THUEE SYSTEM DESCRIPTION FOR NIST 2020 SRE CTS CHALLENGE

Yu Zheng[1], *Miao Zhao*[1], *Yufeng Ma*[1], *Min Liu*[1], *Xinyue Ma*[2],
Tianyu Liang[2], Tianlong Kong[2], Liang He[2]*, Minqiang Xu[1]*

[1]SpeakIn Technologies Co. Ltd., ShangHai, China
[2]Department of Electronic Engineering Tsinghua University, Beijing, China

ABSTRACT

This document presents SRE20 system description for the joint effort of the teams at Speakin and department of electronic engineering Tsinghua university (THUEE). The subsystems including ResNet34, ResNet74, ResNet152, RepVGG_B2, EFTDNN_LSTMP, EFTDNN_SE and ETDNN-LSTMP are developed in this evaluation.

Index Terms— NIST2020 SRE CTS challenge, ResNet, RepVGG, EF-TDNN, LSTMP, additive margin , refinement

1. INTRODUCTION

The THUEE submission is the joint effort of the teams at Speakin and the department of electronic engineering Tsinghua university (THUEE). We worked as two sub-teams until the final part of the evaluation. The subsystems, including ResNet34, ResNet74, ResNet152, RepVGG_B2, EFTDNN_LSTMP, EFTDNN_SE, ETDNN_LSTMP were developed in this evaluation. All the subsystems contained a deep neural network followed by scoring and calibration. The ResNet74 and ResNet152 systems use cosine distance to score the trials, while ResNet34, EFTDNN and ETDNN based systems use adapted PLDA backend without adaptive symmetric score normalization. For each system, we describe the data involved in training, the system setup and their hyper-parameters. Finally, we report the experiment results of each subsystem and the fusion system on the SRE18 developing set, SRE19 evaluation set and SRE20 progress set.

2. DATASETS

2.1. Training dataset

The datasets used for training included:

- Our own Chinese telephone data (CHI-tel).
- NIST SRE04-10.
- NIST SRE12 telephone data (SRE12-tel).
- NIST SRE16 telephone data (SRE16-tel).
- NIST SRE18 evaluation datasets.
- NIST SRE19 evaluation datasets.
- VoxCeleb 1+2.
- Switchboard phase1-3.
- MIXER6 telephone phone calls (MX6-tel).
- LibriSpeech.

In total, there are 255214 speakers in this dataset.We collected Chinese telephone data (CHI-tel) for a total of 230,000 speakers. We applied the following techniques to augment each utterance:

- Adding reverberation: artificially reverberation using a convolution with simulated RIRs from the AIR dataset
- Adding music: taking a music file (without vocals) randomly selected from MUSAN[1], trimmed or repeated as necessary to match duration, and added to the original signal (5-15dB SNR).
- Adding noise: MUSAN noises were added at one second intervals throughout the recording (0-15dB SNR).
- Adding Babble: MUSAN speech was added to the original signal (13-20dB SNR).

2.2. Development dataset

SRE19-dev dataset was used for fusion and calibration. SRE18-CMN2 was used for evaluation.

3. SYSTEMS

3.1. ResNet

We used ResNet-74 and ResNet-152 in our systems.

*Corresponding author: heliang@mail.tsinghua.edu.cn xuminqiang@speakin.ai

3.1.1. *Data preparation*

We used all the data described above for training. All 16 kHz recordings were downsampled to 8 kHz using ffmpeg. We used 64 log Mel-filter bank energies with a 25 ms window size and a time shift of 10 ms. Mean normalization was applied using a moving window of 3 seconds. The Kaldi [2] energy VAD was used to detect speech activity.

3.1.2. *Architecture description*

The block to build resnet was bottleneck [3] .The base channel was 64. The block numbers of resnet74 and resnet152 were (3, 4, 14, 3) and (3, 8, 36, 3). A statistics pooling layer was used to aggregate along the time across.

3.1.3. *Loss function*

Additive margin softmax(AMsoftmax) and angular softmax(AAMsoftmax) is proposed in[4] [5]. We combine them, add penalty to both angle and angle cosine, and call it CM-softmax.The loss function is defined as:

$$L = -\frac{1}{n}\sum_{i=1}^{n} \log \frac{\mathrm{e}^{s[\cos(\theta_{y_i}+m_1)-m_2]}}{\mathrm{e}^{s[\cos(\theta_{y_i}+m_1)-m_2]} + \sum_{i\neq y_i} \mathrm{e}^{s\cos\theta_{\mathrm{j}}}} \quad (1)$$

where $\cos\theta_{y_i} = w_{y_i}^T f_i / \|w_{y_i}\| \|f_i\|$, w_{y_i} is the weight vector of class f_i , and f_i is the sample input. Also, s is the scale factor and m_1 and m_2 are penalty margins for angle and angle cosine.

3.1.4. *training*

Since most of our training data is Chinese speech data, there will be domain gaps during the evaluation phase. Therefore, the training was divided into two stages.The first stage used all the training data, and the second stage performed model refinement [6], in-domain data was used to further finetune the first stage model. The detailed parameters of training are as follows:

- **stage 1** All of the DNN architectures were trained using PyTorch with data parallelism over 8 Nvidia 3090 RTX GPUs. All data in section 2.1 were used for training.Therefore, the number of nodes in the last layer of the network was 255214. We used SGD with momentum (set to 0.9) with a batch size of 320. Four seconds slice of audio was selected to train our model for each speech segment.
  In this stage, we adopted ReduceLROnPlateau scheduler with a frequency of validating every 8,000 iterations, and the patience was 2. The minimum learning rate was 1.0e-6, and the decay factor is 0.5. Furthermore, the m_1 gradually increased from 0 to 0.2 [7] and m_2 gradually increased from 0 to 0.1. Then a $model_{base}$ was selected.
- **stage 2** The training data at this stage is NIST SRE04-10 + NIST SRE18 eval + NIST SRE19 eval, 4596 speakers in total. And the number of nodes in classifier is set to 4596. Initialize the network with the $model_{base}$. Since the number of nodes in the last layer of the current network is different from that of $model_{base}$, and the training data is a subset of all dataset, extract the weight of the 4596 speakers in $model_{base}$ as the initial weight of the current classifier. The chunk size is 1000, m_2 is 0, m_1 increases exponentially from 0.2 to 0.8 in 4000 iterations. The detection frequency of the validation is 2000 steps while the batchsize is 160. In this stage, we increased the chunk size to 1000. m_2 was 0, and m_1 increased exponentially from 0.2 to 0.8 in 4000 iterations. The detection frequency of the validation was 2000 steps while the batchsize was 160.

3.2. RepVGG

In our previous work, we have proved that the repvgg structure[8] is very effective in speaker recognition.We select RepVGG-A2 as our backbones. The model adopt 64 base channels. Other parts of the network structure and training methods are the same as ResNet.

3.3. Extended F-TDNN with additional modules

3.3.1. *Data Usage*

There are a huge amount of training data available for system development under open condition. Specially, training data includes SRE04-10, MIXER6, Switchboard (SWBD), Voxceleb 1&2 and Fisher datasets. These datasets (i.e., SRE, SWBD, Voxceleb) are augmented by different folds for different systems after convolving with far-field Room Impulse Responses (RIRs), or by adding noise from the MUSAN corpus. The operation of data augmentation is dependent on Kaldi x-vector recipe. Speaker filtering criterion is applied to training datasets. All speakers with less than 8 utterances and less than 400 frames per utterance were excluded for training.

3.3.2. *Architecture Description*

Inspired by the success of E-TDNN and F-TDNN, we explored a deeper and wider version of the F-TDNN in [9], called extended factorized TDNN (EF-TDNN). As for basic EF-TDNN architecture summarized in Table 1 without long short-term memory with projection (LSTMP) [10] layer, the 19th layer is fully connection layer. Instead of factorizing the TDNN layer into a convolution times a feed-forward layer, we have a F-TDNN layer with a constrained 2×1 convolution to dimension 256, followed by another constrained 2×1 convolution to dimension 256, followed by a 2×1 convolution back to the hidden-layer dimension (eg. 1024). The

Table 1. Extended Factorized TDNN with LSTMP

Layer	Layer Type	Context factor1	Context factor2	Context factor3	Skip conn, from layer	Size	Inner size
1	TDNN-ReLU	t-2 : t+2				512	
2	Dense-ReLU	t				512	
3	F-TDNN-ReLU	t-3, t-1	t-1, t+1,	t+1, t+3		1024	256
4	Dense-ReLU	t				1024	
5	F-TDNN-ReLU	t	t	t		1024	256
6	Dense-ReLU	t				1024	
7	F-TDNN-ReLU	t-5,t-2	t-2, t+1	t+1, t+4		1024	256
8	Dense-ReLU	t				1024	
9	F-TDNN-ReLU	t	t	t	5	1024	256
10	Dense-ReLU	t				1024	
11	F-TDNN-ReLU	t-5, t-2	t-2, t+1	t+1, t+4		1024	256
12	Dense-ReLU	t				1024	
13	F-TDNN-ReLU	t-5,t-2	t-2, t+1	t+1, t+4	3, 7	1024	256
14	Dense-ReLU	t				1024	
15	F-TDNN-ReLU	t-5, t-2	t-2, t+1	t+1, t+4		1024	256
16	Dense-ReLU	t				1024	
17	F-TDNN-ReLU	t	t	t	7, 11, 15	1024	256
18	Dense-ReLU	t				1024	
19	LSTMP	t				1024	
20	Dense-ReLU	t				2048	
21	Pooling (mean+stddev)	full-seq				2×2048	
22	Dense-ReLU					1024	
23	Dense-ReLU					1024	
24	Dense-Softmax					N spks.	

dimension now goes from, $1024 \rightarrow 256 \rightarrow 256 \rightarrow 1024$, within one layer. We adopted AMsoftmax with m = 0.15 as loss function for following systems.

- **EFTDNN_LSTMP** The architecture is shown in Table 1. Compared with basic EF-TDNN, the 19th layer is replaced by LSTMP [10], a recurrent layer. LSTMP is combination of two improved methods. One is to introduce a recurrent projection layer between the LSTM layer and the output layer. The other is to introduce another non-recurrent projection layer to increase the projection layer size without adding more recurrent connections. In our LSTMP layer, it has 1024-dimensional cell, 512 recurrent and non-recurrent projection dimensions. In addition, we also designed ETDNN_LSTMP based on ETDNN [11] in the same way, .

- **EFTDNN_SE** Squeeze-and-excitation (SE) block [12] includes two steps, namely squeeze and excitation operations. SE block only takes into account the importance of features on different channels to generate new feature map with channel attention. Further details about SE can be found in [12]. EFTDNN_SE system mainly changes the structure of F-TDNN layer by adding SE layer into it. Basic F-TDNN layer has three stages, but F-TDNN layer in EFTDNN_SE has four stages including a constrained 2×1 convolution to dimension 256, followed by another constrained 2×1 convolution to dimension 256, followed by a SE layer setting reduction ratio as 16, and then followed by a 2×1 convolution back to the hidden-layer dimension.

3.3.3. *Training Details*

Subsystems in this subsection were trained using Kaldi recipe and tensorflow. We used SGD setting momentum 0.9 as optimizer with a batch size of 128. Learning rate is set to 0.004. When loss on valid dataset did not decrease for 4 epochs, the learning rate halved. Stop training until the learning rate is less than 1.0e-6.

4. SCORING

Originally, we used plda as the backend of the CNN network and SRE18 eval + SRE19 eval as the backend training data. When we used a small-scale network, the results of the plda backend were better than the results of cosine. But with the increase of the model size and the addition of the refinement strategy, the result of plda degraded while the result os cosine score improved. Finally, the cosine result surpassed the one of plda. We used cosine distance for scoring.

5. FUSION AND CALIBRATION

All of our individual systems were calibrated using logistic regression on the SRE19 dev data. The fusions were equal weighted averages of the scores of the systems.

6. RESULTS

Table 2 presents the two subsystems trained for this challenge. Their performance is evaluated on the SRE18 dev CMN2 set and SRE20 progress set. All of our subsystems were calibrated independently using logistic regression on the SRE19 dev data. The fusion was an equal weighted average of the scores of the systems. The main problem identified in this challenge was the out-of-domain training speech corpora. The key to the problem was how to use the data when there was few matching data.

Table 3 presents the some subsystems trained with different acoustic features for this challenge. Their performance is evaluated on the SRE19 EVAL CMN2 set and SRE20 progress set. The final observation from experiments is that domain adaptation of PLDA and adaptive symmetric score normalization (AS-Norm) is not useful to reduce EER and min-DCF. As for the data for LDA/PLDA adaptation, it is achieved by filtering more adaptive datasets according to the results on the leaderboard. Fused systems in table 3 is the linear fusion of some subsystems by BOSARIS Toolkit. Before the fusion, each score is calibrated with PAV from BOSARIS toolkit [13] on SRE19 EVAL set.

7. REFERENCES

[1] Guoguo Chen David Snyder and Daniel Povey, "Musan: A music, speech, and noise corpus," *arXiv preprint arXiv:1510.08484*, 2015.

[2] Gilles Boulianne Lukas Burget Ondrej Glembek Nagendra Goel Mirko Hannemann Petr Motlicek Yanmin Qian

Table 2. Fused systems on the SRE21 DEV set.

System	scoring	SRE18 DEV CMN2		SRE20 PROGRESS SET			SRE20 TEST SET		
		eer	minc	eer	minc	act	eer	minc	act
ResNet74	cos	2.477	0.083	2.40	0.072	0.084	-	-	-
ResNet152	cos	2.528	0.084	2.37	0.066	0.085	-	-	-
RepVGG_b2	cos	2.560	0.103	2.12	0.068	0.082	-	-	-
fused	cos	-	-	2.23	0.063	0.076	**2.53**	**0.061**	**0.068**

Table 3. Performance of partial systems on the SRE19 EVAL CMN2 and SRE20 PROGRESS SET

System	Feature	SRE19 EVAL CMN2		SRE20 PROGRESS SET		
		EER(%)	min-DCF	EER(%)	min-DCF	act-DCF
EFTDNN_LSTMP	PLP	6.69	0.462	4.51	0.197	0.202
ETDNN_LSTMP	MFCC	5.73	0.443	4.51	0.197	0.202
EFTDNN_SE	MFCC	6.68	0.461	4.23	0.180	0.186
EFTDNN_SE	PLP	6.51	0.458	4.58	0.181	0.185
ResNet34	MFCC	5.32	0.417	3.11	0.146	0.293
Fused system	-	5.06	0.364	3.17	0.138	0.148

Petr Schwarz et al. Daniel Povey, Arnab Ghoshal, "The kaldi speech recognition toolkit," *IEEE 2011 workshop on automatic speech recognition and understanding. IEEE Signal Processing Society*, pp. number EPFL–CONF–192584, 2011.

[3] Shaoqing Ren Kaiming He, Xiangyu Zhang and Jian Sun., "Deep residual learning for image recognition," CVPR, 2016, p. 770–778.

[4] W. Liu F. Wang, J. Cheng and H. Liu, "Additive margin softmax for face verification," *IEEE Signal Processing Letters*, vol. 25, 2018.

[5] J. Guo J. Deng and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," *CoRR*, vol. abs/1801.07698, 2018.

[6] J. Guo J. Deng and S. Zafeiriou, "The idlab voxsrc-20 submission: Large margin fine-tuning and quality-aware score calibration in dnn based speaker verification," *ICASSP 2021-2021*, vol. abs/1801.07698, pp. 5814–5818, 2021.

[7] B. Desplanques J. Thienpondt and K. Demuynck, "The idlab voxceleb speaker recognition challenge 2020 system description," *arXiv preprint arXiv:2010.12468*, 2020.

[8] Miao Zhao, Yufeng Ma, Min Liu, and Minqiang Xu, "The speakin system for voxceleb speaker recognition challange 2021," *arXiv preprint arXiv:2109.01989*, 2021.

[9] Daniel Povey, Gaofeng Cheng, Yiming Wang, Ke Li, Hainan Xu, M. Yarmohammadi, and S. Khudanpur, "Semi-orthogonal low-rank matrix factorization for deep neural networks," in *INTERSPEECH*, 2018, p. 3743–3747.

[10] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," *ArXiv*, vol. abs/1402.1128, 2014.

[11] Jesus Villalba, Nanxin Chen, David Snyder, Daniel Garcia-Romero, Alan McCree, and etc, "The JHU-MIT System Description for NIST SRE18," in *NIST Speaker Recognition Evaluation Workshop*, 2018.

[12] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and E. Wu, "Squeeze-and-excitation networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, pp. 2011–2023, 2020.

[13] N. Brümmer and E. de Villiers, "The BOSARIS Toolkit: Theory, Algorithms and Code for Surviving the New DCF," *arXiv e-prints*, Apr. 2013.