The Team 20 System for Short-Duration Speaker Verification Challenge 2020

Tao Jiang[1], *Miao Zhao*[1], *Lin Li*[2], *Qingyang Hong*[1]

[1]School of Informatics, Xiamen University, China
[2]School of Electronic Science and Engineering, Xiamen University, China

qyhong@xmu.edu.cn, lilin@xmu.edu.cn

Abstract

In this paper, we present the Team 20 (T20) system for Task 1 in the Short-Duration Speaker Verification (SdSV) Challenge. We leveraged the system pipeline from three aspects, including the data processing, front-end training and back-end processing. In addition, we have explored some training strategies such as spectrogram augmentation and transfer learning. The experimental results show that the attempts we had done are effective and our best single system, a transfer model with spectrogram augmentation and attentive statistic pooling, significantly outperforms the official baseline on both progress subset and evaluation subset. Finally, a fusion of seven subsystems are chosen as our primary system which yielded 0.0856 and 0.0862 in term of minDCF, for the progress subset and evaluation subset respectively.

Index Terms: speaker recognition, text-dependent, data augmentation, transfer learning

1. Introduction

This paper describes the T20's submissions for the Task 1 of the Short-Duration Speaker Verification (SdSV) Challenge 2020 [1]. This challenge was split into two separate tracks: Task 1 and Task 2. Task 1 of the SdSV Challenge 2020 is the text-dependent (TD) speaker verification and Task 2 is the text-independent (TI) mode. Our team only participants the Task 1.

This paper is organized as follows: Section 2 describes the setup of the datasets for the challenge. Section 3 introduces the features and three types of DNN embedding-based speaker verification systems. The performance of our systems on both progress and evaluation subset is reported and analyzed in Section 4.

2. Datasets

2.1. Training Data

2.1.1. Task 1 In-Domain Data

For the Task1, the in-domain training data is the Part 1 of Deep-Mine dataset [2, 3], which contains more than 101,000 utterances from 963 speakers.

Since Task 1 only consider the TC as target and the rest will be considered as imposter, we no longer use the form of a label given by a speaker. In other words, the same speaker is distinguished by phrase labels, so the number of in-domain data's *spkid* increased from 963 to 8886.

2.1.2. Opening Dataset

Under the fixed condition of the challenge, we also used the following datasets in our submissions except the Task 1 in-domain data.

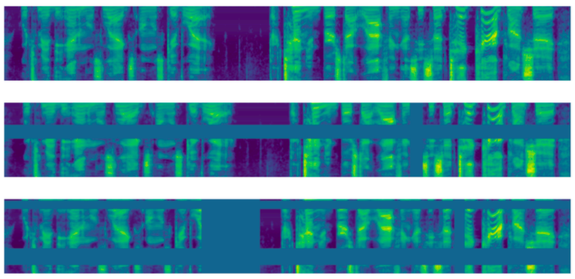

Figure 1: *SpecAugment: Multi-masking on acoustic feature*

- VoxCeleb-1 [4] : a dataset which contains more than 100,000 utterances for 1,251 celebrities, which are extracted from videos uploaded to YouTube.
- VoxCeleb-2 [5] : a dataset which contains more than 1,000,000 utterances for 6112 celebrities, which are extracted from videos uploaded to YouTube.

2.2. Development Dataset

In our systems, we extract 63 speakers from the Task 1 in-domain data to form the development set *(Dev)* since there is no separate development data provided for the challenge. Besides, the tags of the development set are treated like the Task 1 in-domain data mentioned above. As a result, the in-domain training set *(Train)* has only 900 speakers which we used to training models.

2.3. Augmentation Dataset

To obtain robust speaker embedding on SV systems we applied two augmentation strategies.

2.3.1. Traditional Augmentation

We utilized Kaldi toolkit [6] to add noise to the data and the augmentation datasets are Musan [7][1] and AIR[2] for room impulse response (RIR).

2.3.2. Spectrogram Augmentation

Inspired by the data augmentation from computer vision, S.Daniel et al proposed SpecAugment (SpecAug) [8], an augmentation method that operates on the log mel spectrogram of the input audio, rather than the raw audio itself as shown in Figure 1. Considering that the masking method adopted by SpecAug is zero-setting, there is a problem of mean shift dur-

[1]http://www.openslr.org/resource/17
[2]http://www.openslr.org/resource/28

ing training and testing. Therefore, we have performed mean correction on the frequency dimension.

3. Speaker Recognition System

ALL features are done by Kaldi toolkit and systems are trained by Pytorch toolkits [9].

3.1. Feature Extraction

Three features including Mel-frequency cepstral coefficient (MFCC), Filterbank features (Fbank) and Perceptual linear predictive features (PLP) are adopted in our systems. The settings of feature extraction are:

- 23-dimensional Kaldi MFCC with Pitch - 16kHz, frequency limits 40-7800Hz, 25ms frame length, 3-dimensional pitch.
- 40-dimensional Kaldi Fbank - 16kHz, frequency limits 40-7800Hz, 25ms frame length.
- 20-dimensional Kaldi PLP with Pitch - 16kHz, frequency limits 40-7800Hz, 25ms frame length, 3-dimensional pitch.

All the features were applied cepstral mean-normalization (CMN) with a sliding window of 3 seconds.

3.2. Speaker Model

3.2.1. TDNN Xvector

For the TDNN Xvector system, here are some details about TDNN Xvetor training:

- Using three features MFCC, PLP and Fbank.
- Except for baseline, all other models use SpecAug during model training. Besides, the frequency mask parameter and time mask parameter both are set to 0.2 and we use random rows and columns which set to 2 and 2 respectively.
- Using a warm restart technique for stochastic gradient descent [10] to improve training performance. And the T_{Max} and T_{multi} are set to 3 and 2 respectively.

3.2.2. ETDNN Xvector

Similar to the previous TDNN model, the ETDNN model also uses three features MFCC, PLP and Fbank and the setup for SpecAug and warm restart are the same with TDNN model.

3.2.3. Transfer Xvector

Lacking of a dataset which contains sufficient speakers and phrase information has always been a problem in the field of TD speaker verification. Therefore, we applied the transfer learning strategy on our experiments. Specifically, the fine-tuning steps are as follows:

1. Using the VoxCeleb $1 + 2$ to train the TI deep speaker model which based on TDNN architecture.
2. Using the parameters, obtained by the TI model, which not included the output layer's parameters, as TD model's initial parameters.
3. Using the $Train$ dataset tp train a new model.

To get a robust transferred model, the TI model we used should achieve a good performance on its source domain, which yields

Table 1: *The results of the base x-vector in Dev set with different PLDA parameters*

PLDA Back-End	*Dev*	
xvector_mfcc_baseline	EER%	minDCF
submean+norm	0.53	0.0614
submean+lda256+norm	0.58	0.0793
submean+whiten+norm	0.51	0.0621
submean+lda256+whiten+norm	0.58	0.0789

2.16% EER in test of VoxCeleb 1. In addition, since transfer model using the parameters obtained from TI model as initial parameters, the new model converges quickly. Therefore, the parameters of warm restart, T_{max} and T_{multi} are set to 3 and 1, respectively, after our experimental comparisons.

To further makes the transfer learning match the task of target domain, we experiment with an attentive pooling mechanism in transfer architecture. The strategy we adopt is similar to that proposed in [11].

4. Experiment

4.1. Back-End Processing

We used the Kaldi toolkit to train the PLDA model and the dataset is corresponding to the *Train* set. To get a better PLDA model, we used the *Train* to train the PLDA with different strategies such as submean, linear discriminant analysis (LDA) which reduced the dimension to 256, whiten and so on. As we can see from the Table 2, the x-vectors obtained from our baseline model are used to compare these strategies. We found that the PLDA model training with whiten, submean and normalization achieve a better result, while the usage of LDA degrade the performance. As a result, all of our submitted systems used submean, whiten and normalization for PLDA training.

4.2. Result and Analysis

4.2.1. Single Systems

We start with TDNN-Xvector on *Train* which outperform official baseline about 17% in minDCF for both progress subset and evaluation subset. Based on this, we verified the effectiveness of SpecAug. As we can see from the Table 3, the results of system 1 and system 2 confirmed that applying SpecAug to network training can further improve the performance, which yields about 5% relative improvement on both subsets. As a consequence, our subsequent systems all use this strategy during training.

Comparing the results of system 2, 4, 6 and system 3, 5, 7, respectively, it can be clearly seen that ETDNN-Xvector performs better TDNN-Xvector under the same training strategy. Besides, with the same architecture, we can also see the effect of three different acoustic features. The Fbank feature performs better than MFCC and PLP, which means that lower-level feature remains more raw information which makes it more helpful for DNN modeling.

Our best single system is the Transfer-Xvector-SpecAug-Attentive, which achieves 0.1016 minDCF and 2.48% EER on progress subset, and 0.1024 minDCF and 2.51% EER on evaluation subset. In Table 3, we also can draw the conclusion that using attentive pooling rather than average pooling is effective in transfer model. Attentive pooling yields 4.75% and 4.21% relative improvement on progress subset and evaluation subset,

Table 2: *The EER% and minDCF results of our single systems on the Task 1 of SdSV Challenge 2020 on the Progress subset and Evalution subset*

#	Feature	Systems	Progress subset		Evaluation subset	
			minDCF	EER%	minDCF	EER%
0	MFCC	official i-vector/HMM baseline [12]	0.1472	3.47	0.1464	3.49
1	MFCC	TDNN-Xvector	0.1210	3.05	0.1219	3.04
2	MFCC	TDNN-Xvector-SpecAug	0.1149	2.90	0.1156	2.92
3	MFCC	ETDNN-Xvector-SpecAug	0.1091	2.61	0.1089	2.64
4	PLP	TDNN-Xvector-SpecAug	0.1171	2.98	0.1176	3.02
5	PLP	ETDNN-Xvector-SpecAug	0.1078	2.66	0.1082	2.68
6	Fbank	TDNN-Xvector-SpecAug	0.1066	2.71	0.1077	2.72
7	Fbank	ETDNN-Xvector-SpecAug	0.1056	2.66	0.1059	2.67
8	MFCC	Transfer-Xvector-SpecAug	0.1067	2.63	0.1069	2.63
9	MFCC	Transfer-Xvector-SpecAug-Attentive	**0.1016**	**2.48**	**0.1024**	**2.51**

Table 3: *Performance of the average fusion system on The Progress subset and The Evaluation subset of Task 1 in SdSV Challenge 2020*

System	Progress subset		Evaluation subset	
	minDCF	EER	minDCF	EER
fusion 2+4+6	0.0979	2.53	0.0979	2.55
fusion 3+5+7	0.0923	2.37	0.9353	2.39
fusion 2+3+4+9	0.0904	2.34	0.0907	2.36
fusion 3+5+7+9	0.0863	2.22	0.0873	2.25
fusion 2+3+4+5+6+7+9	**0.0856**	**2.22**	**0.0862**	**2.24**

respectively.

4.2.2. Fusion Systems

The fusion strategy is average weight and the performance of the fused systems on the progress subset and evaluation set is shown in Table 4. Comparing to the best single system Transfer-Xvector-SpecAug-Attentive, all score fusion systems perform better, which show that the fusion system is more robust and stable. What's more, we can see from Table 4 that models under the same architecture are highly complementary on feature level. Finally, our best fusion system consists of 7 subsystems, which achieved 0.0856 and 0.0862 in minDCF for two subsets, respectively, which yields 15.7% and 15.8% relative improvements in term of minDCF for progress subset and evaluation subset over the best single system.

5. Conclusion

This paper presents the system submitted by T20 in the TD task of SdSV challenge 2020.

6. References

[1] H. Zeinali, K. A. Lee, J. Alam, and L. Burget, "Short-duration speaker verification (sdsv) challenge 2020: the challenge evaluation plan," *arXiv preprint arXiv:1912.06311*, 2019.

[2] H. Zeinali, H. Sameti, and T. Stafylakis, "Deepmine speech processing database: Text-dependent and independent speaker verification and speech recognition in persian and english." in *Odyssey*, 2018, pp. 386–392.

[3] H. Zeinali, L. Burget, J. Černocký *et al.*, "A multi purpose and large scale speech corpus in persian and english for speaker and speech recognition: the deepmine database," *arXiv preprint arXiv:1912.03627*, 2019.

[4] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: a large-scale speaker identification dataset," *arXiv preprint arXiv:1706.08612*, 2017.

[5] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," *arXiv preprint arXiv:1806.05622*, 2018.

[6] 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.

[7] D. Snyder, G. Chen, and D. Povey, "Musan: A music, speech, and noise corpus," *arXiv preprint arXiv:1510.08484*, 2015.

[8] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[9] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[10] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[11] K. Okabe, T. Koshinaka, and K. Shinoda, "Attentive statistics pooling for deep speaker embedding," in *INTERSPEECH*, 2018.

[12] H. Zeinali, H. Sameti, and L. Burget, "Hmm-based phrase-independent i-vector extractor for text-dependent speaker verification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. PP, pp. 1–1, 04 2017.