CS-REP: MAKING SPEAKER VERIFICATION NETWORKS EMBRACING RE-PARAMETERIZATION

Ruiteng Zhang[1], *Jianguo Wei*[1,2], *Wenhuan Lu*[1], *Lin Zhang*[3], *Yantao Ji*[4], *Junhai Xu*[1], *Xugang Lu*[5]

[1]College of Intelligence and Computing, Tianjin University, Tianjin, China
[2]Computer College, Qinghai Nationalities University, Xining, China
[3]National Institute of Informatics, Tokyo, Japan
[4]School of Software Engineering, Xi'an Jiaotong University, Xi'an, China
[5]National Institute of Information and Communications Technology, Kyoto, Japan

ABSTRACT

Automatic speaker verification (ASV) systems, which determine whether two speeches are from the same speaker, mainly focus on verification accuracy while ignoring inference speed. However, in real applications, both inference speed and verification accuracy are essential. This study proposes cross-sequential re-parameterization (CS-Rep), a novel topology re-parameterization strategy for multi-type networks, to increase the inference speed and verification accuracy of models. CS-Rep solves the problem that existing re-parameterization methods are not suitable for typical ASV backbones. When a model applies CS-Rep, the training-period network utilizes a multi-branch topology to capture speaker information, whereas the inference-period model converts to a time-delay neural network (TDNN)-like plain backbone with stacked TDNN layers to achieve the fast inference speed. Based on CS-Rep, an improved TDNN with friendly test and deployment called Rep-TDNN is proposed. Compared with the state-of-the-art model ECAPA-TDNN, Rep-TDNN increases the actual inference speed by about 50% and reduces the EER by 10%. The code and trained models are available at https://github.com/zrtlemontree/CS-Rep.

Index Terms— Speaker verification, cross-sequential transformation, re-parameterization, inference speed

1. INTRODUCTION

Automatic speaker verification (ASV) systems determine whether two speeches are from the same speaker or not. The front-end models of ASV systems to extract speaker embeddings from speech utterances to represent the corresponding speaker, such as i-vector [1], d-vector [2], and x-vector [3]. The back-end functions of ASV systems calculate similarity scores between two embeddings, e.g., probabilistic linear discriminant analysis(PLDA)[4] and cosine similarity.

Currently, ASV systems are widely used in many real-world applications, such as criminal investigation, payment [5], and real-time meeting recordings [6]. In those realistic scenarios, the inference speed is becoming increasingly important.

Recently, the use of multi-branch topology (e.g., ResNet [7] has a convolutional branch and a shortcut branch) [8, 9] to improve the theoretical efficiency and accuracy of models has been reported. These designs capture multi-scale information through multi-type convolution kernels with a small number of channels. Additionally, many studies [10, 11] have used multi-branch designs to strengthen the time-delay neural network [3](TDNN). ECAPA-TDNN[10] with multi-branch topology has exhibited excellent performance in recent ASV competitions [12, 13, 14, 15, 16]. Unfortunately, the actual inference speed of the multi-branch designs is typically much slower than its theoretical speed. The main reason is that the multi-branch designs increase the memory access cost (MAC) and are unfriendly for parallel computing [17]. Therefore, this topology does not solve the contradiction of speed and performance in the real applications.

To improve the actual inference speed for the multi-branch models, Ding et al. [18] proposed a re-parameterization approach to convert the multi-branch network to a plain topology in the inference-period. However, their strategy is based on the structure of "conv-bn-activation," but the mainstream TDNN-based ASV models rely on the "conv-activation-bn" structure to realize good verification accuracy (we confirm that this is reasonable in Section 4.2). Many successful implementations of ASV have adopted the structure, such as the x-vector [3] (implemented by Kaldi [19]), E-TDNN [20], and ECAPA-TDNN [10], etc. In those ASV models, since their convolutional layer is not adjacent to the batch normalization (BN) [21] layer, they cannot be re-weighted through Ding's method.

In this study, we propose the cross-sequential re-parameterization (CS-Rep, Fig. 1) to increase the inference speed and verification accuracy of ASV systems, especially for the models based on the "conv-activation-bn"[1] architecture. Specifically, we build a multi-branch network with the "conv-activation-bn" structure during the training period. In the inference period, CS-Rep adjusts the order of modules of the sequential layer to the "bn-conv-activation" losslessly, causing the BN to be adjacent to the convolutional layer. However, the "bn-first"[2] case will generate influences on each channel of the convolutional layer. We propose the "bn-first" re-parameterization method to model these influences by the discrete convolution operator. Then, CS-Rep adopts this approach to convert the multi-branch network to a TDNN-like plain topology while maintaining the multi-scale information capturing ability. Therefore, the model adopted CS-Rep achieves an efficient architecture with friendly parallel computing.

Junhai Xu is the corresponding author. Thanks to NSFC of China (No.61876131, No. U1936102), Key R&D Program of Tianjin (No.19ZXZNGX00030). Regional Innovation Cooperation Project of Sichuan (Grant No.22QYCX0082)

[1]The sequential layer [22] is a layer including multiple modules by order. For example, the "conv-activation-bn" is a sequential layer with the order of 1-D (TDNN) or 2-D convolutional layer, activation function, and BN layer. Fig. 1 (a) shows the diagram of the sequential layer with three branches.

[2]The "bn-first" means the case that the BN layer precedes the convolutional layer in the sequential layer. e.g., the "bn-conv-activation."

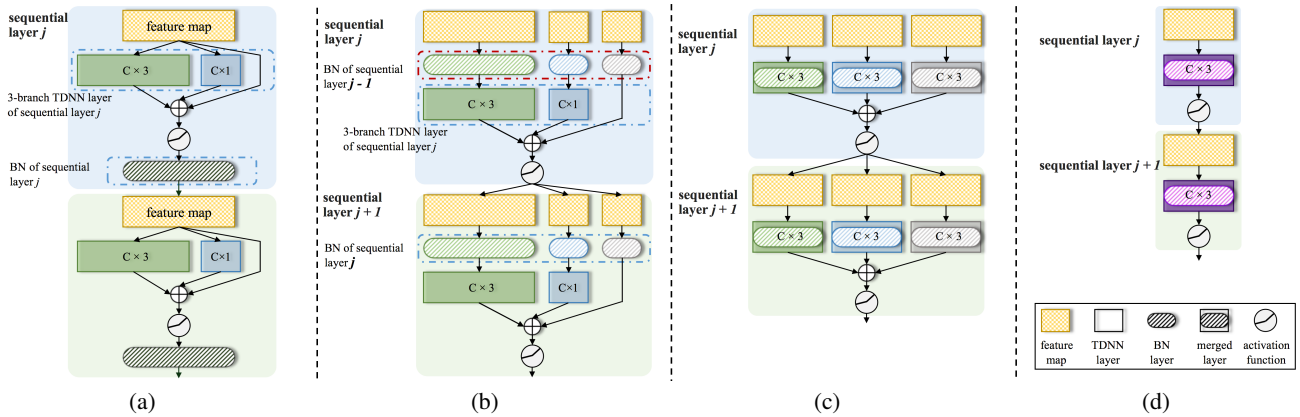

Fig. 1: The topology changes for our model when adopting the proposed CS-Rep: (a) original topology of model, (b) step, (c) steps 2 and 3, and (d) step 4.

The contributions of this study are listed as follows. (1) We propose CS-Rep to increase the inference speed, decrease the parameters, and reduce the floating-point operations (FLOPs) of networks with the "conv-activation-bn" structure while maintaining the accuracy. (2) We also combine the "bn-first" and "conv-first" [18] re-parameterization as an out-of-the-box method, which can be conveniently used for all types of sequential layers ("conv-first" and "bn-first") to improve the inference speed. (3) Based on CS-Rep, we construct Rep-TDNN. It achieves the state-of-the-art performance (1.09% EER on VoxCeleb1-test, without any data augmentation), and its inference speed is 50% faster than the state-of-the-art model ECAPA-TDNN[3].

The remainder of the paper is organized as follows. Related works are briefly introduced in Section 2, the CS-Rep method and the novel Rep-TDNN model are detailed in Section 3, experimental results are presented and analyzed in Section 4, and the conclusions are given in Section 5.

2. RELATED WORK

Time-delay neural networks: The TDNN-based x-vector [3] system extracts robust embeddings by capturing the long-term temporal relations of utterances through the one-dimensional convolutional layer [23]. Several improved versions of TDNN have been proposed, such as E-TDNN [24] with stable performance and satisfactory inference speed, and ECAPA-TDNN [10] with compelling accuracy.

Squeeze-excitation block: Squeeze-excitation (SE) block [25] has been proved successful in modeling global channel interdependencies in speech, bringing improved accuracy to ASV [10].

Multi-branch topology: Multi-branch topology [8, 9] involves multiple branches of different types in one sequential layer, so it helps to capture important information from multiple scales. In ASV, many solid models have adopted it, such as InceptionNet [26], Res2Net [9], ECAPA-TDNN [10], and HS-Net [11]. However, multi-branch topology increases the complexity of network topology, leading to negative impacts on the inference speed.

RepVGG: RepVGG [18] proposed a structural re-parameterization technique to improve the performance and reduce the computational complexity during inference. Their model applied the plain topology in the inference period and the multi-branch structure in the training period. But there are two limitations in their method: (1) the re-parameterization method proposed in RepVGG is not suitable for ASV because the approach bases on the "conv-bn-activation" structure, but mainstream ASV models adopt the "conv-activation-bn" structure; (2) their work lacks the condition of the re-parameterization in TDNNs.

[3]The code of ECAPA-TDNN is presented in https://github.com/speechbrain/speechbrain.

3. PROPOSED METHODS

This section will describe the proposed CS-Rep strategy through a three-branch TDNN. CS-Rep will be used to develop Rep-TDNN with an extremely efficient architecture and high performance.

3.1. Multi-branch training structure

The three-branch structure [18] is introduced to build a TDNN block with excellent speaker information extraction. In our TDNN block, one sequential layer consists of three branches (Fig. 1 (a)): a TDNN branch $TDNN^{(3)}$ with context = 3, a TDNN branch $TDNN^{(1)}$ with context = 1, and a shortcut connection, respectively. The operation within one sequential layer can be defined as:

$$\mathbf{M}_j = BN_j(ReLU(TDNN_j^{(3)}(\mathbf{M}_{j-1}) + TDNN_j^{(1)}(\mathbf{M}_{j-1}) + \mathbf{M}_{j-1})), \quad (1)$$

where $\mathbf{M}_j \in \mathbb{R}^{B\times N\times T}$ is the output matrix of the j-th sequential layer, B is the mini-batch size, N is the number of the feature dimension, T is the number of frames, $ReLU$ is the ReLU function, and BN_j is the batch normalization (BN) layer of the j-th sequential layer.

3.2. Cross sequential re-parameterization method

This subsection describes how to convert a multi-branch network based on the "conv-activation-bn" structure to a plain network using the cross sequential re-parameterization function (CS-Rep). Fig. 1 and Fig. 2 display the topology diagram and parameter diagram for the network during the processing of CS-Rep. The four steps of the approach are as follows:

Step 1: Cross-sequential converting. We distribute BN_{j-1} from sequential-layer$_{j-1}$ to the head of the branches of sequential-layer$_j$, causing the model with the "conv-activation-bn" structure to the "bn-conv-activation" structure. This transform leads to the BN and TDNN layers contiguous and allowing them to be merged. This strategy is called cross-sequential transformation (CST). The diagram of CST is displayed in Fig. 1 (b). After utilizing CST, the sequential layer of Eq. (1) can be transformed to

$$\begin{aligned}\mathbf{M}_j = & ReLU(TDNN_j^{(3)}(BN_{j-1}(\mathbf{M}_{j-1})) + \\ & TDNN_j^{(1)}(BN_{j-1}(\mathbf{M}_{j-1})) + BN_{j-1}(\mathbf{M}_{j-1})).\end{aligned} \quad (2)$$

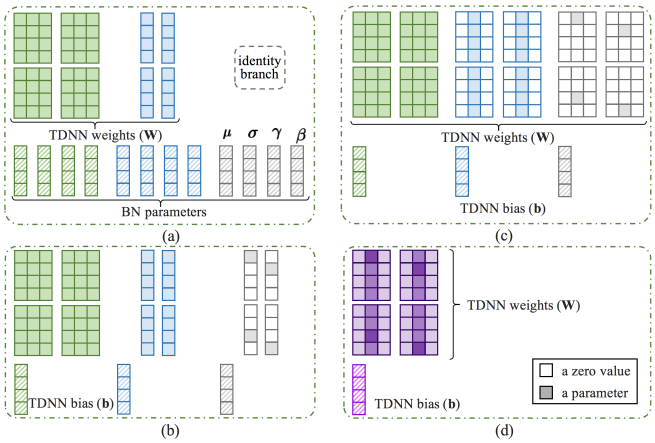

Fig. 2: Parameters converting of the "bn-first" re-parameterization in TDNN: (a) original parameters of weight and BN, (b) steps 1 and 2, (c) step 3, and (d) step 4.

Then we define that $\mathbf{W}_j^{(C)} \in \mathbb{R}^{N_o \times N_i \times C}$ is the weight of TDNN ($TDNN_j$) from the j-th sequential layer, N_i and N_o are the numbers of input and output channels respectively, C is the number of the context of TDNN, and "$*$" is the discrete convolution operator. Eq. (2) can be re-written as:

$$\begin{aligned}\mathbf{M}_j =& ReLU(\mathbf{W}_j^{(3)} * BN_{j-1}(\mathbf{M}_{j-1}) + \\ & \mathbf{W}_j^{(1)} * BN_{j-1}(\mathbf{M}_{j-1}) + BN_{j-1}(\mathbf{M}_{j-1})).\end{aligned} \tag{3}$$

Step 2: Merge BN_{j-1} into $TDNN_j$. The BN [21]in TDNN is

$$BN(\mathbf{M}, \boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\gamma}, \boldsymbol{\beta})_{:,n_o,:} = (\mathbf{M}_{:,n_o,:} - \mu_{n_o}) \frac{\gamma_{n_o}}{\sigma_{n_o}} + \beta_{n_o}, \tag{4}$$

where $n_o \in \{1, 2, ..., N_o\}$ denotes the n_o-th output channel, and $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$, $\boldsymbol{\gamma}$, and $\boldsymbol{\beta}$ represent the mean of mini-batch, the variance of mini-batch, and the scale and shift parameters, respectively. And μ_{n_o}, σ_{n_o}, γ_{n_o}, and β_{n_o} are the elements of vectors $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$, $\boldsymbol{\gamma}$, and $\boldsymbol{\beta}$, respectively. In the "bn-first" (BN is before the TDNN operator) re-parameterization, the condition that BN is before the convolutional layer will generate influences on each convolutional layer channel. The influence can be calculated as the bias of the convolutional layer through the discrete convolution operator. The "bn-first" re-parameterization converts every TDNN branch and its preceding BN into a TDNN ($\mathbf{W}$) with a bias vector ($\mathbf{b}$) to reduce the depth of network (Fig. 1 (c)), and it can be defined as:

$$\begin{aligned}\mathbf{W}'_{j,(n_o,:,:)} &= \mathbf{W}_{j,(n_o,:,:)} \cdot \frac{\gamma_{j-1,(n_o)}}{\sigma_{j-1,(n_o)}}, \\ b'_{j,(n_o)} &= \mathbf{W}_{j,(n_o,:,:)} * (-\frac{\boldsymbol{\gamma}_{j-1}\boldsymbol{\mu}_{j-1}}{\boldsymbol{\sigma}_{j-1}} + \boldsymbol{\beta}_{j-1}),\end{aligned} \tag{5}$$

where j denotes the j-th sequential layer, $\mathbf{W}'_j$ and $\mathbf{b}'_j$ are the weight and bias of the new TDNN branch through the "bn-first" re-weight, and $\mathbf{W}'_{j,(n_o,:,:)}$ is tensor slice of $\mathbf{W}'_j$ and $b'_{j,(n_o)}$ is the element of $\mathbf{b}'_j$. Fig. 2 (b) displays the weight diagram.

Then, we generalize a hot-swap algorithm for the "bn-first" (Eq. (5)) and "conv-first" [18] re-parameterization methods. The algorithm can be written as:

$$\begin{aligned}\mathbf{W}'_{j,(n_o,:,:)} &= \mathbf{W}_{j,(n_o,:,:)} \cdot \Phi_{n_o}; \text{ where } \boldsymbol{\Phi} = \begin{cases} \frac{\boldsymbol{\gamma}_{j-1}}{\boldsymbol{\sigma}_{j-1}}, & \text{bn first} \\ \frac{\boldsymbol{\gamma}_j}{\boldsymbol{\sigma}_j}, & \text{conv first,} \end{cases} \\ b'_{j,(n_o)} &= \begin{cases} \mathbf{W}_{j,(n_o,:,:)} * (-\frac{\boldsymbol{\gamma}_{j-1}\boldsymbol{\mu}_{j-1}}{\boldsymbol{\sigma}_{j-1}} + \boldsymbol{\beta}_{j-1}), & \text{bn first} \\ -\frac{\gamma_{j,(n_o)}\mu_{j,(n_o)}}{\sigma_{j,(n_o)}} + \beta_{j,(n_o)}, & \text{conv first.} \end{cases}\end{aligned} \tag{6}$$

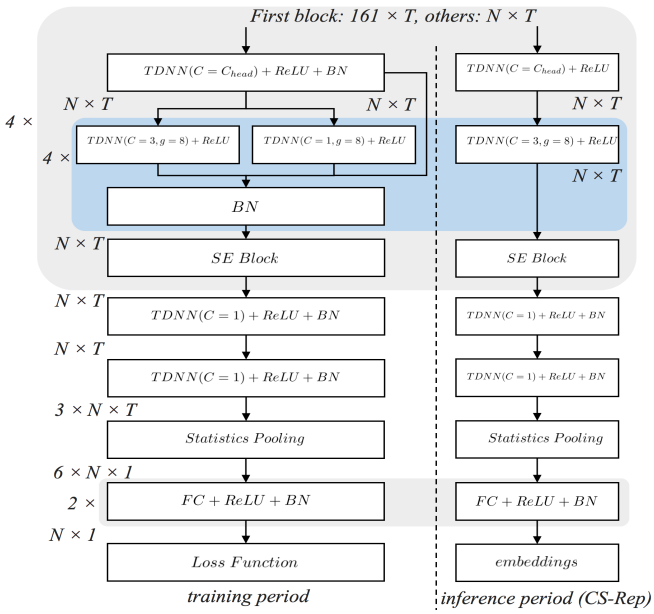

Fig. 3: The topologies of Rep-TDNN without and with CS-Rep. The context size of the head TDNN in each block is $C_{head} \in [5, 1, 1, 5]$, the g represents the groups of layers, and the channel N of each TDNN layer and branch is 512. *FC*, *BN*, *SE* denote fully connected layer, batch normalization, and squeeze-excitation, respectively. (**Left**): The multi-branch design in the training period. (**Right**): The plain topology in the inference period.

For the re-weight of the identity branch, it can be converted to a TDNN branch with context = 1 ($\mathbf{W}^{id}$):

$$\mathbf{w}^{id}_{n_o,n_i,:} = \begin{cases} 1, & n_o = n_i \\ 0, & n_o \neq n_i, \end{cases} \tag{7}$$

where $\mathbf{w}^{id}_{n_o,n_i,:}$ is the tensor slice of $\mathbf{W}^{id}$, N_i and N_o are the numbers of input and output channels of TDNN, and $n_i \in \{1, 2, ..., N_i\}$ and $n_o \in \{1, 2, ..., N_o\}$. The diagram of building a TDNN layer equivalent to the shortcut connection is shown in the gray branch of Fig. 2(b). After converting the shortcut branch to the TDNN branch, it can be re-weighted by the "bn-first" re-parameterization approach.

Step 3: Pad the weight of branches to the same dimension. As for the TDNN branches with context = 1, we can use zero-padding on its weight to pad to context = 3 (Fig. 2 (c)). After step 3 (Fig. 1 (c)), the sequential-layer$_j$ is

$$\begin{aligned}\mathbf{M}_j =& ReLU((\mathbf{W}'^{(3)}_j * \mathbf{M}_{j-1} + \mathbf{b}'^{(3)}_j) + \\ & (\mathbf{W}'^{(1\to3)}_j * \mathbf{M}_{j-1} + \mathbf{b}'^{(1\to3)}_j) + (\mathbf{W}'^{(0\to3)}_j * \mathbf{M}_{j-1} + \mathbf{b}'^{(0\to3)}_j)),\end{aligned} \tag{8}$$

where $\mathbf{W}'^{(1\to3)}_j$ means that the TDNN weight is changed from context = 1 to context = 3 through zero-padding. $\mathbf{b}'_j$ is the bias of the corresponding branch, and the superscript of $\mathbf{b}'_j$ is only used to distinguish the branch uniformly while it does not mean padding.

Step 4: Merge all branches to one TDNN layer. Based on the linear additivity of the discrete convolution operator [18], the weight and bias of every branch can be added to one branch to implement the merge-branch-convert easily. The final model only constructed by TDNN and ReLU(Fig.1(d) and Fig.2(d)),which can be written as:

$$\begin{aligned}&... \\ &\mathbf{M}_j = ReLU(TDNN'^{(3)}_j(\mathbf{M}_{j-1})) \\ &\mathbf{M}_{j+1} = ReLU(TDNN'^{(3)}_{j+1}(\mathbf{M}_j)) \\ &...\end{aligned} \tag{9}$$

Table 1: The benchmark on VoxCeleb1 test, VoxCeleb1-E, and VoxCeleb1-H. Models were inferred on the single GPU to calculate the inference speed. Speed[4] in this table was defined as frame-numbers/processing-time.

Description	Backbone	Speed↑	VoxCeleb1-test		VoxCeleb1-E		VoxCeleb1-H	
			EER↓ (%)	minDCF↓	EER↓ (%)	minDCF↓	EER↓ (%)	minDCF↓
Our implementation	E-TDNN ("conv-activation-bn")	175,512	1.617	0.2324	1.584	0.3036	2.771	0.4051
Our implementation	E-TDNN ("conv-bn-activation")	175,826	2.096	0.3579	1.889	0.3629	3.358	0.4575
Our implementation	ResNet-34	72,656	2.021	0.2696	2.200	0.4136	3.822	0.4753
Our implementation	ECAPA-TDNN	62,802	1.181	0.2685	1.343	0.2717	2.578	0.4188
Proposed	Rep-TDNN (original topology)	58,877	1.117	0.1914	1.193	0.2364	2.163	0.3522
Proposed	Rep-TDNN (CS-Rep)	92,903	1.090	0.1964	1.199	0.2377	2.272	0.3569

3.3. Proposed model

Based on CS-Rep in Section 3.2, we propose an improved TDNN called Rep-TDNN, which has excellent performance, fast inference speed, few parameters, and fewer floating-point operations (FLOPs). The overview of Rep-TDNN is displayed in Fig. 3. The frame-level model consists of four blocks, including TDNN layers, LeakyReLU activation functions, BN, and SE-Block.

In the training period (Fig. 3 (right)), each block has one head TDNN layer (the first TDNN layer in the gray box) that integrates feature information and four TDNN sequential layers with three branches (a branch with context = 3, a branch with context = 1, and a shortcut branch; describes in Section 3.1) with the "conv-activation-bn" structure. The context C_{head} of each head TDNN layer is [5, 1, 1, 5]. SE-block is adopted in each block to model the channel attention. Statistics pooling aggregates frame-level features to segment-level features passed to two fully connected (FC) layers. AAM-Softmax [27] loss is to choose for loss function.

In the inference period, the CS-Rep method is adopted to convert the model from the multi-branch topology to the plain topology (Fig. 3 (left)). The parameters and FLOPs of Rep-TDNN (CS-Rep) are only 6.9e+6 and 1.4e+9, which are the same as for E-TDNN [10, 23].

4. EXPERIMENTS

4.1. Experimental details

Dataset: To investigate the performance of the models, we created a benchmark on the VoxCeleb [28, 29] dataset. We implemented some successful architectures, including E-TDNN [23], ResNet34 [10], and ECAPA-TDNN [10], as our baselines. Models were trained on the VoxCeleb2 development sets without data augmentation, and tested on the VoxCeleb1 test set, VoxCeleb1-E, and VoxCeleb1-H [29]. 161-dimensional spectrograms were generated through hamming sliding window with a width of 20 ms and a step of 10 ms. The cepstral mean and variance normalization (CMVN) was applied. No voice activity detection was adopted.

Implementation details: Models were trained by AAM-Softmax loss with $m = 0.25$ and $s = 30$. In the training stage, the mini-batch size of 64 was used. Each training speech sample was randomly sampled for 300 frames from recordings. Stochastic gradient descent (SGD) with momentum 0.9, weight decay 1e-5, and initial learning rate 0.1 was utilized. Full-length utterances were adopted in the testing stage to extract embeddings. The GPU type and CUDA version were NVIDIA 2080Ti and 10.2. Adaptive score normalization (AS-Norm) [30] with cosine similarity was applied for all models. L2-normalized speaker embeddings of each training speaker were chosen as the imposter cohort with a size of 1000.

Evaluation metrics: We utilized the equal error rate (EER) and minimum detection cost function (minDCF) from NIST SRE10 [31] as the performance metrics. Moreover, we adopted the actual inference speed (frame-numbers/processing-time) to compare the speed of different models. It is more objective than the indirect speed metrics parameters and FLOPs, which can avoid the bad case in which a model with few parameters and complex topology has a low FLOPs but the actual inference speed may be slow [17].

[4]Other common metrics for speed, e.g., RTF, RTX, FPS, etc., which can be easily converted from these results.

4.2. Experimental results

The results of the state-of-the-art baseline on VoxCeleb1-test, VoxCeleb1-E, and VoxCeleb1-H are shown in Table 1. The results of the inference speed are the average of 5 runs.

In Table 1, the E-TDNN with the "conv-activation-bn" structure obtained approximately 1.6%, 1.6%, and 2.8% EER on the above three evaluation sets, respectively, whereas the E-TDNN with the "conv-bn-activation" structure only achieved 2.1%, 1.9%, and 3.4% EER, respectively. These results confirm that the "conv-activation-bn" architecture is necessary for high-accuracy ASV networks.

Compared with the best E-TDNN, the EER results of Rep-TDNN were much lower, relatively reduced by 32.6%, 24.3%, and 18.0% on three evaluation conditions, respectively. Satisfactory performances showed the strong discrimination of speakers for Rep-TDNN. As shown in Table 1, Rep-TDNN obtained approximately 1.2% EER and 2.2% EER on VoxCeleb1-E and VoxCeleb1-H, respectively, which are both hard evaluation test sets. Compared with the results with ECAPA-TDNN, Rep-TDNN had 10.7% and 11.9% reductions of EER on two hard test sets and a 47.9% increase of inference speed. We can find that compared to ECAPA-TDNN, Rep-TDNN was more efficient and more robust in hard conditions.

Furthermore, CS-Rep can increase the inference speed for the model with negligible influence on the accuracy. Compared with the Rep-TDNN without CS-Rep, the inference speed of the model with CS-Rep was greatly improved (by 58%) while EERs stayed the same. In Table 1, Rep-TDNN through CS-Rep can process 92,903 frames per second, whereas Rep-TDNN with original topology only can process 58,877 frames per second. The results show that CS-Rep is an essential component for developing ASV models with the fast inference speed and high accuracy.

Generally, Rep-TDNN showed relatively improvements of approximately 50% in inference speed and 10% in performance compared with state-of-the-art ASV models. Hence, it has great potential for the industry, saving spends for test and deployment.

5. CONCLUSION

This study presents an upgraded re-parameterization approach called CS-Rep to re-parameterize models with a multi-branch design more efficiently. It can be used to reduce the size of parameters, reduce inference time, and improve the performance of ASV systems. We also integrate the "bn-first" and "conv-first" of re-parameterization equations into a plug-and-play module. Based on CS-Rep, this work proposes Rep-TDNN, which obtains 18% – 32.6% relative reduction in EER compared with solid baselines. The positive results show that Rep-TDNN reduces the inference time by approximately 50% and improves the performance by approximately 10% compared with the state-of-the-art model ECAPA-TDNN.

6. REFERENCES

[1] Najim Dehak, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[2] Ehsan Variani, Xin Lei, Erik McDermott, Ignacio Lopez Moreno, and Javier Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *Proc. ICASSP*, 2014, pp. 4052–4056.

[3] David Snyder, Daniel Garcia-Romero, Daniel Povey, and Sanjeev Khudanpur, "Deep neural network embeddings for text-independent speaker verification," in *Proc. Interspeech*, 2017, pp. 999–1003.

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

[5] Jiwei Xu, Xinggang Wang, Bin Feng, and Wenyu Liu, "Deep multi-metric learning for text-independent speaker verification," *Neurocomputing*, vol. 410, pp. 394–400, 2020.

[6] Shota Horiguchi, Yusuke Fujita, Shinji Watanabe, Yawen Xue, and Kenji Nagamatsu, "End-to-end speaker diarization for an unknown number of speakers with encoder-decoder based attractors," in *Proc. Interspeech*, 2020, pp. 269–273.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016, pp. 770–778.

[8] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He, "Aggregated residual transformations for deep neural networks," in *Proc. CVPR*, 2017, pp. 1492–1500.

[9] Shanghua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip HS Torr, "Res2net: A new multi-scale backbone architecture," *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[10] Brecht Desplanques, Jenthe Thienpondt, and Kris Demuynck, "Ecapa-tdnn: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification," in *Proc. Interspeech*, 2020, pp. 3830–3834.

[11] Yu-Jia Zhang, Yih-Wen Wang, Chia-Ping Chen, Chung-Li Lu, and Bo-Cheng Chan, "Improving Time Delay Neural Network Based Speaker Recognition with Convolutional Block and Feature Aggregation Methods," in *Proc. Interspeech*, 2021, pp. 76–80.

[12] Arsha Nagrani, Joon Son Chung, Jaesung Huh, Andrew Brown, Ernesto Coto, Weidi Xie, Mitchell McLaren, Douglas A Reynolds, and Andrew Zisserman, "Voxsrc 2020: The second voxceleb speaker recognition challenge," *arXiv preprint arXiv:2012.06867*, 2020.

[13] Jenthe Thienpondt, Brecht Desplanques, and Kris Demuynck, "The idlab voxceleb speaker recognition challenge 2020 system description," *arXiv preprint arXiv:2010.12468*, 2020.

[14] Weiqing Wang, Danwei Cai, Xiaoyi Qin, and Ming Li, "The dku-dukeece systems for voxceleb speaker recognition challenge 2020," *arXiv preprint arXiv:2010.12731*, 2020.

[15] Hossein Zeinali, Kong Aik Lee, Jahangir Alam, and Lukas Burget, "Short-duration speaker verification (sdsv) challenge 2021: the challenge evaluation plan," *arXiv preprint arXiv:1912.06311*, 2019.

[16] Jenthe Thienpondt, Brecht Desplanques, and Kris Demuynck, "Integrating frequency translational invariance in tdnns and frequency positional information in 2d resnets to enhance speaker verification," *arXiv preprint arXiv:2104.02370*, 2021.

[17] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *Proc. ECCV*, 2018, pp. 116–131.

[18] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun, "Repvgg: Making vgg-style convnets great again," in *Proc. CVPR*, 2021, pp. 13733–13742.

[19] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al., "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society, 2011.

[20] Daniel Garcia-Romero, Alan McCree, David Snyder, and Gregory Sell, "Jhu-hltcoe system for the voxsrc speaker recognition challenge," in *Proc. ICASSP*, 2020, pp. 7559–7563.

[21] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. ICML*, 2015, pp. 448–456.

[22] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al., "Pytorch: An imperative style, high-performance deep learning library," in *Proc. NeurIPS*, 2019, vol. 32, pp. 8026–8037.

[23] Ruiteng Zhang, Jianguo Wei, Wenhuan Lu, Longbiao Wang, Meng Liu, Lin Zhang, Jiayu Jin, and Junhai Xu, "Aret: Aggregated residual extended time-delay neural networks for speaker verification.," in *Proc. Interspeech*, 2020, pp. 946–950.

[24] David Snyder, Daniel Garcia-Romero, Gregory Sell, Alan McCree, Daniel Povey, and Sanjeev Khudanpur, "Speaker recognition for multi-speaker conversations using x-vectors," in *Proc. ICASSP*, 2019, pp. 5796–5800.

[25] Jie Hu, Li Shen, and Gang Sun, "Squeeze-and-excitation networks," in *Proc. CVPR*, 2018, pp. 7132–7141.

[26] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proc. AAAI*, 2017, pp. 4278–4284.

[27] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proc. CVPR*, 2019, pp. 4690–4699.

[28] Joon Son Chung Arsha Nagrani and Andrew Zisserman, "Voxceleb: a large-scale speaker identification dataset," in *Proc. Interspeech*, 2017, pp. 2616–2620.

[29] Arsha Nagrani Joon Son Chung and Andrew Zisserman, "Voxceleb2: Deep speaker recognition," in *Proc. Interspeech*, 2018, pp. 1086–1090.

[30] Sandro Cumani, Pier Domenico Batzu, Daniele Colibro, Claudio Vair, Pietro Laface, and Vasileios Vasilakakis, "Comparison of speaker recognition approaches for real applications," in *Proc. Interspeech*, 2011, pp. 2365–2368.

[31] Alvin F Martin and Craig S Greenberg, "The nist 2010 speaker recognition evaluation," in *Proc. Interspeech*, 2010, pp. 2726–2729.