the training in which most negative trials are correctly classified, using too many well classified negative trials will make the model neglect missclassified hard negative trials, which should contribute significantly to the performance improvement. The third issue is about score calibration, a process to convert similarity scores into proper log-likelihood ratios (LLRs) [39], which are important for making hard decisions and providing a probabilistic meaning for such applications as forensic [54]. The model trained with end-to-end loss function is expected to output calibrated scores directly, while this has never been studied before [39].

To deal with the aforementioned issues, we propose in this work a pairwise loss function, which is named as *curriculum bipartite ranking weighted BCE* (CBRW-BCE), and present an approach based on the new loss function to training speaker verification models with good discriminative and calibration performance. The major contributions of this paper are as follows.

- **A pairwise loss function, i.e., CBRW-BCE, is proposed, which jointly considers the discriminant and calibration performance.** It is a weighted binary cross-entropy type of loss function, which enables the trained model to be both discriminative and well calibrated. This method of score calibration has never been investigated in the end-to-end training.
- **A bipartite ranking method is applied in the training process to circumventing the issue of imbalanced trials.** With the use of bipartite ranking [41], this method learns a ranking function, which assigns a similarity score to every trial in such a way that the positive trials have higher scores than the negative trials. To train such a ranking function, every positive trial is compared to all the negative trials, thereby avoiding the trial imbalance problem naturally.
- **A curriculum learning method is proposed to help improve both the training stability and performance of the model** by conducting dynamic hard negative mining. It dynamically selects a number of negative trials according to the convergence status of the model during training, which enables the training process converge to a good state. To the best of our knowledge, it is the first time that the convergence state of the network is fed back into the mining of hard negative samples.

The remainder of this paper is organized as follows. Section II discusses the problem of end-to-end speaker verification. Detailed description of the proposed loss function is presented in Section III. In Sections IV and V, we justify through experiments the effectiveness of the developed method. Finally, important conclusions are given in Section VI.

II. Preliminaries

The test and training of a speaker verification model can be formulated as follows. Without loss of generality, we assume that u^1 represents an enrollment utterance (note that there can be multiple enrollment utterances from one speaker) and u^2 denotes a test utterance. As illustrated in Fig. 1(a), a DNN-based speaker verification model $f(\cdot)$ is formulated as follows:

$$s = f(u^1, u^2; \theta, w, b) \underset{H_1}{\overset{H_0}{\gtrless}} \xi, \tag{1}$$

where θ, w and b are learnable parameters, and ξ is a decision threshold. The hypothesis H_0 denotes that (u^1, u^2) are from a same speaker while the hypothesis H_1 denotes that (u^1, u^2) are from different speakers. Let us denote $(\mathbf{x}^1, \mathbf{x}^2)$ as the embedding vectors of (u^1, u^2) learned by DNN.

End-to-end speaker verification systems are trained on an iterative way. Let us denote by $\mathcal{T}_t$ the mini-batch of training trials at the tth training iteration, i.e.,

$$\mathcal{T}_t = \{(\mathbf{x}_n^1, \mathbf{x}_n^2, l_n) | \mathbf{x}_n^1, \mathbf{x}_n^2 \in \mathbb{R}^{\mathrm{d}}, n = 1, 2, \cdots, N\}, \tag{2}$$

where N denotes the total number of training trials in the mini-batch, $(\mathbf{x}_n^1, \mathbf{x}_n^2)$ denotes the embedding vectors for the nth trial, and $l_n \in \{1, 0\}$ is the ground truth label describing whether the nth paired utterances are from a same speaker. It is evident that every mini-batch may have different training trials. Suppose that we evaluate the similarity between two utterances by the linearly transformed cosine similarity:

$$s_n = w \times \frac{(\mathbf{x}_n^1)^{\mathrm{T}} \mathbf{x}_n^2}{\|\mathbf{x}_n^1\| \|\mathbf{x}_n^2\|} + b, \tag{3}$$

where $(\cdot)^{\mathrm{T}}$ denotes the transpose operator, $\|\cdot\|$ denotes the ℓ_2 norm, and w and b are initialized to be 10 and -5 respectively. Given (2) and (3), the training of a speaker verification model can be formulated as the following metric learning problem:

$$\theta^*, w^*, b^* = \arg\min_{\theta, w, b} \mathcal{L}(\mathcal{S}_t; \theta, w, b), \tag{4}$$

where $\mathcal{S}_t = \{(s_n, l_n) | n = 1, 2, \cdots, N\}$, and $\mathcal{L}(\cdot)$ is a metric-learning-based loss function. See Table II for the notation used in this paper.

In this work, we use the BCE loss function, which is defined as:

$$\begin{aligned} \mathcal{L}_{\mathrm{BCE}} = &- \frac{1}{\upsilon_l} \sum_{n=1}^{N} l_n \log\big[P(s_n)\big] \\ &- \frac{1}{N - \upsilon_l} \sum_{n=1}^{N} (1 - l_n) \log\big[1 - P(s_n)\big], \end{aligned} \tag{5}$$

where $\upsilon_l = \sum_{n=1}^{N} l_n$, and $P(\cdot)$ denotes the probabilistic output of DNN. In this work, the sigmoid activation function is used in the last layer, so we have $P(s_n) = \frac{1}{1+e^{-s_n}}$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TASLP.2022.3161155, IEEE/ACM Transactions on Audio, Speech, and Language Processing

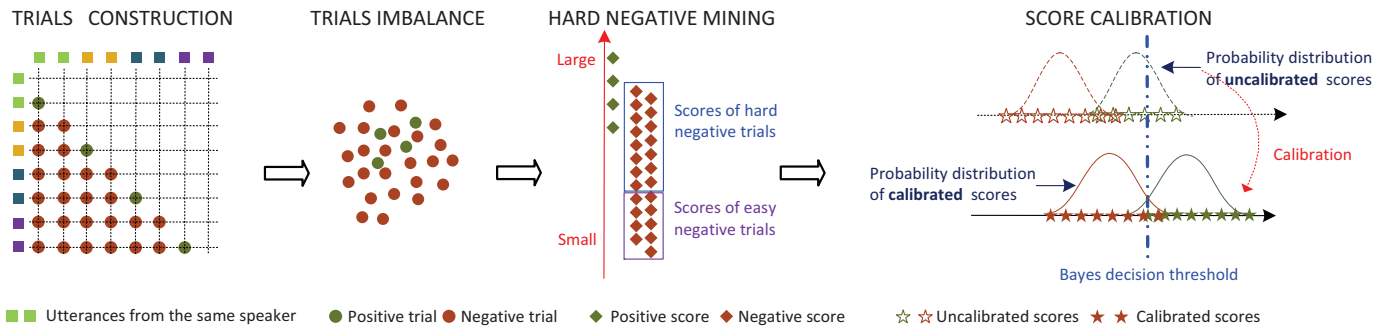


Fig. 2: Illustration of the three difficult problems in the training of an end-to-end speaker verification model. TRIALS CONSTRUCTION: the process of constructing training trials from a batch of individual samples; TRIALS IMBALANCE: the number of the constructed negative trials is much larger than the number of the constructed positive trials; HARD NEGATIVE MINING: the process of mining informative hard negative trials for training a stronger model; SCORE CALIBRATION: the process of calibrating similarity scores into log likelihood ratios for facilitating the hard decisions under the Bayes decision theory.

TABLE II: Main notation used in this work.

Notation	Definition
(u^1, u^2)	A pair of utterance, where u^1 denotes an enrollment utterance and u^2 denotes a test utterance.
$(\mathbf{x}^1, \mathbf{x}^2)$	Embeddings of u^1 and u^2 extracted by DNN.
$\mathcal{T}_t$	Set of the embeddings of the training trials of the tth training iteration.
$\mathcal{S}_t$	Set of the similarity scores calculated from $\mathcal{T}_t$.
$\mathcal{N}_t$	Subset of $\mathcal{S}_t$ consists of negative similarity scores.
$\mathcal{P}_t$	Subset of $\mathcal{S}_t$ consists of positive similarity score.
θ	Parameters of DNN.
w, b	Parameters of the similarity scoring function.
$(\cdot)_t$	Denotes for the tth training iteration.
$(\cdot)_n$, $(\cdot)_i$, $(\cdot)_j$	Indices for "training trials", "negative training trias", and "positive training trials", respectively.
$(\cdot)^*$	Stands for the optimal parameters.

III. The CBRW-BCE Approach

As discussed previously and also illustrated in Fig. 2, three challenging issues have to be handled meticulously to train a good end-to-end speaker verification model, i.e., imbalanced trails, hard negative mining, and score calibration. In the following subsections, we will present a method and discuss how to deal with these issues.

A. Deep metric learning based on the bipartite ranking

This subsection addresses the issue of how to deal with severely imbalanced trials in training. For simplicity, the similarity score set $\mathcal{S}_t$ in (4) is further divided into a negative subset and a positive subset, i.e., $\mathcal{N}_t = \{s_i | i = 1, 2, \cdots, I\}$ and $\mathcal{P}_t = \{s_j | j = 1, 2, \cdots, J\}$ respectively, where $I + J = N$ and $\mathcal{N}_t \cup \mathcal{P}_t = \mathcal{S}_t$. Bipartite ranking attempts to learn a ranking function by assigning higher scores to positive trials while lower scores to negative trials for the binary classification problem in (1), which can be formulated as minimization of the following loss function:

$$\mathcal{L}(\mathcal{N}_t, \mathcal{P}_t; \theta, w, b) = \frac{1}{IJ} \sum_{i=1}^{I} \sum_{j=1}^{J} \Big[\mathbb{I}(s_j < s_i) + \frac{1}{2} \mathbb{I}(s_i = s_j) \Big], \tag{6}$$

where $s_i \in \mathcal{N}_t$, $s_j \in \mathcal{P}_t$, and $\mathbb{I}(\cdot)$ is an indicator function whose value is 1 if the input statement is true and 0 otherwise. As shown in [43], the right-hand side of (6) can be expressed as

$$\mathcal{L}(\mathcal{N}_t, \mathcal{P}_t; \theta, w, b) = 1 - \widehat{\mathrm{AUC}}(\mathcal{N}_t, \mathcal{P}_t), \tag{7}$$

where $\widehat{\mathrm{AUC}}(\cdot)$ is the so-called Wilcoxon-Mann-Whitney statistic [44], which represents an empirical area under the ROC curve (AUC) estimated over $\mathcal{N}_t$ and $\mathcal{P}_t$. Therefore, minimizing (6) is equivalent to maximizing the AUC but with a benefit of naturally mitigating the problem of imbalanced trials.

Unfortunately, the integer optimization problem in (6) is NP-hard. A common way to circumvent this NP-hard problem is to relax the indicator function by a differentiable function, e.g., $\mathbb{I}(z \leq 0) \leq \max(0, 1 - z)$. In this work, we use a hinge function, i.e., $\max(0, \delta - z)$, with a tunable hyperparameter δ to relax (6) , i.e.,

$$\mathcal{L}(\mathcal{N}_t, \mathcal{P}_t; \theta, w, b) = \frac{1}{IJ} \sum_{i=1}^{I} \sum_{j=1}^{J} \Big[\max(0, \delta - s_j + s_i) + \frac{1}{2} \max(0, \delta) \Big]. \tag{8}$$

The second term in the brackets on the right-hand side of (8) is irrelevant to s_i and s_j and hence neglecting it does not affect the solution of optimization. So, from the optimization perspective, the loss function in (8) is equivalent to the following one:

$$\mathcal{L}(\mathcal{N}_t, \mathcal{P}_t; \theta, w, b) = \frac{1}{IJ} \sum_{i=1}^{I} \sum_{j=1}^{J} \max(0, \delta - s_j + s_i). \tag{9}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TASLP.2022.3161155, IEEE/ACM Transactions on Audio, Speech, and Language Processing

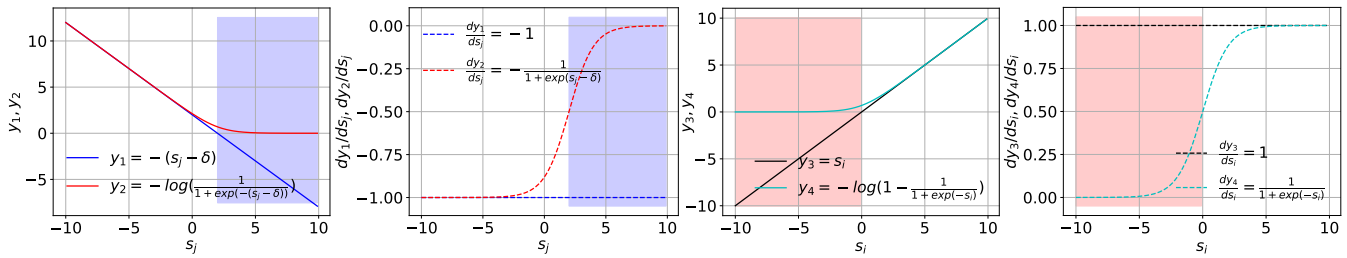


Fig. 3: Illustration of (12) with $\delta = 2$. The first and third subfigures illustrate, respectively, the first and second inequalities in (12), and the second and fourth subfigures illustrate, respectively, the gradients of the two inequalities. The modulus of the gradient is expected to be small in the shaded regions.

It is easy to see from (9) that the loss function increases only if $\delta - s_j + s_i > 0$. Let us define an index matrix $\mathbf{\Pi} \in \{0,1\}^{I\times J}$:

$$\mathbf{\Pi}(i,j) = \begin{cases} 1, & \text{if } s_j - \delta < s_i \\ 0, & \text{otherwise} \end{cases}. \tag{10}$$

Then, (9) can be rewritten as

$$\begin{aligned}\mathcal{L}(\mathcal{N}_t, \mathcal{P}_t; \theta, w, b) =& \frac{1}{IJ}\sum_{i=1}^{I}\sum_{j=1}^{J}\mathbf{\Pi}(i,j)\big[s_i - (s_j - \delta)\big] \\ =& -\sum_{j=1}^{J}\omega_j(s_j - \delta) + \sum_{i=1}^{I}\omega_i s_i, \end{aligned} \tag{11}$$

where δ is a parameter to constrain the model to have small within-speaker variance, and $\omega_j = \frac{1}{IJ}\sum_{i=1}^{I}\mathbf{\Pi}(i,j)$ and $\omega_i = \frac{1}{IJ}\sum_{j=1}^{J}\mathbf{\Pi}(i,j)$, which encode the bipartite ranking information of the training trials, are the weights for cost-sensitive learning to mitigate the trial imbalance problem.

Note that in the training process, s_j reflects the probability of the correct prediction of positive trials. So, the larger the value of s_j, the more accurate is the prediction. In order to make the training process emphasize on the positive trials that are not well predicted and meanwhile neglecting those that are already well classified, we expect the modulus of the gradient of the loss function to be large if the value of s_j is small, and vice versa. The modulus is expected to approach zero if the value of s_j is too large. Note that this should be opposite for negative trials.

To achieve this goal, we further relax (11) to the following:

$$\begin{cases} -(s_j - \delta) \le -\log(\frac{1}{1+e^{-(s_j-\delta)}}) \\ s_i \le -\log(\frac{1}{1+e^{s_i}}) = -\log(1 - \frac{1}{1+e^{-s_i}}). \end{cases} \tag{12}$$

Figure 3 illustrates the relaxation in (12). As seen, the relaxation meets our expectation. There is another reason that we make the relaxation by the sigmoid function, i.e., it enables us to derive a loss function that can jointly maximize the discriminability and calibration performance of the model, which will be discussed later in Section III-C. Substituting (12) into (11) gives the bipartite ranking weighted BCE (BRW-BCE) loss function:

$$\begin{aligned}&\mathcal{L}_{\text{BRW-BCE}}(\mathcal{N}_t, \mathcal{P}_t; \theta, w, b) = \\ &-\sum_{j=1}^{J}\omega_j \log(\frac{1}{1+e^{-(s_j-\delta)}}) - \sum_{i=1}^{I}\omega_i \log(1 - \frac{1}{1+e^{-s_i}}).\end{aligned} \tag{13}$$

Algorithm 1: The sample selection function.

Input: The similarity scores of negative training trials, i.e., $\mathcal{N}_t$; hyperparameters $\alpha_t, \beta_t \in [0,1]$ with $\alpha_t < \beta_t$;

Output: Selected hard negative trials $\widehat{\mathcal{N}_t}$.

1 Replace α_t and β_t with $\hat{\alpha}_t = \lfloor I \times \alpha_t \rfloor + 1$, $\hat{\beta}_t = \lceil I \times \beta_t \rceil$, where the values of $\hat{\alpha}_t$ and $\hat{\beta}_t$ satisfies $1 \le \hat{\alpha}_t \le \hat{\beta}_t \le I$ with I being the size of $\mathcal{N}_t$;

2 Rank all scores in $\mathcal{N}_t$ in ascending order, so that $s_{k_1} < s_{k_2} < \cdots < s_{k_I}$, where the subscripts $\{k_i\}_{i=1}^{I}$ are unduplicated integral numbers between 1 and I ;

3 Select scores lying between the $\hat{\alpha}_t$th and $\hat{\beta}_t$th positions, thereby forming $\widehat{\mathcal{N}_t} = \{s_{k_{\hat{\alpha}_t}}, s_{k_{\hat{\alpha}_t+1}}, \cdots, s_{k_{\hat{\beta}_t-1}}, s_{k_{\hat{\beta}_t}}\}$;

Result: $\widehat{\mathcal{N}_t}$

B. Hard negative mining based on curriculum learning

Many elements in $\mathcal{N}_t$ are computed from easy negative trials, which do no provide useful information for training a good model once they are classified correctly. As a result, it is important to neglect those elements during training. This subsection presents the principle of the proposed hard negative mining method based on curriculum learning.

Basically, the method feeds all negative trials into the loss function at the beginning of the training process, and then gradually removes more and more easy negative trials along the training progress. This strategy will not only help overcome the training instability issue caused by hard negative trials at the early training stage, but also help improve the performance of the final model using the hard negative trials at the late training stage. The detailed process of this strategy is as follows.

First, a *sample selection function* (SSF) based on the similarity scores is applied (the process is shown in Algorithm 1), where two hyperparameters α_t and β_t are used to control the degree of difficulty in classification of the selected samples.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TASLP.2022.3161155, IEEE/ACM Transactions on Audio, Speech, and Language Processing

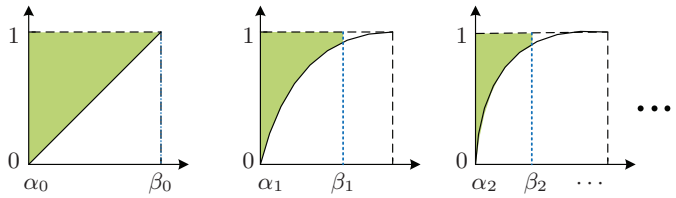


Fig. 4: Illustration of the curriculum hard negative mining from the perspective of partial AUC optimization.

The curriculum hard negative mining is conducted by first initializing α_t and β_t as:

$$\alpha_0 = 0, \quad \beta_0 = 1. \tag{14}$$

Then, with the value of α_t being fixed, we adaptively shrink β_t for every Δ iterations according to

$$\beta_t = \begin{cases} \min(\beta_{t-1}, 1 - P_\Delta), & \text{if } \ t \in \{\Delta, 2\Delta, 3\Delta, \cdots\} \\ \beta_{t-1}, & \text{otherwise} \end{cases}, \tag{15}$$

where

$$P_\Delta = \frac{1}{\Delta} \sum_{m=t-\Delta}^{t-1} \widehat{\mathrm{AUC}}(\mathcal{N}_m, \mathcal{P}_m). \tag{16}$$

denotes the average of the empirical AUC values from the $(t-\Delta)$th to the $(t-1)$th iterations. Because the value of $\widehat{\mathrm{AUC}}(\mathcal{N}_m, \mathcal{P}_m)$ fluctuates during training, the minimization operator is used to ensure that the value of β_t decreases monotonically. Note that if too difficult negative trials are selected with Algorithm 1 in the early stage of training, e.g., setting β_t to a small value such as 0.1, the training process may not converge. Therefore, quick shrinkage of the value of β_t should be avoided in the early stage of training. For this, we introduce a hyperparameter, i.e., Δ, to control the descent speed of β_t, where $\Delta \geq 1$.

Finally, the CBRW-BCE loss function is computed as

$$\begin{cases} \widehat{\mathcal{N}}_t = \mathrm{SSF}(\alpha_t, \beta_t; \theta, w, b), \\ \mathcal{L}_{\text{CBRW-BCE}} = \mathcal{L}_{\text{BRW-BCE}}(\widehat{\mathcal{N}}_t, \mathcal{P}_t; \theta, w, b). \end{cases} \tag{17}$$

According to (7) and (15), $\mathcal{L}_{\text{CBRW-BCE}}$ essentially maximizes the partial AUC as the value of β_t gradually decreases, which is illustrated in Fig. 4.

As seen in Algorithm 1, the algorithm consists of three steps. So, the complexity of the algorithm can be estimated accordingly. Specifically, the first step needs only one addition and two multiplications while the third step only needs to move $(\hat{\beta}_t - \hat{\alpha}_t + 1)$ numbers. The complexity of the second step depends on the sorting function. If, for example, the Bubble Sort algorithm is used, the time and space complexities are, respectively, $\mathcal{O}(I^2)$ and $\mathcal{O}(1)$. In practice, the best performance of CBRW-BCE can be reached with a not large batch size. So, the complexity of the algorithm is not high.

C. Score calibration

The discriminability and calibration of a speaker verification model are not closely related. The discriminability refers to the problem of learning the ranking of the similarity scores while the calibration refers to the ability of converting the absolute similarity scores to LLRs without changing their ranking order. Therefore, a loss function that is good for discrimination does not necessarily lead to good calibration and vice versa.

As shown in Fig. 1(b), an end-to-end calibration method trains a speaker verification model with a calibration sensitive loss function so that the model would directly output calibrated scores. Such a loss function should be able to optimize both the discrimination and calibration performance jointly. In practice, the logistic regression is a common choice for calibration [19]. In [19], it was implemented as an independent module from the deep learning based speaker verification model. Because the logistic regression is optimized with the BCE loss function, it should in principle be able to realize the end-to-end calibration. However, as described in Section I, BCE cannot achieve the best discriminability. In comparison, CBRW-BCE, which is a generalized form of BCE, is able to optimize both the discriminability and calibration performance jointly, which can be seen from (13).

The hyperparameter δ and the ranking between positive and negative trials in (10) are originally used for improving the discriminability of the model, which are not optimal for calibration. To achieve both good discriminability and calibration, CBRW-BCE first guides DNN to achieve good discriminant performance while leaving the scores weakly calibrated. Once the DNN training converges in terms of some evaluation metrics, e.g., EER or pAUC, CBRW-BCE then turns to focus on refining the model to output well-calibrated scores with the relevant parameters being set as $\delta = 0$, $\omega_j = \frac{1}{J}$, and $\omega_i = \frac{1}{\hat{I}}$ where $\hat{I}$ is the size of $\widehat{\mathcal{N}}_t$. Meanwhile, θ is fixed to avoid deterioration in discriminant performance. Note that the values of w and b in (3) do not affect the ranking order of the similarity scores, so these two parameters are optimized throughout the entire training process.

To leave this section, we summarize the CBRW-BCE based end-to-end training algorithm for speaker verification, which is given in Algorithm 2.

IV. Experimental setup

A. Datasets

1) Training data: All neural networks are trained on the development set of VoxCeleb2 [34], which consists of 1,092,009 utterances from 5994 speakers. The VoxCeleb2 is automatically collected from open-source media, and the speech segments are corrupted with real world noise. It is also multilingual with speech from speakers of 145 different nationalities.

In addition to training models on VoxCeleb2 directly, experiments are also carried out with data augmentation. Specifically, an online data augmentation [46] is applied to the development set of VoxCeleb2 with the MUSAN [47] and RIRs [48] datasets to increase the amount and diversity of the training data.

[2]One mini-batch of data consists of $2U$ utterances from U speakers, with 2 utterances per speaker. The $2U$ utterances are combined with each other, which generates $U(2U-1)$ trials including U positive trials and $2U(U-1)$ negative trials.

Algorithm 2: End-to-end training of CBRW-BCE.

Input: The training data $\mathcal{D}^{\mathrm{t}}$, and the validation data $\mathcal{D}^{\mathrm{v}}$;
Output: Optimized parameters θ^*, w^*, b^*
;
Initialization: $\alpha_0 = 0.0$, $\beta_0 = 1.0$, and $\delta = 2.0$;
$t \leftarrow 0$;
refine-falg $\leftarrow$ False;
while *the training loss does not converge* **do**
  Load a minibatch data from $\mathcal{D}^{\mathrm{t}}$ and propagate it forward the network as in Fig. 1(b) to get $\{\mathbf{x}_u\}_{u=1}^{2U}$;
  Construct pairwise trials[2]from $\{\mathbf{x}_u\}_{u=1}^{2U}$, and compute similarity scores according to (3) to get $\mathcal{N}_t$ and $\mathcal{P}_t$;
  $t \leftarrow t + 1$;
  if *refine-flag*== False **then**
    Compute the loss function $\mathcal{L}_{\text{CBRW-BCE}}$ according to (17);
    Back propagate $\mathcal{L}_{\text{CBRW-BCE}}$ to update θ, w and b;
    if $t \in \{\Delta, 2\Delta, 3\Delta, \cdots\}$ **then**
      Shrink β_t according to (15): $\beta_t \leftarrow \min(\beta_{t-1}, 1 - P_\Delta)$;
      Conduct validation on $\mathcal{D}^{\mathrm{v}}$ (in terms of EER) to judge whether to start refining, if yes: *refine-flag* $\leftarrow$ True;
    else
      $\beta_t \leftarrow \beta_{t-1}$;
    end
  else
    Compute loss function $\mathcal{L}_{\text{CBRW-BCE}}$ according to (17) with $\omega_j = \frac{1}{J}$, $\omega_i = \frac{1}{I}$, $\delta = 0$ and $\beta_t = 0.1$;
    Back propagate $\mathcal{L}_{\text{CBRW-BCE}}$ to update w and b;
  end
end
Result: θ^*, w^*, b^*

2) Evaluation data: Three lists of evaluation trials of the VoxCeleb1 dataset [42] are used for evaluation. As summarized in Table III, the three lists are: (i) the cleaned up version of the original verification test list, i.e., the VoxCeleb1, which consists of 37,611 trials from 40 speakers; (ii) the cleaned up extend list of VoxCeleb1-E that randomly samples 579,818 pairs from the entire VoxCeleb1 dataset covering 1251 speakers; and (iii) the challenging VoxCeleb1-H list, where 550,894 test pairs are drawn from identities with the same gender and nationality. Note that the list files can be downloaded from the VoxCeleb website[3].

Besides, the trained models are also evaluated with the Speakers in the Wild (SITW) [49] dataset. SITW is collected from open-source media. It contains 299 speakers. Each recording varies from 6 to 180 seconds. In this work, two official "trial-core-core" lists of the "dev" and "eval" parts are used, which include 338,226 trials and 721,788 trials respectively. For simplicity, we denote the two sets as Dev.Core and Eval.Core respectively.

3) Validation data: The validation list is constructed from the test part of VoxCeleb2, which consists of a total of 36,237 utterances from 118 speakers. 40,708 trials are randomly formed using the utterances from the 118 speakers, with up to 110 sentences per speaker. The validation list consists of 6,193 target trials and 34,515 non-target trials. During the validation process, only the first 3-second signal of every utterance is used to compute the similarity scores.

Note that there are no overlapped speakers among the training, evaluation and validation data.

[3]https://www.robots.ox.ac.uk/~vgg/data/voxceleb/vox1.html

TABLE III: Basic information of the evaluation data.

Dataset	# of speakers	# of utterances	# of pairs
VoxCeleb1 (cleaned)	40	4708	37,611
VoxCeleb1-E (cleaned)	1251	145,160	579,818
VoxCeleb1-H (cleaned)	1190	137,924	550,894

TABLE IV: Architecture of the TDNN. Here, the symbol T denotes the total frames of the input utterance [4]. The batch normalization and Relu activations are not shown in the table.

Layer	Layer context	Total context	Input $\times$ Output
Frame1	$[t-2, t+2]$	5	150×512
Frame2	$\{t-2, t, t+2\}$	9	1536×512
Frame3	$\{t-3, t, t+3\}$	15	1536×512
Frame4	$\{t\}$	15	512×512
Frame5	$\{t\}$	15	512×1500
Statistic pooling	$[0, T)$	T	$1500T \times 3000$
Segment6	$\{0\}$	T	3000×512
Segment7	$\{0\}$	T	512×512
Loss functions	—	—	—

B. Comparison among different loss functions

The hyperparameter δ of CBRW-BCE in all experiments is set to 2.0 unless otherwise stated. We compare CBRW-BCE with two classification-based loss functions and five metric-learning-based loss functions.They are listed as follows:

- **Standard Softmax loss** [3].
- **ArcSoftmax loss** [25] in which the scale and margin hyperparameters are set to 30 and 0.2, respectively [29].
- **Softmax-based generalized end-to-end loss (GE2E)** [28], [29].
- **Angular prototypical loss (Ang-Prototy)** [29].
- **Triplet loss** [33], detailed description of which is given in Appendix A.
- **Contrastive loss** [34], detailed description of which is given in Appendix A.
- **Binary cross entropy (BCE)** [17]. Based on the definition in (5), two versions are evaluated, i.e., 1) no hard negative mining is used, which is denoted as "BCE (w/o)", and 2) only 10% most large negative scores of $\mathcal{S}_t$ is selected to compute $\mathcal{L}_{\text{BCE}}$, i.e. 10% most difficult negative training trials of a batch are used at each training iteration, which is denoted as "BCE (w)".

C. Experimental settings

1) Neural networks: Three types of neural networks are used as the backbone networks in our experiments. The first

TABLE V: Architecture of the Fast ResNet-34. The symbol T denotes the total frames of the input utterance. The batch normalization and Relu activations are not shown in the table.

Layer name	Structure	Output size
Input	—	$40 \times T \times 1$
Conv2D-1	7×7, Stride=(2, 1)	$20 \times T \times 16$
SEBasicBlock-1	$\begin{bmatrix} 3\times3 & 16 \\ 3\times3 & 16 \end{bmatrix} \times 3$, Stride=(1,1)	$20 \times T \times 16$
SEBasicBlock-2	$\begin{bmatrix} 3\times3 & 32 \\ 3\times3 & 32 \end{bmatrix} \times 4$, Stride=(2,2)	$10 \times \frac{T}{2} \times 32$
SEBasicBlock-3	$\begin{bmatrix} 3\times3 & 64 \\ 3\times3 & 64 \end{bmatrix} \times 6$, Stride=(2,2)	$5 \times \frac{T}{4} \times 64$
SEBasicBlock-4	$\begin{bmatrix} 3\times3 & 128 \\ 3\times3 & 128 \end{bmatrix} \times 3$, Stride=(1,1)	$5 \times \frac{T}{4} \times 128$
Average	—	$1 \times \frac{T}{4} \times 128$
Self-attentive pooling	—	1×128
Dense layer	—	1×512
Loss functions	—	—

one is a time delay neural network (TDNN) [4], which is summarized in Table IV. The TDNN consists of five frame level layers, a statistic pooling layer and two segment level layers. The network without the loss function layer has 4.2 million parameters. The second network is a modified ResNet-34, called the Fast ResNet-34 [29], which is summarized in Table V. The basic residual blocks are squeeze-and-excitation blocks (SEBasicBlock). This network without the loss function layer has 1.4 million parameters. The third one is an emphasized channel attention, propagation and aggregation TDNN (ECAPA-TDNN) [50]. It is an enhanced version of TDNN by introducing one-dimensional Res2Net modules, squeeze-and-excitation blocks, multi-layer feature aggregation, and a channel- and context-dependent statistic pooling layer. The work in [50] presents two setups to the convolutional frame layers of ECAPA-TDNN, which are a light setup of 512-channels with 6.2 million parameters and a larger setup of 1024-channels with 14.7 parameters respectively.

2) Inputs of networks: For training TDNN, training utterances are partitioned into small frames with a frame length of 25 ms and a frame shift of 10 ms. 30-dimensional Mel-frequency cepstrum coefficients (MFCCs) are then extracted from every frame followed by cepstral mean normalization over a sliding window of 3 seconds. A 2-second long speech segment is then randomly selected from every utterance, resulting 200 consecutive frames of normalized MFCC vectors, which are used as the network inputs.

Similarly, the input features are extracted for Fast ResNet-34 and ECAPA-TDNN. The difference is that for Fast ResNet-34, 40-dimensional log Mel-filterbank energies are extracted every frame. Then, instant mean and variance normalization is applied [29]. While for ECAPA-TDNN, 80-dimensional log Mel-filterbank energies are extracted every frame followed by cepstral mean normalization [50].

Note that no voice activity detection is used since the speech signals in both the training and test datasets consist of not much silence.

3) Training and evaluation details: The Adam optimizer is used. For the TDNN, Fast ResNet-34 and ECAPA-TDNN of 512-channels, the learning rate is initialized to 0.001 and it decreases by 5% every 5 epochs. For the ECAPA-TDNN of 1024-channels, the learning rate is initialized to 0.001 and decreases by 3% per epoch, and a weight decay of $2\mathrm{e}^{-5}$ is applied. For the classification-based loss functions, a fixed batch size of 256 is used. For the metric-learning-based loss functions, we use a batch size of 200 speakers with q utterances per speaker, where $q = 3$ for GE2E and $q = 2$ for the others. For TDNN and Fast ResNet-34, we iterate at most 100 epochs and 150 epochs for the classification-based and metric-learning-based loss functions respectively. For the ECAPA-TDNN of 512-channels, 60 epochs are iterated for all the studied loss functions. For the ECAPA-TDNN of 1024-channels, we iterate 80 and 120 epochs, respectively, for the classification-based and metric-learning-based loss functions. Finally, the best model in terms of EER for every method is used on the validation set for evaluation.

For the evaluation on the VoxCeleb1 dataset, we follow the work in [29] and sample ten segments at regular intervals from every test utterance and each segment is 4-second long (note that if it is shorter than 4 seconds, the utterance is padded to be 4-second long with a copy of samples in the front). Ten embedding vectors are extracted subsequently for every test utterance. We fist use the cosine similarity scoring function to evaluate all the studied loss functions. For every evaluation trial, which consists of two test utterances, the cosine similarity scores between all possible combinations ($10 \times 10 = 100$) of the ten embedding vectors of the two utterances are computed. The mean of the scores is computed and used as the final score of the trial[4] In addition, a stage-wise linear discriminant analysis (LDA) with probabilistic linear discriminant analysis (PLDA) back-end is also evaluated [4], where ten embedding vectors of each utterance are first averaged, and the PLDA score is then computed over the averaged embedding vectors.

For evaluation on the SITW dataset, we follow the work in [31] and extract an embedding vector by directly using the entire speech signal of every test utterance. For the end-to-end evaluation, a cosine similarity scoring back-end is used for all the studied loss functions. Besides, a stage-wise LDA with PLDA back-end is also evaluated.

The 512-dimensional embedding vector of TDNN is the output of the "segment7" in Table IV. The 512-dimensional embedding vector of the Fast ResNet-34 is the output of the "Dense layer" in Table V. The 192-dimensional embedding vector of ECAPA-TDNN is the output of its last batch-normalization layer [50].

D. Evaluation metrics

The following metrics are used to evaluate the discriminant performance: EER, partial AUC (pAUC) with $\alpha = 0$ and $\beta = 0.05$ [14], and normalized minimum detection cost function (minDCF) with $P_{\mathrm{tar}} = 0.05$ and $C_{\mathrm{miss}} = C_{\mathrm{fa}} = 1$ [52], and the detection error tradeoff (DET) curves.

[4]For the calibration experiments in Section V-D, the linearly transformed cosine similarity in (3) is used.

TABLE VI: Results on the VoxCeleb1. The terms "(w)" and "(w/o)" denote the BCE with and without the hard negative mining respectively. The term "[aug]" denotes that the model is trained with augmented data. Δ is the hyperparameter of the curriculum learning.

		VoxCeleb1			**VoxCeleb1-E**			**VoxCeleb1-H**		
Networks	Loss functions	EER [%]	minDCF	pAUC [%]	EER [%]	minDCF	pAUC [%]	EER [%]	minDCF	pAUC [%]
TDNN	Softmax	4.81	0.339	87.07	5.20	0.338	86.95	8.86	0.491	75.07
	ArcSoftmax	3.02	**0.220**	93.98	**2.86**	**0.193**	**94.91**	**5.05**	**0.301**	**88.59**
	GE2E	3.18	0.247	92.97	3.14	0.234	93.36	5.90	0.386	84.02
	Ang-Prototy	3.08	0.238	93.43	3.06	0.229	93.68	5.78	0.367	84.95
	Triplet (semi-hard)	3.27	0.253	92.52	3.46	0.258	92.19	6.43	0.422	81.81
	Contrastive	4.27	0.337	88.04	4.28	0.321	88.85	8.21	0.532	74.02
	BCE (w/o)	4.05	0.318	89.08	4.03	0.313	89.51	7.80	0.513	75.36
	BCE (w)	3.68	0.292	90.70	3.50	0.263	92.06	6.65	0.433	80.86
	CBRW-BCE ($\Delta = 100$)	**3.01**	0.221	**94.05**	3.05	0.217	94.04	5.43	0.343	86.47
Fast ResNet-34	Softmax	5.10	0.343	87.14	5.16	0.341	86.79	8.87	0.496	75.05
	ArcSoftmax	2.18	0.155	96.73	2.42	0.160	96.14	4.29	**0.257**	91.05
	GE2E	2.82	0.235	93.83	3.21	0.247	92.85	6.23	0.420	81.99
	Ang-Prototy	2.33	0.197	95.38	2.55	0.190	95.41	4.91	0.323	87.98
	Triplet (semi-hard)	3.03	0.251	93.07	3.44	0.260	92.13	6.83	0.450	79.83
	Contrastive	2.93	0.257	93.24	3.28	0.242	92.97	6.35	0.411	82.36
	BCE (w/o)	4.16	0.309	89.51	4.29	0.337	88.24	8.16	0.540	73.52
	BCE (w)	2.53	0.214	94.90	2.82	0.211	94.43	5.44	0.356	85.84
	CBRW-BCE ($\Delta = 100$)	2.17	0.172	96.27	2.32	0.168	96.15	4.37	0.285	90.23
	CBRW-BCE ($\Delta = 8$)	**1.94**	**0.150**	**96.96**	**2.15**	**0.157**	**96.59**	**4.07**	0.268	**91.12**
ECAPA-TDNN 512-Channels	ArcSoftmax	1.51	0.113	98.18	1.81	0.116	97.71	3.21	**0.195**	94.33
	Ang-Prototy	1.56	0.127	97.93	1.91	0.137	97.27	3.68	0.235	92.67
	Triplet (semi-hard)	1.95	0.151	97.14	2.21	0.161	96.41	4.41	0.286	89.93
	BCE (w)	1.87	0.142	97.12	2.12	0.157	96.65	4.13	0.273	90.84
	CBRW-BCE ($\Delta = 100$)	**1.48**	**0.094**	**98.45**	**1.61**	**0.112**	**97.98**	**3.06**	0.196	**94.51**
	ArcSoftmax [aug]	1.44	0.104	98.37	1.61	0.105	98.09	2.98	0.181	94.97
	CBRW-BCE ($\Delta = 100$) **[aug]**	1.60	0.117	98.00	1.72	0.123	97.73	3.26	0.206	93.94
	CBRW-BCE ($\Delta = 8$) **[aug]**	1.43	0.117	98.27	1.64	0.114	97.96	3.14	0.200	94.32
ECAPA-TDNN 1024-Channels	ArcSoftmax [aug]	1.21	0.094	98.80	1.36	**0.089**	98.55	**2.57**	**0.159**	**96.00**
	Ang-Prototy [aug]	1.19	0.113	98.61	1.50	0.106	98.21	3.01	0.193	94.66
	Triplet (semi-hard) [aug]	2.30	0.185	95.85	2.46	0.179	95.67	4.83	0.313	88.53
	BCE (w) [aug]	1.43	0.127	98.19	1.77	0.130	97.53	3.60	0.237	92.75
	CBRW-BCE ($\Delta = 100$) **[aug]**	**1.10**	**0.088**	**98.93**	1.40	0.098	98.42	2.72	0.170	95.49
	CBRW-BCE ($\Delta = 8$) **[aug]**	1.16	0.090	98.82	**1.33**	0.093	**98.56**	2.64	0.171	95.69
	CBRW-BCE ($\Delta = 100$)	1.14	0.087	98.90	1.40	0.098	98.42	2.59	0.166	95.84

For the calibration performance, we report the normalized actual detection cost function (actDCF) with $P_{\rm tar} = 0.05$ and $C_{\rm miss} = C_{\rm fa} = 1$. The threshold ξ in (1) is computed with the Bayes decision theory [19]. The metric $C_{\rm llr}$ measures how well the scores represent the LLRs [19]. Because it is affected by both the discrimination and calibration performance, we evaluate the effect of miss-calibration by $\Delta C_{\rm llr} = {\rm act}C_{\rm llr} - {\rm min}C_{\rm llr}$ [39]. The normalized Bayes-error-rate (BER) curves [19] are also shown.

V. Results and analysis

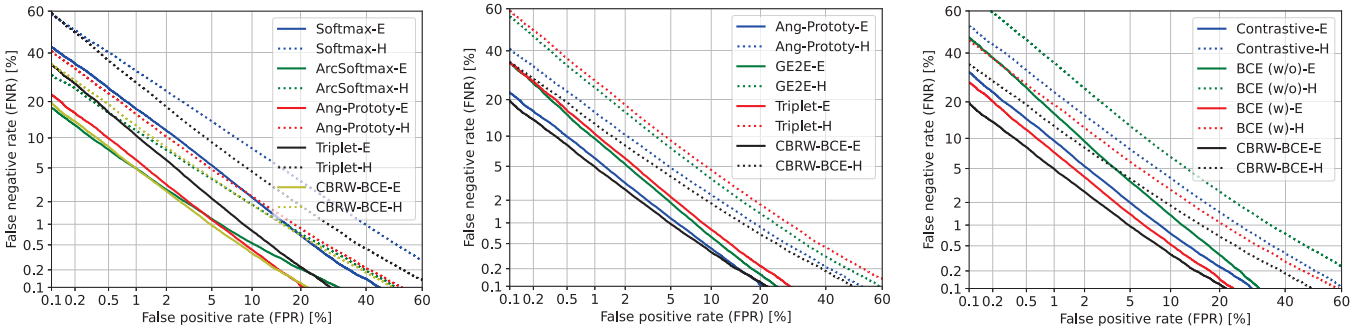
A. Discriminant performance of CBRW-BCE with the cosine similarity scoring back-end

Table VI and Fig. 5 present the discriminant performance of CBRW-BCE and the compared methods on the VoxCeleb1 dataset. From Table VI, one can see that CBRW-BCE achieves EERs comparable AcrSoftmax and performs better than the other compared loss functions. The performance advantage can also be seen from the DET curves in Fig. 5(a). Although ArcSoftmax is slightly better than CBRW-BCE in terms of minDCF and pAUC, it is seen from Fig. 5(a) that the DET curve of CBRW-BCE is lower than that of ArcSoftmax when the false positive rate (FPR) is larger than 2%.

Comparing all the studied metric-learning-based loss functions, one can see that CBRW-BCE is superior to the others in all evaluations. This indicate that CBRW-BCE is able to serve as a state-of-the-art metric-learning-based loss function for the end-to-end speaker verification. The DET curve comparison in Fig. 5(b) further demonstrates the advantage of CBRW-BCE. Particularly, if comparing CBRW-BCE with the pairwise loss functions that have similar properties as shown in Table I, we see that CBRW-BCE yields significantly better performance. Similar observations can be made from Fig. 5(c). Finally, between the results of "BCE (w/o)" and "BCE (w)" in Table VI, one can see that the fixed hard negative mining is able to improve the discriminant performance significantly.

Table VII lists the results on the SITW dataset. It is seen that CBRW-BCE achieves better discriminant performance than all the compared loss functions.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TASLP.2022.3161155, IEEE/ACM Transactions on Audio, Speech, and Language Processing



(a) Comparison of classification and metric learning. (b) Comparison of various metric leaning losses. (c) Comparison of pairwise metric learning loss.

Fig. 5: DET curves of the Fast ResNet-34 trained with various loss functions, where the test sets are the VoxCeleb1-E (marked by "-E") and VoxCeleb1-H (marked by "-H") respectively.

TABLE VII: Results on the SITW dataset.

	Datasets	Loss functions	EER [%]	minDCF	pAUC [%]
Fast ResNet-34	Dev.Core	ArcSoftmax	4.08	0.219	92.37
		Ang-Prototy	3.47	0.241	92.70
		Triplet (semi-hard)	4.47	0.289	89.63
		BCE (w)	3.31	0.253	92.74
		CBRW-BCE ($\Delta = 100$)	3.28	0.223	93.46
		CBRW-BCE ($\Delta = 8$)	**3.08**	**0.211**	**94.12**
	Eval.Core	ArcSoftmax	4.18	0.253	91.56
		Ang-Prototy	3.76	0.276	90.99
		Triplet (semi-hard)	4.67	0.322	87.96
		BCE (w)	3.91	0.300	90.38
		CBRW-BCE ($\Delta = 100$)	3.72	0.259	91.81
		CBRW-BCE ($\Delta = 8$)	**3.20**	**0.242**	**93.08**
ECAPA-TDNN, 512-Channels	Dev.Core	ArcSoftmax	3.29	0.169	94.70
		Ang-Prototy	2.70	0.181	95.32
		Triplet (semi-hard)	3.00	0.191	94.69
		BCE (w)	2.46	0.205	95.28
		CBRW-BCE ($\Delta = 100$)	**2.31**	**0.152**	**96.48**
	Eval.Core	ArcSoftmax	3.15	0.183	94.58
		Ang-Prototy	3.01	0.200	94.43
		Triplet (semi-hard)	3.34	0.215	93.55
		BCE (w)	3.31	0.230	93.26
		CBRW-BCE ($\Delta = 100$)	**2.84**	**0.181**	**95.19**

B. Discriminant performance of CBRW-BCE with the stage-wise PLDA back-end

Although the focus of this paper is on the end-to-end speaker verification, we also evaluate the discriminant performance with the stage-wise PLDA back-end. Before computing the PLDA scores, LDA is applied to reduce the dimensionality of the embedding vectors from 512 to 400 for the Fast ResNet-34, and from 192 to 180 for ECAPA-TDNN. The results are given in Table VIII. Observations similar to those from the end-to-end evaluation can be seen. Besides, comparing Tables VI, VII and VIII, we see that the cosine similarity scoring function achieves better performance than the PLDA back-end in our experiments.

C. Performance study of different components of CBRW-BCE

This subsection investigates the impact of each component on the performance of CBRW-BCE. The embedding extractor is the Fast ResNet-34. The batch size is set to 200×2 unless otherwise stated. No data augmentation is used.

1) Effects of the bipartite-ranking-based weighting and curriculum learning: We study the effects of the bipartite-ranking-based weighting and curriculum learning by comparing the following three loss functions, i.e., $\mathcal{L}_{\text{BCE}}$ given in (5), $\mathcal{L}_{\text{BRW-BCE}}$ defined in (13), and $\mathcal{L}_{\text{CBRW-BCE}}$ defined in (17). Experimental results are presented in Table IX, where the models are trained with $\delta = 2$ and $\Delta = 8$. It is seen from the results that $\mathcal{L}_{\text{BRW-BCE}}$ performs consistently better than $\mathcal{L}_{\text{BCE}}$ (w/o) on all the evaluated datasets, which demonstrates the advantage of the proposed bipartite-ranking-based weighting over the traditional BCE loss function. Moreover, $\mathcal{L}_{\text{CBRW-BCE}}$ achieves approximately 40% relative EER reduction over $\mathcal{L}_{\text{BRW-BCE}}$, which demonstrates the importance of using the curriculum learning.

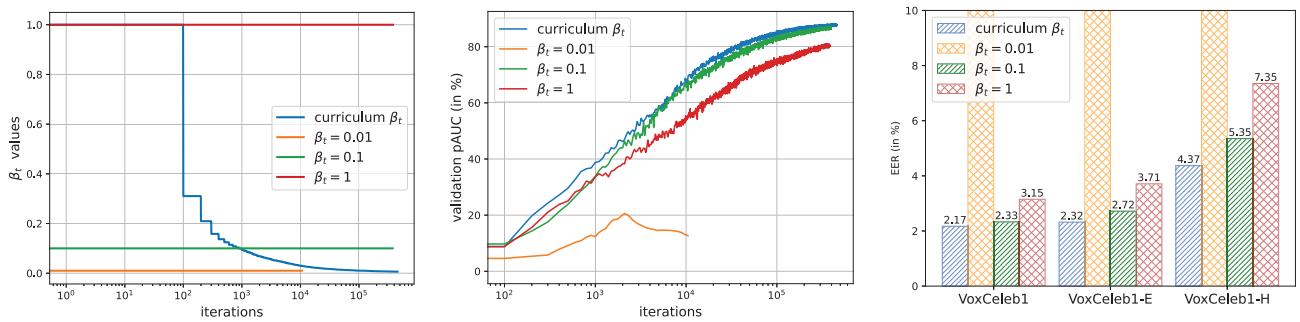
2) Comparison between the curriculum learning and the fixed hard negative mining: The value of the parameter β_t in the curriculum hard negative mining determines the degree of discriminant difficulty of the hard negative trials. It is learned asymptotically from 1 to a small value close to 0. To study the effectiveness of this curriculum learning method, we compare it with a method with a fixed β_t value selected from $\{1, 0.1, 0.01\}$. The results are presented in Fig. 6. From Fig. 6(a), one can see that in the curriculum learning, the value of β_t gradually decreases from 1 to around 0.01 as the training iteration progresses, which is consistent with our original motivation. It is seen from Fig. 6(b) that the pAUC produced by the curriculum learning over different iterations reflects the convergence status of the DNN model training. In contrast, the DNN does not converge when β_t is fixed to a small value, e.g., 0.01, which is due to the use of too difficult training samples in the early training stage as a result of a too small, fixed value of β_t. One may choose a large, fixed value for β_t, e.g., 1, but the performance is then not good, which is also evident from the results.

One may consider to treat β_t as a hyperparameter and tune it on the validation set. So, we carry out corresponding experiments and obtain the best value, which is $\beta_t = 0.1$. From the results in Fig. 6(b), it is seen that $\beta_t = 0.1$ yields a convergence curve similar to that with the curriculum learning, but the pAUC of the resulting model is worse (though slightly)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TASLP.2022.3161155, IEEE/ACM Transactions on Audio, Speech, and Language Processing

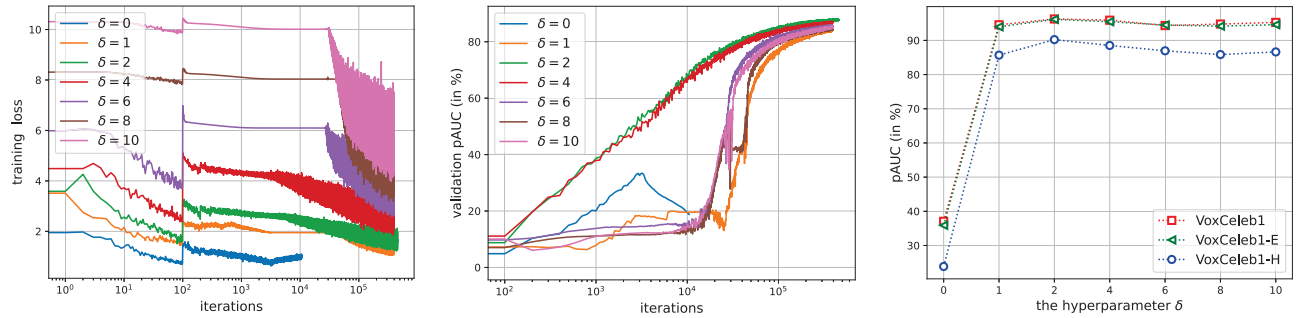
TABLE VIII: Discriminant performance with the stage-wise LDA with PLDA back-end.

	Objectives	VoxCeleb1			VoxCeleb1-H			SITW Eval.Core			SITW Dev.Core		
		EER [%]	minDCF	pAUC [%]	EER [%]	minDCF	pAUC [%]	EER [%]	minDCF	pAUC [%]	EER [%]	minDCF	pAUC [%]
Fast ResNet-34	Softmax	10.15	0.663	62.88	18.80	0.899	36.52	5.49	0.364	85.25	5.04	0.321	87.25
	ArcSoftmax	**3.44**	**0.225**	**93.06**	**5.62**	**0.334**	**86.66**	4.79	0.290	89.40	4.77	0.261	90.05
	Ang-Prototy	3.64	0.257	91.91	6.45	0.386	83.51	5.17	0.301	88.11	4.47	0.267	90.21
	Triplet (semi-hard)	4.41	0.310	89.23	8.32	0.496	75.87	6.24	0.374	83.76	6.20	0.326	85.98
	BCE (w)	4.31	0.282	90.37	7.16	0.429	80.51	4.76	0.316	88.35	4.37	0.276	90.10
	CBRW-BCE ($\Delta = 100$)	3.67	0.237	92.65	6.06	0.363	84.92	5.03	0.292	88.72	**4.24**	0.261	90.73
	CBRW-BCE ($\Delta = 8$)	3.46	0.240	92.59	5.99	0.352	85.52	**4.58**	**0.283**	**89.81**	4.28	**0.254**	**91.05**
ECAPA-TDNN 512-Channels	ArcSoftmax	2.31	0.163	96.30	**4.14**	**0.242**	**91.77**	3.26	0.215	93.75	3.20	0.201	94.28
	Ang-Prototy	2.49	0.177	95.83	4.78	0.279	89.83	3.20	0.211	94.11	3.47	0.206	93.77
	Triplet (semi-hard)	3.37	0.219	93.59	5.85	0.344	86.12	4.04	0.257	91.65	3.48	0.221	93.08
	BCE (w)	2.83	0.201	94.60	5.30	0.320	87.76	3.48	0.245	92.60	3.17	0.219	93.70
	CBRW-BCE ($\Delta = 100$)	**2.26**	**0.159**	**96.40**	4.35	0.257	91.12	**3.20**	**0.207**	**94.17**	**2.70**	**0.182**	**95.48**



(a) The values of β_t on different iterations. (b) The validation pAUCs on different iterations. (c) The EERs of test sets with different β_t.

Fig. 6: Effects of the learnable parameter β_t on the discriminant performance of CBRW-BCE, given $\delta = 2$ and $\Delta = 100$.



(a) The decline of the loss on different iterations. (b) The validation pAUCs on different iterations. (c) The pAUCs of test sets with different δ.

Fig. 7: Effects of the manually-tunable hyperparameter δ on the discriminant performance of CBRW-BCE, given $\Delta = 100$.

on the validation set. The above advantage stays the same over the test sets as well, as seen in Fig. 6(c). Moreover, the curriculum hard negative mining method avoids the tedious hyperparameter tuning process.

3) Effect of the hyperparameter Δ on curriculum learning: As seen from the previous experiments, the curriculum leaning is an important component that affects the performance of CBRW-BCE. The curriculum leaning has a tunable parameter, i.e., Δ, which may affect the performance. To study the effect of the parameter, we set it from 2 to 512. The results on the VoxCeleb1 test sets are shown in Table X, where the models are trained with $\delta = 2$. It is seen from that CBRW-BCE is insensitive to Δ though its performance drops slightly as the value of Δ increases. Theoretically, if the value of Δ is larger than the number of the training iterations, $\mathcal{L}_{\text{CBRW-BCE}}$ becomes $\mathcal{L}_{\text{BRW-BCE}}$. To avoid this to happen, we should set Δ to a small value, e.g., 8.

4) Effect of the hyperparameter δ on performance: There is one hyperparamenter, i.e., δ, in CBRW-BCE that can control the within-speaker variances. To study the impact of this parameter on performance, we vary the value of δ from 0 to 10 with a step size of 1 or 2 in training. Figure 7 plots the performance curves with different values of δ. Specifically, Figure 7(a) plots the curves of the training loss with respect to the training iterations. It is seen that the curves have sudden changes at the 10^2th iteration, which is caused by the decrease

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TASLP.2022.3161155, IEEE/ACM Transactions on Audio, Speech, and Language Processing

TABLE IX: Effects of the bipartite-ranking-based weighting and curriculum learning on the discriminant performance, where the "(w/o)" denotes without hard negative mining.

Test Data	Metrics	Different loss functions		
		$\mathcal{L}_{\text{BCE}}$ (w/o)	$\mathcal{L}_{\text{BRW-BCE}}$	$\mathcal{L}_{\text{CBRW-BCE}}$
VoxCeleb1	EER [%]	4.16	3.15	1.94
	minDCF	0.309	0.282	0.150
	pAUC [%]	89.51	91.97	96.96
VoxCeleb1-E	EER [%]	4.29	3.71	2.15
	minDCF	0.337	0.287	0.157
	pAUC [%]	88.24	90.88	96.59
VoxCeleb1-H	EER [%]	8.16	7.35	4.07
	minDCF	0.540	0.484	0.268
	pAUC [%]	73.52	77.64	91.12

TABLE X: Performance versus Δ with the curriculum learning

Test Data	Metrics	Different Δ				
		$\Delta = 2$	$\Delta = 8$	$\Delta = 32$	$\Delta = 128$	$\Delta = 512$
VoxCeleb1	EER [%]	2.22	1.94	1.87	2.07	2.29
	minDCF	0.177	0.150	0.151	0.162	0.168
	pAUC [%]	96.27	96.96	97.08	96.75	96.25
VoxCeleb1-E	EER [%]	2.29	2.15	2.26	2.38	2.42
	minDCF	0.167	0.157	0.161	0.172	0.176
	pAUC [%]	96.23	96.59	96.36	95.97	95.86
VoxCeleb1-H	EER [%]	4.31	4.07	4.17	4.46	4.54
	minDCF	0.283	0.268	0.270	0.287	0.294
	pAUC [%]	90.29	91.12	90.87	89.94	89.60

TABLE XI: Performance versus the batch size

Test Data	Metrics	Different Batch-Sizes			
		32×2	64×2	128×2	256×2
VoxCeleb1	EER [%]	2.95	2.35	2.21	1.98
	minDCF	0.270	0.205	0.166	0.164
	pAUC [%]	92.40	95.51	96.40	96.78
VoxCeleb1-E	EER [%]	3.19	2.65	2.29	2.36
	minDCF	0.262	0.199	0.170	0.169
	pAUC [%]	92.26	94.98	96.15	96.03
VoxCeleb1-H	EER [%]	7.10	5.26	4.34	4.44
	minDCF	0.561	0.355	0.286	0.287
	pAUC [%]	73.67	86.22	90.09	89.93

in value of the curriculum parameter β_t as shown in Fig. 6(a).

Figure 7(b) plots the pAUC curves with respect to the training iterations on the validation set. Comparing the first two sub-figures of Fig. 7, one can see that, when $\delta = 2$ and 4, the DNN model is optimized successfully while some problems appear for the other studied values of δ. Specifically, when $\delta = 0$, the training loss of the DNN model increase unexpectedly when the training iterations are between 10^3 and 10^4, which leads to a decrease in the pAUC curve on the validation set. If δ is set to a relative small value (i.e. 1) or large values (i.e. $6, 8, 10$), the training loss decreases slowly during the 10^2th and 10^4th iterations; after that, the training process converges quickly. This may be caused by the bad local minimum of the loss function.

Figure 7(c) shows the EER results of the proposed method with different values of δ.

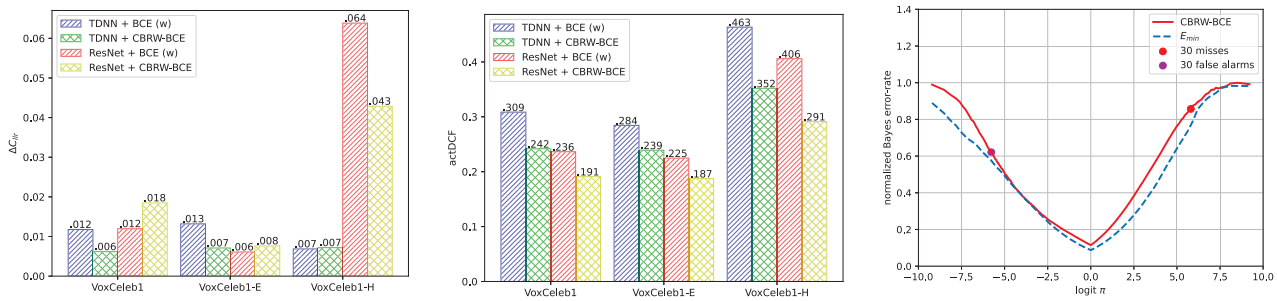
5) Effect of the batch size on performance: For every training iteration, the classification-based loss functions, e.g., ArcSoftmax, conduct optimization with respect to all the training speakers. In contrast, the metric-learning-based loss functions only conduct local optimization with respect to a batch of randomly selected speakers. Therefore, the batch size is an important parameter that affect significantly the performance of the metric-learning-based loss functions. To validate the effect of the batch size on CBRW-BCE, we set $\delta = 2$, $\Delta = 100$, and vary the batch size from 32×2 to 256×2 and evaluate the performance. The results are presented in Table XI. It is seen from Table XI that that there is a certain performance drop with the decrease of the batch size. However, CBRW-BCE can still produce reasonably good model even when the batch size is reduced to 32×2. Note that a larger batch size does not always yield a better model since the increase of the batch size will reduce the number of training iterations for every epoch. In practice, batch size of around 128×2 is a reasonable choice.

D. Calibration performance of CBRW-BCE

This subsection investigates the calibration performance of CBRW-BCE. The linearly transformed cosine similarity scores are computed according to (3). As shown in Table I, except for the BCE, most compared loss functions do not calibrate models directly. Therefore, the "BCE (w)" is used as the baseline to compare. Figure 8 plots the ΔC_{llr} and actDCF results of the compared methods. It is seen that both CBRW-BCE and "BCE (w)" are generally well calibrated with small ΔC_{llr} values. It is also seen from Fig. 8(b) that CBRW-BCE obtains consistently better actDCF than "BCE (w)" in all the test scenarios. This is because that actDCF depends not only on the calibration performance but also on the discriminant performance. Figure 8(c) plots the normalized BER curve of CBRW-BCE, which is trained with Fast ResNet-34 and tested on the VoxCeleb1-H. The results demonstrate the great calibration performance of CBRW-BCE as its BER curve is very close to the ideal reference curve (the dashed line).

We also compare CBRW-BCE with a general normal-distribution-based (ND) calibration method, which has a closed-form solution and is presented in APPENDIX B. In practice, the normal distribution assumption may not always be appropriate for the cosine similarity scores as some loss functions, e.g., ArcSoftmax, add a margin to ensure the learned embedding vectors have small within-speaker variance, which leads to some skewness in the probability density of the scores. Consequently, we also modify the ND calibration baseline to a skew-normal-distribution-based (SND) calibration baseline, which is given in APPENDIX B. To verify the effectiveness of calibration, we conducted a comparison study between the SND calibration and the ND calibration on simulated similarity scores and the results are presented in Fig. 9 where the first sub-figure illustrates the distributions and the second one plots the normalized BER curves. The statistics of ND and SND are directly estimated from the simulated scores. It is seen from the results that the modified SND achieves good calibration performance while the normal-distribution-based method does not produce good calibration results.

Table XII lists the end-to-end calibration results of CBRE-BCE, the calibration results of ND and SND based on the

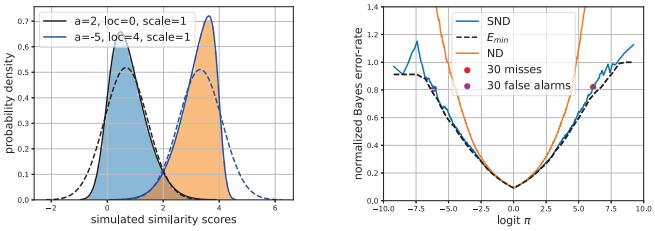


(a) The ΔC_{llr} on different test sets

(b) The actDCF on different test sets

(c) The BER curve on the VoxCeleb1-H test set.

Fig. 8: Results of the calibration performance of CBRW-BCE with $\delta = 2$ and $\Delta = 100$, and BCE (w). The "ResNet" denotes the Fast ResNet-34.



(a) Simulated skew normal scores. The terms "a", "loc", and "scale" denote the shape, location, and scale parameters of the skew normal distributions.

(b) Normalized BER curves of the simulated scores, where $\text{logit}\pi = \log\frac{\pi}{1-\pi}$, and $\pi \in (0.01, 0.99)$ denotes the prior of positive scores.

Fig. 9: A study of the skew normal distribution based calibration method on simulated skew normal scores.

TABLE XII: Calibration performance comparison between the CBRW-BCE with independent calibration methods. The "-ND" and "-SND" denote the normal-distribution-based and skew-normal-distribution-based methods, respectively.

		VoxCeleb1-E		**VoxCeleb1-H**	
Networks	Methods	ΔC$_{\text{llr}}$	actDCF	ΔC$_{\text{llr}}$	actDCF
TDNN	ArcSoftmax-ND	0.0109	0.236	0.0978	0.671
	ArcSoftmax-SND	0.0095	**0.209**	0.0755	0.539
	Ang-Prototy-ND	0.0076	0.251	0.1036	0.681
	Ang-Prototy-SND	0.0088	0.230	0.0924	0.522
	CBRW-BCE	**0.007**	0.239	**0.007**	**0.352**
Fast ResNet-34	ArcSoftmax-ND	**0.0051**	0.175	0.0682	0.485
	ArcSoftmax-SND	0.0082	**0.163**	0.0650	0.390
	Ang-Prototy-ND	0.0088	0.210	0.1072	0.626
	Ang-Prototy-SND	0.0128	0.191	0.1095	0.455
	CBRW-BCE	0.008	0.187	**0.043**	**0.291**

models trained by ArcSoftmax and Ang-Prototy with a cosine similarity back-end. The statistics of the ND and SND methods are estimated from the scores of the validation trials, and the embedding vectors of the validation utterances are extracted on a 4-second-segment basis so as to be consistent with the test segments. Note that if an utterance is shorter than 4 seconds, it is padded to 4-second long similar to what was done in the previous experiments. Comparing to the results of ND and SND, one can see that although both get small ΔC_{llr} values, the modified SND method achieves consistently better actDCF than the ND method, which proves the effectiveness of the modification. Furthermore, on the VoxCeleb1-H test set, the CBRW-BCE achieves considerable improvement in calibration performance as compared to the ND and SND methods. On the VoxCeleb1-E test set, the performance of CBRW-BCE is close to the best baseline though it is slightly worse. These results show that the model trained by CBRW-BCE is well calibrated.

VI. Conclusion and Future Work

In this paper, we proposed a loss function called CBRW-BCE with a method to train DNN models based on the bipartite ranking and curriculum learning for end-to-end speaker verification. Experimental results showed that the speaker verification model trained with the proposed loss function and training method improves both the discriminant and calibration performance. Specifically, CBRW-BCE achieved the state-of-the-art discriminant performance, which is comparable to that of the classification-based ArcSoftmax loss and better than the other studied methods including the Softmax, triplet, and contrastive loss functions. Regarding the calibration performance, CBRW-BCE demonstrated better performance than the end-to-end calibration baseline BCE, and the normal-distribution and skew-normal-distribution based calibration methods.

While the proposed method has demonstrated great potential, efforts are in progress to investigate why the training process of CBRW-BCE becomes unstable if the margin hyperparameter δ is set to 0. In addition, how to reduce the complexity of CBRW-BCE is worth of a further study.

Appendix A

The triplet loss and contrastive loss are summarized as follows. The triplet loss is computed from the training set $\mathcal{T}_t = \{(\mathbf{x}_n^{\text{a}}, \mathbf{x}_n^{\text{p}}, \mathbf{x}_n^{\text{n}}) | n = 1, 2, \cdots, N\}$ as

$$\mathcal{L}_{\text{trip}} = \frac{1}{N}\sum_{n=1}^{N} \max(0, \|\mathbf{x}_n^{\text{a}} - \mathbf{x}_n^{\text{p}}\|^2 - \|\mathbf{x}_n^{\text{a}} - \mathbf{x}_n^{\text{n}}\|^2 + m), \quad (18)$$

where $\mathbf{x}_n^{\text{a}}$ is an anchor embedding vector, $\mathbf{x}_n^{\text{p}}$ is a positive embedding vector from the same speaker as $\mathbf{x}_n^{\text{a}}$, $\mathbf{x}_n^{\text{n}}$ is a

negative embedding vector from a different speaker as $\mathbf{x}_n^{\mathrm{a}}$, and $\|\cdot\|$ denotes the ℓ_2 norm. All the embedding vectors are ℓ_2-normalized [12] before feeding into the $\mathcal{L}_{\mathrm{trip}}$, and m is set to 0.2 [29]. We employ the commonly used semi-hard negative mining method [53] for the triplet loss.

The contrastive loss is computed on the same training set $\mathcal{T}_t = \{(\mathbf{x}_n^1, \mathbf{x}_n^2, l_n)|n = 1, 2, \cdots, N\}$ as CBRW-BCE according to

$$\mathcal{L}_{\mathrm{cont}} = \sum_{n=1}^{N} \Big[l_n d_n^2 + (1 - l_n)\max(\rho - d_n, 0)^2 \Big], \qquad (19)$$

where $d_n = \|\mathbf{x}_n^1 - \mathbf{x}_n^2\|$. All the embedding vectors are ℓ_2-normalized [12] before feeding into $\mathcal{L}_{\mathrm{cont}}$, and ρ is set to 1. For the negative trials, we select the most difficult 10% samples from $\mathcal{T}_t$ for training at every mini-batch.

In (18) and (19), because we apply ℓ_2-normalization to all the embedding vectors before feeding them into the loss functions, the square of the ℓ_2 distance of two vectors is equivalent to their cosine similarity. Specifically, assuming that $\mathbf{x}$ and $\mathbf{y}$ are two ℓ_2-normalized vectors, i.e. $\|\mathbf{x}\| = \|\mathbf{y}\| = 1$, we have:

$$\|\mathbf{x} - \mathbf{y}\|^2 = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - 2\mathbf{x}^{\mathrm{T}}\mathbf{y} = 2 - 2\frac{\mathbf{x}^{\mathrm{T}}\mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} \qquad (20)$$

Therefore, the cosine similarity scoring function is also a reasonable back-end for the models trained with (18) and (19).

APPENDIX B

The work in [45] compared various calibration methods, including linear/non-linear and generative/discriminative ones. Among those, the Gaussian distribution based one can meet most requirements as long as the mean and variance of the Gaussian distribution are estimated correctly. Assume that the score s obeys Gaussian distributions, i.e., $s \sim N(\mu_0, \sigma_0^2)$ for H_0 and $s \sim N(\mu_1, \sigma_1^2)$ for H_1. The LLR can then be computed as,

$$\begin{aligned} \mathrm{LLR} &= \log\left[\frac{P(s|H_0)}{P(s|H_1)}\right] \\ &= \frac{1}{2}\log\frac{\sigma_1^2}{\sigma_0^2} + \frac{1}{2}\Big[\frac{(s-\mu_1)^2}{\sigma_1^2} - \frac{(s-\mu_0)^2}{\sigma_0^2}\Big], \end{aligned} \qquad (21)$$

where the parameters (μ_0, σ_0^2) are estimated from a set of positive scores, denoted by $\mathcal{P}$, of the calibration training data, i.e.,

$$\hat{\mu}_0 = \frac{1}{|\mathcal{P}|}\sum_{s\in\mathcal{P}} s, \quad \hat{\sigma}_0^2 = \frac{1}{|\mathcal{P}|}\sum_{s\in\mathcal{P}}(s - \hat{\mu}_0)^2, \qquad (22)$$

and (μ_1, σ_1^2) are estimated similarly but from a set of negative scores, denoted by $\mathcal{N}$, of the calibration training data.

If there are some skewness in the probability density, as illustrated in Fig. 10, a skew normal distribution is more appropriate to model the scores. Motivated by the fact that misclassification errors only happen when the scores fall into the shaded area as illustrated in Fig. 10, we still use LLR given in (21) but estimate μ_0 and σ_0^2 according to

$$\hat{\mu}_0 = \hat{\mu}_0', \quad \hat{\sigma}_0^2 = \frac{1}{|\mathcal{P}'|}\sum_{s\in\mathcal{P}'}(s - \hat{\mu}_0')^2, \qquad (23)$$

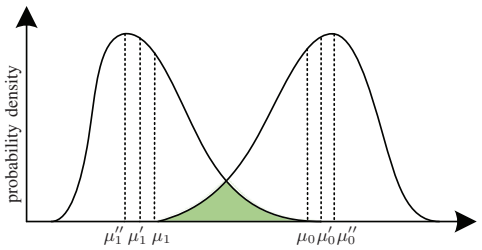


Fig. 10: Diagram of the probability density function of the skew normal distributions, where μ_0 and μ_1 denote the means of the distributions, μ_0' and μ_1' denote the medians, and μ_0'' and μ_1'' denote the modes, respectively.

where $\hat{\mu}_0'$ denotes the median of $\mathcal{P}$, and $\mathcal{P}'$ consists of the scores in $\mathcal{P}$ that are less than the median $\hat{\mu}_0'$. The parameters (μ_1, σ_1^2) are estimated analogously. Specifically, the μ_1 is estimated using the median $\hat{\mu}_1'$ of $\mathcal{N}$, and the value of σ_1^2 is estimate from negative scores in $\mathcal{N}$ that are larger than the median $\hat{\mu}_1'$.

REFERENCES

[1] T. Kinnunen and H. Li, "An overview of text-independent speaker recognition: From features to supervectors," *Speech Communication*, vol. 52, no. 1, pp. 12–40, Jan. 2010.

[2] M.-W. Mak and J.-T. Chien, *Machine learning for speaker recognition*. Cambridge: Cambridge University Press, 2020.

[3] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2014, pp. 4052–4056.

[4] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2018, pp. 5329–5333.

[5] W. Cai, J. Chen, J. Zhang, and M. Li, "On-the-fly data loader and utterance-level aggregation for speaker and language recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing.*, vol. 28, pp. 1038–1051, 2020.

[6] J. Zhou, T. Jiang, Z. Li, L. Li, and Q. Hong, "Deep speaker embedding extraction with channel-wise feature responses and additive supervision softmax loss function," in *Proc. Interspeech*, 2019, pp. 2883–2887.

[7] S. Wang, J. Rohdin, O. Plchot, L. Burget, K. Yu, and J. Černockỳ, "Investigation of specaugment for deep speaker embedding learning," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2020, pp. 7139–7143.

[8] B. Gu, W. Guo, L. Dai, and J. Du, "An improved deep neural network for modeling speaker characteristics at different temporal scales," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2020, pp. 6814–6818.

[9] Z. Gao, Y. Song, I. McLoughlin, W. Guo, and L. Dai, "An improved deep embedding learning method for short duration speaker verification," in *Proc. Interspeech*, 2018, pp. 3578–3582.

[10] S. Dey, T. Koshinaka, P. Motlicek, and S. Madikeri, "DNN based speaker embedding using content information for text-dependent speaker verification," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2018, pp. 5344–5348.

[11] H. Zhang, L. Wang, K. A. Lee, M. Liu, J. Dang, and H. Chen, "Meta-learning for cross-channel speaker verification," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2021, pp. 5839–5843.

[12] 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.

[13] Z. Bai, X.-L. Zhang, and J. Chen, "Cosine metric learning based speaker verification," *Speech Communication*, vol. 118, pp. 10–20, Apr. 2020.

[14] ——, "Speaker verification by partial AUC optimization with mahalanobis distance metric learning," *IEEE/ACM Transactions on Audio, Speech, and Language Processing.*, vol. 28, pp. 1533–1548, 2020.

[15] L. Li, Y. Zhang, J. Kang, T. F. Zheng, and D. Wang, "Squeezing value of cross-domain labels: a decoupled scoring approach for speaker verification," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2021, pp. 5829–5833.

[16] Y. Cai, L. Li, A. Abel, X. Zhu, and D. Wang, "Deep normalization for speaker vectors," *IEEE/ACM Transactions on Audio, Speech, and Language Processing.*, vol. 29, pp. 733–744, 2021.

[17] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer, "End-to-end text-dependent speaker verification," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2016, pp. 5115–5119.

[18] D. Snyder, P. Ghahremani, D. Povey, D. Garcia-Romero, Y. Carmiel, and S. Khudanpur, "Deep neural network-based speaker embeddings for end-to-end speaker verification," in *Proc. IEEE Spoken Language Technology Workshop (SLT)*, 2016, pp. 165–170.

[19] N. Brümmer and E. De Villiers, "The bosaris toolkit: Theory, algorithms and code for surviving the new dcf," *arXiv e-prints*, p. arXiv:1304.2865, Apr. 2013.

[20] L. Ferrer, M. K. Nandwana, M. McLaren, D. Castan, and A. Lawson, "Toward fail-safe speaker recognition: Trial-based calibration with a reject option," *IEEE/ACM Transactions on Audio, Speech, and Language Processing.*, vol. 27, no. 1, pp. 140–153, Jan. 2019.

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

[22] Z. Huang, S. Wang, and K. Yu, "Angular softmax for short-duration text-independent speaker verification." in *Proc. Interspeech*, 2018, pp. 3623–3627.

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

[24] W. Xie, A. Nagrani, J. S. Chung, and A. Zisserman, "Utterance-level aggregation for speaker recognition in the wild," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2019, pp. 5791–5795.

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

[26] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2013, pp. 6655–6659.

[27] X. Chen, X. Liu, M. J. Gales, and P. C. Woodland, "Recurrent neural network language model training with noise contrastive estimation for speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2015, pp. 5411–5415.

[28] L. Wan, Q. Wang, A. Papir, and I. L. Moreno, "Generalized end-to-end loss for speaker verification," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2018, pp. 4879–4883.

[29] J. S. Chung, J. Huh, S. Mun, M. Lee, H.-S. Heo, S. Choe, C. Ham, S. Jung, B.-J. Lee, and I. Han, "In defence of metric learning for speaker recognition," in *Proc. Interspeech*, 2020, pp. 2977–2981.

[30] S. M. Kye, Y. Jung, H. B. Lee, S. J. Hwang, and H. Kim, "Meta-learning for short utterance speaker recognition with imbalance length pairs," in *Proc. Interspeech*, 2020, pp. 2982–2986.

[31] Z. Bai, X.-L. Zhang, and J. Chen, "Partial AUC optimization based deep speaker embeddings with class-center learning for text-independent speaker verification," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2020, pp. 6819–6823.

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

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

[34] J. S. Chung, A. Nagrani, and A. Zisserman, "VoxCeleb2: Deep speaker recognition," in *Proc. Interspeech*, 2018, pp. 1086–1090.

[35] F. R. Rahman Chowdhury, Q. Wang, I. L. Moreno, and L. Wan, "Attention-based models for text-dependent speaker verification," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2018, pp. 5359–5363.

[36] 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. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2019, pp. 5796–5800.

[37] N. Ryant, K. Church, C. Cieri, A. Cristia, J. Du, S. Ganapathy, and M. Liberman, "First DIHARD challenge evaluation plan," *Tech. Rep.*, 2018.

[38] Y. Fujita, N. Kanda, S. Horiguchi, Y. Xue, K. Nagamatsu, and S. Watanabe, "End-to-end neural speaker diarization with self-attention," in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2019, pp. 296–303.

[39] L. Ferrer and M. McLaren, "A discriminative condition-aware backend for speaker verification," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2020, pp. 6604–6608.

[40] H. Xuan, A. Stylianou, X. Liu, and R. Pless, "Hard negative examples are hard, but useful," in *Proc. European Conference on Computer Vision*, 2020, pp. 126–142.

[41] A. K. Menon and R. C. Williamson, "Bipartite ranking: a risk-theoretic perspective," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 6766–6867, 2016.

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

[43] C. Cortes and M. Mohri, "AUC optimization vs. error rate minimization," in *Advances in neural information processing systems*, 2003, pp. 313–320.

[44] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (roc) curve," *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.

[45] N. Brümmer, A. Swart, and D. van Leeuwen, "A comparison of linear and non-linear calibrations for speaker recognition," in *Proc. Odyssey The Speaker and Language Recognition Workshop*, 2014, pp. 14–18.

[46] H. S. Heo, B.-J. Lee, J. Huh, and J. S. Chung, "Clova baseline system for the voxceleb speaker recognition challenge 2020," *arXiv e-prints*, p. arXiv:2009.14153, Sep. 2020.

[47] D. Snyder, G. Chen, and D. Povey, "Musan: A music, speech, and noise corpus," *arXiv e-prints*, p. arXiv:1510.08484, Oct. 2015.

[48] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2017, pp. 5220–5224.

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

[50] 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.

[51] R. K. Das, R. Tao, and H. Li, "HLT-NUS submission for 2020 NIST conversational telephone speech SRE," *arXiv e-prints*, p. arXiv:2111.06671, Nov. 2021.

[52] A. Nagrani, J. S. Chung, J. Huh, A. Brown, E. Coto, W. Xie, M. McLaren, D. A. Reynolds, and A. Zisserman, "Voxsrc 2020: The second voxceleb speaker recognition challenge," *arXiv e-prints*, p. arXiv:2012.06867, Dec. 2020.

[53] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proc. IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[54] J. P. Campbell, W. Shen, W. M. Campbell, R. Schwartz, J.-F. Bonastre, and D. Matrouf, "Forensic speaker recognition," *IEEE Signal Processing Magazine*, vol. 26, no. 2, pp. 95–103, 2009.