THUEE SYSTEM DESCRIPTION FOR NIST 2019 SRE CTS CHALLENGE

Yi Liu, Tianyu Liang, Can Xu, Xianwei Zhang, Xianhong Chen, Wei-Qiang Zhang, Liang He *
Department of Electronic Engineering
Tsinghua University, Beijing, China

Dandan Song, Ruyun Li, Yangcheng Wu, Peng Ouyang, Shouyi Yin
Institute of Microelectronics,
Tsinghua University, Beijing, China

ABSTRACT
This paper describes the systems submitted by the department of electronic engineering, institute of microelectronics of Tsinghua university and TsingMicro Co. Ltd. (THUEE) to the NIST 2019 speaker recognition evaluation CTS challenge. Six subsystems, including etdnn/ams, ftdnn/as, eftdnn/ams, resnet, multitask and c-vector are developed in this evaluation.

Index Terms— NIST 2019 SRE CTS challenge, eftdnn, multitask, c-vector, additive margin

1. INTRODUCTION
This paper describes the systems developed by the department of electronic engineering, institute of microelectronics of Tsinghua university and TsingMicro Co. Ltd. (THUEE) for the NIST 2019 speaker recognition evaluation (SRE) CTS challenge [1]. Six subsystems, including etdnn/ams, ftdnn/as, eftdnn/ams, resnet, multitask and c-vector are developed in this evaluation. All the subsystems consists of a deep neural network followed by dimension deduction, score normalization and calibration. For each system, we begin with a summary of the data usage, followed by a description of the system setup along with their hyperparameters. Finally, we report experimental results obtained by each subsystem and fusion system on the SRE18 development and SRE18 evaluation datasets.

2. DATA USAGE
For the sake of clarity, the datasets notations are defined as in table 1 and the training data for the six subsystems are list in table 2, 3, and 4.

Table 1. Datasets Notations
| notation | datasets |
|----------|----------|
| SRE      | SRE04/05/06/08/10/MIXER6 |
| SWB      | LDC98S75/LDC99S79/LDC2002S06/LDC2001S13/LDC2004S07 |
| Voxceleb | Voxceleb 1/2 |
| Fisher+SWB I | Fisher + Switchboard I |
| CH+CF   | Callhome+Callfriend |

Table 2. Data usage for etdnn/ams, ftdnn/as, and resnet subsystems

| Components | Data usage |
|------------|------------|
| Neural Network | SRE+SWB+Voxceleb |
| LDA/PLDA | SRE+SRE16+SRE18 |
| PLDA-adapt | SRE+SRE16+SRE18 |
| asnorm | SRE18 unlabel |

Table 3. Data usage for multitask and c-vector subsystems

| Components | Data usage |
|------------|------------|
| GMM-HMM | Fisher+SWB I |
| Neural Network | SRE+SWB+Voxceleb+Fisher+SWB I |
| LDA/PLDA | SRE+SRE16+SRE18 |
| PLDA-adapt | SRE16+SRE18+MIXER6+CH+CF |
| asnorm | SRE18 unlabel |

Table 4. Data usage for eftdnn subsystem

| Components | Data usage |
|------------|------------|
| Neural Network | SRE+SWB+Voxceleb+CH+CF |
| LDA/PLDA | SRE+SRE16+SRE18 eval |
| PLDA-adapt | SRE+SRE16+SRE18 eval |
| asnorm | SRE18 unlabel |

3. SYSTEMS

3.1. Etdnn/ams
Etdnn/ams system is an extended version of tdnn with the additive margin softmax loss [2]. Etdnn is used in speaker verification in [3]. Compared with the traditional tdnn in [4], it has wider context and interleaving dense layers between each two tdnn layers. The architecture of our etdnn network is shown in table 5. It is the same as the etdnn architecture in [3], except that the context of layer 5 of our system is t-3:t+3 instead of t-3, t, t+3. The x-vector is extracted from layer 12 prior to the ReLU non-linearity. For the loss, we use additive margin softmax with $m = 0.15$ instead of traditional softmax loss or angular softmax loss. Additive margin softmax is proposed in [5] and then used in speaker verification in our paper [2]. It is easier to train and generally performs better than angular softmax.
3.2. ftdnn/as

Factorized TDNN (ftdnn) architecture is listed in table 6. It is the same to [3] except that we use 1024 nodes instead of 512 nodes in layer 12 and 13. The x-vector is extracted from layer 12 prior to the ReLU non-linearity. So our x-vector is 1024 dimensional. More details about the architecture can be found in [3].

### Table 6. ftdnn architecture

| Layer | Layer Type | Context | Conn. from | Size | Inner Size |
|-------|------------|---------|------------|------|-----------|
| 1     | TDNN       | t-2:t+2 |            | 512  |           |
| 2     | F-TDNN     | t       |            | 1024 | 256       |
| 3     | F-TDNN     | t       |            | 1024 | 256       |
| 4     | F-TDNN     | t       |            | 1024 | 256       |
| 5     | F-TDNN     | t       | 3          | 1024 | 256       |
| 6     | F-TDNN     | t       | t, t+3     | 1024 | 256       |
| 7     | F-TDNN     | t       | t, t+3     | 1024 | 256       |
| 8     | F-TDNN     | t       | t, t+3     | 1024 | 256       |
| 9     | F-TDNN     | t       | t, t+3     | 1024 | 256       |
| 10    | Dense      | t       |            | 2048 |           |
| 11    | Pooling    | full-seq|            | 4096 |           |
| 12    | Dense      |         |            | 1024 |           |
| 13    | Dense      |         |            | 1024 |           |
| 14    | Dense-Softmax | N. spks |          |      |           |

3.3. eftdnn/ams

Extended ftdnn (eftdnn) is a combination of etdnn and ftdnn. Its architecture is listed in table 7. The x-vector is extracted from layer 22 prior to the ReLU non-linearity.

3.4. resnet

ResNet architecture is also based on tdnn x-vector [4]. The five frame level tdnn layers in [4] are replaced by ResNet34 (512 nodes) + DNN(512 nodes) + DNN(1000 nodes). Further details about ResNet34 can be found in [6]. In our realization, acoustic features are regarded as a single channel picture and feed into the ResNet34. If the dimensions in the residual network don’t match, zeros are added. The statistic pooling and segment level network stay the same. For the loss function, we use angular softmax with \( m = 4 \). The x-vector is extracted from first DNN layer in segment level prior to the ReLU non-linearity. It has 512 dimensions.

3.5. multitask

Multitask architecture is proposed in [7]. It is a hybrid multi-task learning based on x-vector network and ASR network. It aims to introduce phonetic information by another neural acoustic model in ASR to help speaker recognition task. The architecture is shown in Fig. 1.

![Fig. 1. multitask architecture for the speaker embedding extraction.](image-url)
Table 7. eftdnn architecture

| Layer Type | Context factor 1 | Context factor 2 | Context factor 3 | conn. from | Size | Inner size |
|------------|-----------------|-----------------|-----------------|------------|------|-----------|
| 1          | TDNN            | t-2:t+2         | 512             |            |      |           |
| 2          | Dense           |                 | 512             |            |      |           |
| 3          | F-TDNN          | t-3,t-1         | t-1, t+1        | t+1, t+3   | 1024 | 256       |
| 4          | Dense           |                 | 1024            |            |      |           |
| 5          | F-TDNN          | t               | t               | t          | 1024 | 256       |
| 6          | Dense           |                 | 1024            |            |      |           |
| 7          | F-TDNN          | t-5, t-2        | t-2, t+1        | t+1,t+4    | 1024 | 256       |
| 8          | Dense           |                 | 1024            |            |      |           |
| 9          | F-TDNN          | t               | t               | t          | 1024 | 256       |
| 10         | Dense           |                 | 1024            |            |      |           |
| 11         | F-TDNN          | t-5, t-2        | t-2, t+1        | t+1,t+4    | 1024 | 256       |
| 12         | Dense           |                 | 1024            |            |      |           |
| 13         | F-TDNN          | t-5, t-2        | t-2,t+1         | t+1, t+4   | 1024 | 256       |
| 14         | Dense           |                 | 1024            |            |      |           |
| 15         | F-TDNN          | t-5, t-2        | t-2, t+1        | t+1,t+4    | 1024 | 256       |
| 16         | Dense           |                 | 1024            |            |      |           |
| 17         | F-TDNN          | t               | t               | t          | 1024 | 256       |
| 18         | Dense           | t               | 2048            |            |      |           |
| 19         | Dense           | t               | 2048            |            |      |           |
| 20         | Dense           | t               | 2048            |            |      |           |
| 21         | Pooling         | full-seq        | 4096            |            |      |           |
| 22         | Dense           |                 | 1024            |            |      |           |
| 23         | Dense           |                 | 1024            |            |      |           |
| 24         | Dense-Softmax   |                 | N. spks.        |            |      |           |

So, we need to train a GMM-HMM speech recognition system to do phonetic alignment for other datasets. The GMM-HMM is trained using Phonetic dataset with features of 20-dimensional MFCCs with delta and delta-delta, totally 60-dimensional. The total number of senones is 3800. After training, forced alignment is applied to the SRE, Switchboard, and Voxceleb datasets using a fMLLR-SAT system.

### 3.6. c-vector

C-vector architecture is also one of our proposed systems in paper [8]. As shown in Fig. 2, it is an extension of multitask architecture. It combines multitask architecture with an extra ASR Acoustic Model. The output of ASR Acoustic Model is concatenated with x-vector’s frame-level output as the input of statistics pooling. Refer to [8] for more details.

The multitask part of c-vector has the same architecture as in the above section 3.5. ASR Acoustic Model of c-vector is a 5-layer TDNN network. The slicing parameter is \{t-2; t-1; t; t+1; t+2\}, \{t-1; t; t+1\}, \{t-1; t; t+1\}, \{t-3; t; t+3\}, \{t-6; t-3; t\}. The 5-th layer is the BN layer containing 128 nodes and other layers have 650 nodes.

A GMM-HMM is also trained as like in section 3.5 to do phonetic alignment for training datasets.

### 4. FEATURE AND BACK-END

23-dimensional MFCC (20-3700Hz) is extracted as feature for etdnn/ams, ftdnn/as, eftdnn/ams, multitask and c-vector subsys-
| System   | SRE18 DEV | SRE18 EVAL |
|----------|-----------|------------|
|          | EER(%)    | min-DCF    | EER(%)    | min-DCF    |
| etdnn    | 3.95      | 0.222      | 2.59      | 0.198      |
| ftdnn    | 4.28      | 0.258      | 2.89      | 0.217      |
| eftdnn   | 3.67      | 0.196      | 2.56      | 0.204      |
| resnet   | 4.02      | 0.253      | 3.50      | 0.255      |
| multitask| 4.35      | 0.276      | 3.58      | 0.278      |
| c-vector | 3.92      | 0.252      | 3.10      | 0.249      |
| fused    | 3.45      | 0.164      | 2.25      | 0.175      |

23-dimensional Fbank is used as feature for ResNet 16kHz subsystems. A simple energy-based VAD is used based on the C0 component of the MFCC feature [9].

For each neural network, its training data are augmented using the public accessible MUSAN and RIRS_NOISES as the noise source. Two-fold data augmentation is applied for etdnn/ams, ftdnn/as, resnet, multitask and c-vector subsystems. For eftdnn/ams subsystem, five-fold data augmentation is applied.

After the embeddings are extracted, they are then transformed to 150 dimension using LDA. Then, embeddings are projected into unit sphere. At last, adapted PLDA with no dimension reduction is applied.

The execution time is test on Intel Xeon E5-2680 v4. Extracting x-vector cost about 0.087RT. Single trial cost around 0.09RT. The memory cost about 1G for a x-vector extraction and a single trial. In the inference, only CPU is used.

The speed test was performed on Intel Xeon E5-2680 v4 for etdnn/ams, multitask, c-vector and ResNet system. Test on Intel Xeon Platinum 8168 for ftdnn and eftdnn system. Extracting embedding cost about 0.103RT for etdnn/ams, 0.089RT for multitask, 0.092RT for c-vector, 0.132RT for eftdnn, 0.0639RT for ftdnn, and 0.112RT for ResNet. Single trial cost around 1.2ms for etdnn/ams, 0.9ms for multitask, 0.9ms for c-vector, 0.059s for eftdnn, 0.0288s for ftdnn, 1.0ms for ResNet. The memory cost about 1G for an embedding extraction and a single trial. In the inference, we just use CPU.

5. FUSION

Our primary system is the linear fusion of all the above six subsystems by BOSARIS Toolkit on SRE19 dev and eval [10]. Before the fusion, each score is calibrated by PAV method (pav_calibrate_scores) on our development database. It is evaluated by the primary metric provided by NIST SRE 2019.

6. REFERENCES

[1] “Nist 2019 speaker recognition evaluation: Cts challenge,” [online] Available at: https://www.nist.gov/itl/iad/mig/nist-2019-speaker-recognition-evaluation
[2] Yi Liu, Liang He, and Jia Liu, “Large margin softmax for speaker verification,” in INTERSPEECH, 2019, pp. 2873–2877.
[3] Jesus Villalba, Nanxin Chen, David Snyder, Daniel Garcia-Romero, Alan McCree, and etc, “The jhu-mit system descrip-
[4] David Snyder, Daniel Garcia-Romero, Daniel Povey, and Sanjeev Khudanpur, “Deep neural network embeddings for text-independent speaker verification,” in INTERSPEECH, 2017, pp. 999–1003.
[5] F. Wang, J. Cheng, W. Liu, and H. Liu, “Additive margin softmax for face verification,” IEEE Signal Processing Letters, vol. 25, no. 7, pp. 926–930, July 2018.
[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, “Deep residual learning for image recognition,” in CVPR, 2016, pp. 770–778.
[7] Yi Liu, Liang He, Jia Liu, and Michael T. Johnson, “Speaker embedding extraction with phonetic information,” in INTERSPEECH, 2018, pp. 2247–2251.
[8] Yi Liu, Liang He, Jia Liu, and Michael T. Johnson, “Introducing phonetic information to speaker embedding for speaker verification,” EURASIP Journal on Audio, Speech, and Music Processing, accept.
[9] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely, “The kaldi speech recognition toolkit,” in IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), 2011.
[10] N. Brümmer and E. de Villiers, “The BOSARIS Toolkit: Theory, Algorithms and Code for Surviving the New DCF,” arXiv e-prints, Apr. 2013.