The NetEase Games System Description for Text-dependent Sub-challenge of SDSVC 2020

Zhuxin Chen, Duisheng Chen, Hanyu Ding, Yue Lin

NetEase Games AI Lab

{chenzhuxin,gzchenduisheng,gzdinghanyu,gzlinyue}@corp.netease.com

Abstract

This document briefly describes the systems submitted to the text-dependent sub-challenge of SDSVC 2020 by NetEase Games AI Lab. The submitted primary system is a fusion of five x-vector systems and an ASR system. We improve the x-vector and PLDA pipeline for text-dependent speaker verification by changing the layer used to produce embeddings and modifying the back-end training strategies. ASR system is used for rejecting wrong trials. Finally, our primary system achieves minDCF=0.0456 and EER=1.52% on the evaluation set.

Index Terms: speech verification, x-vector, PLDA, SDSVC 2020, short utterance

1. SRE Systems

1.1. Data preparation

The training data that we used consisted of Voxceleb1, Voxceleb2 and the training partition of DeepMine dataset. Since there was no separate development data provided for this challenge, we splited the DeepMine training set into two parts, with 863 speakers (89,801 utterances) as training set and 100 speakers (11,262 utterances) as development set for tuning parameters.

1.2. Acoustic Features

Two kinds of features, namely MFCC and Fbank, were used in our experiments. We extracted 30-dimensional MFCC and 40-dimensional Fbank with a frame-length of 25ms. Cepstral mean subtraction was applied over a 3-second sliding window. An energy-based VAD is employed to filter out non-speech frames.

1.3. Neural network architecture

As mentioned in the abstract, our primary system contained five x-vectors. The following features and neural networks were used in our systems.

- **Fbank-FTDNN**: This system consists of the architecture of SpecAugment and FTDNN, as depicted in Table 3.
- **Fbank-FTDNN-OPGRU**: This system differs from the Fbank-FTDNN system that it adds two OPGRU layers with a delay of 3 frames.
- **Fbank-FTDNN-LSTM**: This system differs from the Fbank-FTDNN system that it adds two LSTM layers with a delay of 3 frames.
- **Fbank-CNN-FTDNN**: This system consists of the architecture of SpecAugment, CNN and FTDNN, as depicted in Table 4.
- **MFCC-CNN-FTDNN**: This system consists of the architecture of SpecAugment, CNN and FTDNN, as depicted in Table 5.

We used Kaldi [1] to train these systems, with a mini-batch size of 128, an initial learning rate of 0.001 and a final learning rate of 0.0001 for 6 epochs. Parallel training of the DNNs were up from 8 to 24 GPUs.

For the training data, we combined Voxceleb1, Voxceleb2 and the training subset of DeepMine dataset, totally about 8226 speakers. The data augmentation techniques described in [2] were applied.

1.4. Embedding extraction

Unlike the typical settings in text-independent speaker verification, we extracted embeddings with the following modifications:

- We used the standard deviation statistics in the pooling layer as embeddings for all systems.
- A data expansion strategy was used for the DeepMine dataset. We copied the feature three times after VAD and concatenated them together as most of the utterances were short-duration.

1.5. Back-end training strategy

We first trained the back-end models using text-dependent dataset. After extracting the embedding as mentioned above, a simple yet effective operation is to modify the training labels of LDA and PLDA models. While the text-independent speaker verification(TI-SV) system only uses the speaker tag, the text-dependent speaker verification(TD-SV) system combines speaker and phrase tag into a unique tag.

In our experiment, we used the training subset of DeepMine dataset, totally about 863 speaker with a fixed set of ten different phrases, to train the back-end models. There were 8630 classes of the back-end training data after modification. Similar to [3], the embeddings were centered, dimensionality reduced using LDA and length normalized. We adjusted the output dimension of LDA based on the results of the development set.

1.6. PLDA Adaptation

It is well known that the performance of speaker verification system benefits from large-scale in-domain data. However, it would be prohibitively expensive to collect large amount of in-domain data for every application, especially for TD-SV task. In this challenge, a PLDA adaptation method was applied that utilized both of Voxceleb dataset, which is a TI-SV labeled dataset, and DeepMine training set, which is a TD-SV labeled dataset.

As shown in Figure 1, the vital components of our back-end model include CORAL transform and PLDA interpolation. During training, the embeddings extracted from the TI-SV and TD-SV training set were centered respectively and two LDA models were trained. While the model with TI-SV data only made use of the speaker label, the other one with TD-SV data utilized both of the speaker and phrase label using the strategy described in section 1.5. The CORAL was used to minimize the covariance distance between the TI-SV and TD-SV data by whitening and re-coloring. We used the CORAL transformed vectors to train the PLDA. Finally, the adaptive PLDA was achieved by linearly interpolating the TI-SV covariance with the TD-SV covariance. For the CORAL transformation, it can be described as following equations:

$$C_{ti} = \mathrm{cov}(D_{ti}) + eye(size(D_{ti}, 2)) \quad (1)$$

$$C_{td} = \mathrm{cov}(D_{td}) + eye(size(D_{td}, 2)) \quad (2)$$

$$D'_{ti} = D_{ti} * C_{ti}^{-1/2} * C_{td}^{1/2} \quad (3)$$

Where D_{ti} and D_{td} are the length normalized (LN) vectors after LDA projection. D'_{ti} is the CORAL transformed vector of TI-SV dataset.

During test, the embeddings were centered and dimensionality reduced using the model trained from TD-SV data.

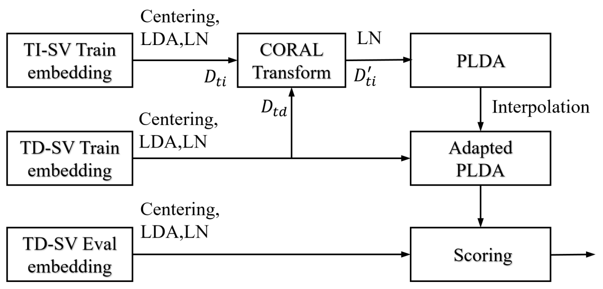

Figure 1: *Flow Diagram of PLDA Adaptation*

2. ASR System

2.1. Data preparation

The same as in section 1.1, we splited the DeepMine training set into two parts and only used the training subset to build the ASR system, which was used for predicting the phoneme sequence.

2.2. Acoustic modeling

The acoustic model was trained using lattice-free maximum mutual information (LF-MMI) criterion with Kaldi tool. We followed the Kaldi's librispeech recipe [1] to build this system but without using i-vector. The feature was 40-dimensional MFCC and there were 16 TDNNF layers in the acoustic model.

2.3. Decoding

In decoding phase, a phone based language model was built according to the transcription of ten phrases. We used Phone Error Rate(PER) as the score of ASR system.

3. Results

Tabel 1 shows the performance of x-vector based single systems on the evaluation set. As can be seen, our best x-vector system achieves MinDCF=0.0584 on the evaluation set.

Tabel 2 shows the performance of our submitted systems. For the single system, we combined the Fbank-CNN-FTDNN PLDA score and the ASR PER score. Here ASR system was used for rejecting wrong trials. In our experiment, if the PER was more than 45%, we would degrade the score. For the primary system, we made a fusion with five x-vector based systems as shown in Tabel 1, and then we combined the SRE and ASR scores. Finally, our primary system achieved minDCF=0.0456 and EER=1.52% on the evaluation set.

Table 1: *The result of x-vector based systems on the evaluation set.*

System	EER(%)	MinDCF
Fbank-FTDNN	1.89	0.0678
Fbank-FTDNN-OPGRU	1.77	0.0608
Fbank-FTDNN-LSTM	1.75	**0.0584**
Fbank-CNN-FTDNN	**1.67**	0.0626
MFCC-CNN-FTDNN	1.78	0.0642

Table 2: *The result of submitted systems on the evaluation set.*

System	EER(%)	MinDCF
single system	1.64	0.0546
primary system	**1.52**	**0.0456**

4. References

[1] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. CONF. IEEE Signal Processing Society, 2011.

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

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

[1] https://github.com/kaldi-asr/kaldi/blob/master/egs/librispeech/s5/local/chain/tuning/run_tdnn_1d.sh

Table 3: *The structure of Fbank-FTDNN, Fbank-FTDNN-LSTM and Fbank-FTDNN-OPGRU systems*

Layer	Layer Type	Context Factor 1	Context Factor 2	Skip Conn. from Layer	Size	Inner Size
1	BN-SpecAug	t			40	
2	TDNN-ReLU-BN	t-2:t+2			512	
3	FTDNN-ReLU-BN	t-2,t	t,t+2		1024	256
4	FTDNN-ReLU-BN	t	t		1024	256
5	FTDNN-ReLU-BN	t-3,t	t,t+3		1024	256
6	FTDNN-ReLU-BN	t	t	3	1024	256
7	FTDNN-ReLU-BN	t-3,t	t,t+3		1024	256
8	FTDNN-ReLU-BN	t	t	2,4	1024	256
9	FTDNN-ReLU-BN	t-3,t	t,t+3		1024	256
10	FTDNN-ReLU-BN	t	t	4,6,8	1024	256
11	FTDNN-ReLU-BN	t-3,t	t,t+3		1024	256
12	FTDNN-ReLU-BN	t	t	6,8,10	1024	256
13	FTDNN-ReLU-BN	t-3,t	t,t+3		1024	256
14	FTDNN-ReLU-BN	t	t	8,10,12	1024	256
15	None/LSTM*2/OPGRU*2	t			1024	
16	Dense-ReLU-BN	t			2000	
17	Pooling(mean+stddev)	Full-seq			2*2000	
18	Dense-ReLU-BN				512	
19	Dense-ReLU-BN				512	
20	Dense-Softmax				Num.spks.	

Table 4: *The structure of Fbank-CNN-FTDNN system.*

Layer	Layer Type	Kernel	Filters	Context Factor 1	Context Factor 2	Skip Conn. from Layer	Size	Inner Size
1	BN-SpecAug			t			40	
2	CNN-ReLU-BN	7*3	64				64*40	
3	Res-Block	3*3	64				64*40	
4	Res-Block	3*3	64				64*40	
5	CNN-ReLU-BN	3*3	128				128*20	
6	Res-Block	3*3	128				128*20	
7	Res-Block	3*3	128				128*20	
8	CNN-ReLU-BN	3*3	256				256*10	
9	Res-Block	3*3	256				256*10	
10	Res-Block	3*3	256				256*10	
11	FTDNN-ReLU-BN			t-1,t	t,t+1		512	
12	FTDNN-ReLU-BN			t-2,t	t,t+2		1024	256
13	FTDNN-ReLU-BN			t	t		1024	256
14	FTDNN-ReLU-BN			t-3,t	t,t+3		1024	256
15	FTDNN-ReLU-BN			t	t	13	1024	256
16	FTDNN-ReLU-BN			t-3,t	t,t+3		1024	256
17	FTDNN-ReLU-BN			t-3,t	t,t+3	12,14	1024	256
18	FTDNN-ReLU-BN			t-3,t	t,t+3		1024	256
19	FTDNN-ReLU-BN			t-3,t	t,t+3	14,16,18	1024	256
20	Dense-ReLU-BN			t			1536	
21	Pooling(mean+stddev)			Full-seq			2*1536	
22	Dense-ReLU-BN						512	
23	Dense-ReLU-BN						512	
24	Dense-Softmax						Num.spks.	

Table 5: *The structure of MFCC-CNN-FTDNN system.*

Layer	Layer Type	Kernel	Filters	Context Factor 1	Context Factor 2	Skip Conn. from Layer	Size	Inner Size
1	BN-SpecAug			t			30	
2	CNN-ReLU-BN	7*3	64				64*30	
3	Res-Block	3*3	64				64*30	
4	Res-Block	3*3	64				64*30	
5	CNN-ReLU-BN	3*3	128				128*15	
6	Res-Block	3*3	128				128*15	
7	Res-Block	3*3	128				128*15	
8	FTDNN-ReLU-BN			t-1,t	t,t+1		512	
9	FTDNN-ReLU-BN			t-2,t	t,t+2		1024	256
10	FTDNN-ReLU-BN			t	t		1024	256
11	FTDNN-ReLU-BN			t-3,t	t,t+3		1024	256
12	FTDNN-ReLU-BN			t	t	10	1024	256
13	FTDNN-ReLU-BN			t-3,t	t,t+3		1024	256
14	FTDNN-ReLU-BN			t-3,t	t,t+3	9,11	1024	256
15	FTDNN-ReLU-BN			t-3,t	t,t+3		1024	256
16	FTDNN-ReLU-BN			t-3,t	t,t+3	11,13,15	1024	256
17	Dense-ReLU-BN			t			1536	
18	Pooling(mean+stddev)			Full-seq			2*1536	
19	Dense-ReLU-BN						512	
20	Dense-ReLU-BN						512	
21	Dense-Softmax						Num.spks.	