NSYSU+CHT Speaker Verification System for Far-Field Speaker Verification Challenge 2020

Yu-Jia Zhang[1], *Chia-Ping Chen*[1], *Chung-Li Lu*[2], *Bo-Cheng Chan*[2], *Shan-Wen Hsiao*[2]

[1]National Sun Yat-Sen University, Taiwan
[2]Chunghwa Telecom Laboratories, Taiwan

cpchen@mail.cse.nsysu.edu.tw, m083040025@student.nsysu.edu.tw
{chungli,cbc,swhsiao}@cht.com.tw

Abstract

In this paper, we describe the system **Team NSYSU+CHT** has implemented for the 2020 Far-field Speaker Verification Challenge (FFSVC 2020). The single systems are embedding-based neural speaker recognition systems. The front-end feature extractor is a neural network architecture based on TDNN and CNN modules, called **TDResNet**, which combines the advantages of both TDNN and CNN. In the pooling layer, we experimented with different methods such as statistics pooling and GhostVLAD. The back-end is a PLDA scorer. Here we evaluate PLDA training/adaptation and use it for system fusion. We participate in the text-dependent (Task 1) and text-independent (Task 2) speaker verification tasks on single microphone array data of FFSVC 2020. The best performance we have achieved with the proposed methods are minDCF 0.7703, EER 9.94% on Task 1, and minDCF 0.8762, EER 10.31% on Task 2.

1. Introduction

The automatic speaker verification (ASV) system has improved significantly with the advancement of deep learning technology. The relevant competitions held every year, whether it is NIST Speaker Recognition Evaluation (SRE)[1], or ASVspoof[2] which is to prevent spoofing attacks. These competitions have made the ASV system develop rapidly.

At present, the most widely used ASV system architecture is embedding-based which is a combination of front-end feature extractor and back-end scorer. The front-end extracts the raw input feature into a high-level representation at the frame-level layer, and integrates it into an utterance level representation through the pooling layer, then extracts embedding through the fully connected layer and classifies with classification heads. The front-end architecture has evolved from the traditional DNN / i-vector[3] to the time-delayed neural network (TDNN) / x-vector[4] and its extended version (E-TDNN)[5] that outperform in the speaker verification competition in recent years ; the convolutional neural network originally established based on image classification is considered to achieve good performance in the speaker recognition task, such as Residual neural network (ResNet)[6, 7, 8]. On the other hand, many studies focus on improving the pooling layer and loss function. In addition to the statistic pooling that is most commonly used in TDNN, self-attentive pooling[9, 10], NetVLAD[11] can make the system aggregating frame- level information better; the loss function is learned from the face recognition and tried many softmax varieties: L-softmax[12], A-softmax[13], AM-softmax[14], AAM-softmax[15]. In addition to the simple cosine similarity of back-end scorer, probabilistic linear discriminant analysis(PLDA)[16] has become one of the most popular methods.

In recent years, with the popularization of IoT devices and smart home products, the processing of short duration commands, and the usage of far-field scenario have become new challenges for the ASV system. The mismatch of recording devices has deepened the difficulty of verification. FFSVC2020[17] is held , which is used to promote the research of ASV system under this scenarios.

Therefore, this paper describe our implementation for FFSVC2020 in Task 1: Far-Field Text-Dependent Speaker Verification from single microphone array and Task 2: Far-Field Text-Independent Speaker Verification From single microphone array takes different solutions. We built our front-end embedding extractor with TDNN-based E-TDNN and CNN-based ResNet. Acoustic features use FBank with pitch. And then we also modified the ResNet architecture and combined it with E-TDNN to become a new network architecture called TDResNet. PLDA is our back-end scorer to evalute the performance of model architectures on each task separately. In addition, we also made changes for the pooling layer replacing the original statistic pooling with NetVLAD improvement: GhostVLAD [18]. More implementation details will be explained in subsequent sections.

2. Network Architecture

2.1. Frame Level Layers

In this section, we have adopted three different architectures. One is the extended version of the time delayed neural network (TDNN) called E-TDNN, and the other is the Resnet architecture of the convolutional neural network. Also, we found that both are complimentary, so we combined them into a new architecture called TDResnet.

2.1.1. TDNN-based

We establish the E-TDNN architecture as our baseline by referring to[5]. The architecture uses 10 layers as the frame-level feature extractor, and the concept of dilation is used in the 3rd, 5th, and 7th layers, which is the essence of TDNN. It used to extend the frame-level receptive fields. The dilation rates are 2, 3, and 4, respectively, so that 23 receptive fields can be finally perceived. Then, the frame-level output is passed through the statistic pooling and two fully connected layers. The frame-level information is integrated into segment-level information. Finally, the output is classified by classification head. Speaker embedding is taken from the first fully connected layers of the segment-level layer. Each layer is followed by Relu and batch normalization.

2.1.2. CNN-based

According to the research of [6, 7, 8], it shows that the architecture of Resnet has great feature extraction ability in the far-field environment with noisy and reverberate. Thus, we adopted Resnet as our CNN-based architecture which is inspired by the thin-ResNet architecture[6]. ResNet implement with fewer parameters, and then the output of the frame-level is also passed through the statistic pooling. Different from E-TDNN, the segment-level layer used only one fully connected layer. Speaker embedding is extracted from this layer. Each layer is also followed by Relu and batch normalization. Each residual block is connected with residual connect. The final architecture is shown in Table 1.

Table 1: *Network architecture of ResNet*

Module	Input 43 FBank-pitch($43 \times T$)	size
ConvModule	Conv1d, 1×43, 64	64
	$\begin{bmatrix} 1 & , & 48 \\ 3 & , & 48 \\ 1 & , & 96 \end{bmatrix} \times 2$	96
	$\begin{bmatrix} 1 & , & 64 \\ 3 & , & 64 \\ 1 & , & 128 \end{bmatrix} \times 3$	128
	$\begin{bmatrix} 1 & , & 128 \\ 3 & , & 128 \\ 1 & , & 256 \end{bmatrix} \times 3$	256
	$\begin{bmatrix} 1 & , & 256 \\ 3 & , & 256 \\ 1 & , & 512 \end{bmatrix} \times 3$	512
Statistic Pooling	Full-Seq	2×512
Segment	FC,512	512
AM-Softmax		Speakers

2.1.3. Combination Of TDNN And CNN

The difference between E-TDNN and ResNet is that the first few layers of E-TDNN are composed of convolution layers with dilation. Dilation is used to obtain a wide range of frame-level layer information, followed by several layers of the same kernel size convolutional layer to extract and integrate frame-level layer information. Although the layers of ResNet is deeper than E-TDNN, the receptive fields of the frame-level layer is not as wide as E-TDNN. Therefore, we believe that the two are complementary. In E-TDNN, if the ResNet block mechanism is used to replace the convolution layers after the dilation layer of the frame-level layers, it could become deeper and bigger, making the feature extraction ability better. According to this idea, mixed architecture was designed. The architecture is shown in the Table2. The first five layers use the original E-TDNN dilation design, the dilation rate are 2, 3, 4, 5 respectively, and the remaining layers use 3 residual blocks of different channel sizes respectively. There are 45 receptive fields can be finally perceived. The architecture retains the ability of E-TDNN to obtain the wide range of frame-level layer information and the ability of ResNet to extract features well.

In this paper[19], the authors proposed an architecture TDResBlock similar to our idea, but they chose to add TDNN module in the residual block. The difference is that we obtained fixed receptive fields from the first few layers with dilation. Then we use residual blocks for feature extraction and integration. But receptive fields is increasing with the number of residual blocks in their method. Every residual block has different receptive fields, so they need to integrate information from different receptive fields. Also, the number of dilations they use can eventually reach 11. If the number of frames is not long enough, using excessive dilations will make the result worse due to the zero padding. So, we only use 5.

Table 2: *Network architecture of TDResNet*

Module	Input 43 FBank-pitch($43 \times T$)	size
TDNN Module	$[t-2, t+2]$	512
	$\{t-2, t, t+2\}$	512
	$\{t-3, t, t+3\}$	512
	$\{t-4, t, t+4\}$	512
	$\{t-5, t, t+5\}$	512
ResNet Module	$\begin{bmatrix} 1 & , & 512 \\ 3 & , & 512 \\ 1 & , & 1024 \end{bmatrix} \times 3$	1024
	$\begin{bmatrix} 1 & , & 1024 \\ 3 & , & 1024 \\ 1 & , & 2048 \end{bmatrix} \times 3$	2048
Statistic Pooling	Full-Seq	2×2048
Segment	FC,512	512
AM-Softmax		Speakers

2.2. Pooling

In addition to the original statistics pooling by calculating mean and standard deviation, we use the GhostVLAD[7], which is improved from NetVLAD. NetVLAD takes the output of frame-level layers as input and then generates a vector V of size $K \times D$ by the following equation:

$$V(j,k) = \sum_{i=1}^{N} \frac{e^{a_k^T x_i + b_k}}{\sum_{k'=1}^{K} e^{a_{k'}^T x_i + b_{k'}}} (x_i(j) - c_k(j)) \quad (1)$$

K represents the total number of clusters, which is a hyper-parameter. D represents the size of each cluster, which is the same as the number of frame-level layers output channels. a_k, b_k, c_k are parameters trained by the network. The first half of the formula is softmax, which is the probability of x_i belong to the cluster k, the second half is to calculate the residual between x_i and the center of the cluster k. The softmax value is used as the weight of the residual, and then sums the results. Finally, the aggregated residuals of each cluster are concatenated into the final output vector V. The improvement of GhostVLAD is that it adds Ghost clusters G to the original K clusters for noise clustering. Original output will become $(K+G) \times D$, but finally exclude ghost clusters and using only $K \times D$. The original paper sets $K = 8$ and $G = 2$.

2.3. Loss Function

In recent years, the speaker recognition system based on Am-softmax loss function training has greatly improved compared to the traditional softmax[20], so we prefer to use Am-softmax rather than the original softmax. This loss introduces the concept of the angular interval into softmax and proposes a new margin. AM-softmax loss function is defined as follows:

$$L = -\frac{1}{n} \sum_{i=1}^{n} \log \frac{e^{s \cdot (\cos \theta_{y_i} - m)}}{e^{s \cdot (\cos \theta_{y_i} - m)} + \sum_{j=1, j \neq y_i}^{c} e^{s \cdot (\cos \theta_j)}} \quad (2)$$

where $\cos\theta_{y_i}$ represents the angle cosine of the feature vector and weight vector of the i sample, m represents the angular margin, and s is scaling factor that to scale the cosine value, m and s both are hyper-parameters, the goal of this loss is to maximize $\cos\theta_{y_i} - m$ to minimize the angle between the feature vector and the weight vector. We set $s = 30$, $m = 0.2$.

2.4. Back-end Scoring

2.4.1. Gaussian PLDA

The back-end scoring system is based on Gaussian PLDA, we first do mean normalization for the extracted speaker embedding to reduce the range of embedding variations, and then reduce the embedding dimension to 250 through LDA. We use the dimensionality reduction of embedding to train PLDA, and the adjustment of PLDA adaptation, and finally calculate the log-likehood score of embedding.

2.4.2. Score Fusion

Each system has a different embedding extractor architecture and scoring of PLDA or PLDA adaptation. To get the best system performance, we combined the scores calculated by multiple systems, we split a model into two subsystems: PLDA and PLDA adaptation subsystems, and uses BOSARIS toolkit[21] to calibrate the weights between different systems, as shown in Figure 1. Our calibration data set uses the FFSVC2020 development set.

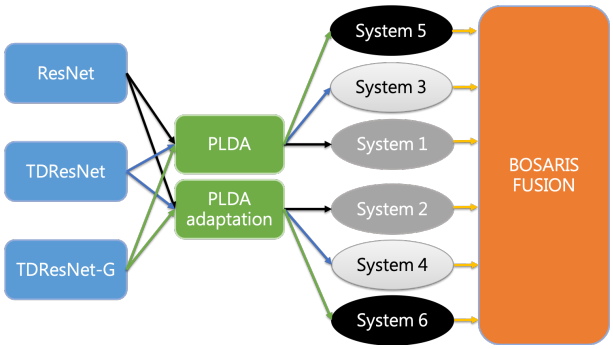


Figure 1: *Schematic illustration of fusion strategy.*

2.5. Model Fine-tuning

We use a lot of text-independent data to train our model to fit the conditions of Task2, but if directly applied to the Task1, the results will be very poor. The straightforward solution is to use text-dependent data retraining the embedding extractor or using the model which is trained by text-independent data as a pre-trained model to train in transfer learning method, but these two methods are bound to take a lot of extra time, so our approach is to choose to replace the PLDA training/adaptation data with text-dependent data. It not only save time, but also achieve a certain effect.

3. Experimental Setup

3.1. Training Data

Besides use the FFSVC 2020 training dataset provided by the challenge, and SLR85 (HI-MIA) mentioned in the evaluation plan. To meet the test scenario of this challenge, we have additionally selected Mandarin Chinese recordings data related to smart home. There are SLR33 (AISHELL), SLR38 (FreeST Chinese), and SLR68 (MAGICDATA). At the same time, to increase the diversity of speakers, we also used the SLR49 (Voxceleb) and SLR12 (LibriSpeech) data sets that are often used in other challenge. Therefore, a total of 7 different data sets were used.

3.2. Data Augmentation

The training data uses data augmentation, which has been used to enhance the robustness of the deep speaker embedding model, and the paper[22] mentioned that there is a mismatch between training data and test data in a far-field environment. In order to simulate the far-field environment, we use KALDI toolkit[23] to augment the training data with reverberation for several larger data sets, and use the data after the augmentation to train the network. Table3 is shown how we use data.

3.3. Acoustic Feature

Our acoustic features use Kaldi 40-dim FBank with 3-dim pitch, and the sample rate is 16kHz, the frame window is 25-ms, and the frame shift is 10-ms. Also, we use energy-based voice activity detection(VAD) to remove non-human voice fragments. Many experiments have shown that VAD has a great impact on the results, and then the cepstrum mean and variance normalization (CMVN) for features are used to reduce the effect of outlier features and imporve training efficiency.

3.4. Devlopment And Evaluation Data

We use the FFSVC2020 development data and evaluation data provided by the challenge. All experiments are tested by development data, and the results of the development set are used to estimate the performance on the evaluation set. So, the development set does not include in embedding or PLDA training, it only used to evaluate the models .

4. Result

Totally, we tested 5 different models on the development set, namely ETDNN, ResNet, TDResNet, TDResNet-G, and Fusion model. Only TDResNet, TDResNet-G, and Fusion model were tested on evaluation data and uploaded for the results. We used the Fusion model as the Primary System1, and this is also our best system in this challenge. The others are as Single System1 and Single System2, of which Single System2 performs better. All results are shown in Table 4.

4.1. Single System

The architecture of Single System1 is the same as described in Section 2.1.3. It is composed of 5 layers of TDNN layer and 6 residual blocks. The pooling uses statistics pooling. The loss

Table 3: *Data usage in the training process*

Dataset	Augmentation	Embedding Training	PLDA Training/Adaptation
LibriSpeech	√	√	
AISHELL	√	√	
FreeST		√	
VoxCeleb	√	√	
MAGICDATA		√	
HI-MIA		√	√
FFSVC2020 Training	√	√	√

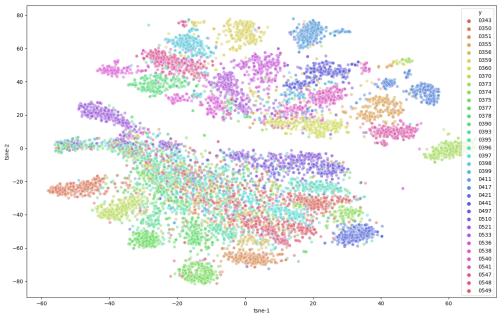
Table 4: *Minimum DCF and EER of the FFSVC2020 development set and evaluation set*

		Development Set								Evaluation Set			
		Task 1				Task 2				Task 1		Task 2	
		PLDA		PLDA Adapt		PLDA		PLDA Adapt					
ID	Model	minDCF	EER	minDCF	EER	minDCF	EER	minDCF	EER	minDCF	EER	minDCF	EER
1	E-TDNN	0.9286	11.94%	0.9231	11.82%	0.9503	12.86%	0.9479	12.43%	-	-	-	-
2	ResNet	0.8755	11.21%	0.8851	10.41%	0.9131	12.47%	0.9191	12.02%	-	-	-	-
3	TDResNet	0.8694	11.02%	0.8740	10.23%	0.9220	11.6%	0.93	10.88%	0.8566	11.45%	0.9132	11.36%
4	TDResNet-G	0.8374	11.59%	0.8295	10.33%	0.8650	12.11%	0.87	11.39%	0.8197	12.19%	0.8994	12.11%
5	Fusion(2+3+4)	-	-	-	-	-	-	-	-	0.7703	9.94%	0.8762	10.31%

function of the model is AM-softmax with m = 0.2 and s = 30. The training data of embedding extractor is used the 7 data sets mentioned in section 3.1, and PLDA / PLDA adaptation uses different data according to different tasks. In Task1, we use FFSVC2020 training data which ID is from one to thirty and HI-MIA, both recording content are 'ni hao, mi ya'. The purpose is to be close to the text-dependent condition, while Task2 uses all of FFSVC2020 training data. Regarding the training hyperparameters, the batch size is 32, the total number of frames per iteration is 7 billion, the initial learning rate is 0.001, and the final decrease to 0.0001. The model is trained for a total of 6 epochs by NVidia GeForce GTX 1080 Ti GPU.

Single System2 takes the same architecture as single system1, the only difference is that the pooling layer is replaced from the original statistics pooling to GhostVLAD. The training data and hyperparameters have not changed.

We also use the t-distributed stochastic neighbor embedding (t-SNE)[24] to visualize the high-dimensional embedding of Single System1 and Single System2 respectively, to evaluate the embedding learned from different pooling layers. The results are shown in Figure 2, we can find out that the embedding extracted by GhostVLAD is divided better than statistics pooling, especially in the upper half of the figure.



(a) TDResNet

(b) TDResNet-G

Figure 2: *The t-SNE visualization of the embeddings extracted from the different model embedding layer on FFSVC2020 development set.*

4.2. Primary System

Use the system after BOSARIS fusion as the Primary System1. We select Model 2,3,4 as front-end model, and each model corresponds to back-end PLDA and PLDA adaptation, so the fusion result is generated by 6 different subsystems. This fusion system is the best of all our systems, with minDCF 0.7703, EER 9.94% ranking 14 on Task1, and minDCF 0.8762, EER 10.31% ranking 11th on Task2.

5. Conclusions

In this study, we participated in FFSVC2020 and implemented front-end systems with TDNN-based and CNN-based. Also, we combined the advantages of both to design a new system called TDResNet. Back-end implements PLDA training/adaptation and used it for system fusion. Each system was evaluated on the development set and evaluation set of FFSVC2020. After evaluation, TDResNet outperformed the original two systems. Besides, we tested GhostVLAD and compared it with the original statistics pooling. We find out that GhostVLAD is better than statistics pooling. In the end, our best system can reach minDCF 0.7703, EER 9.94% on Task1, and minDCF 0.8762, EER 10.31% on Task2.

6. References

[1] NIST, "Nist speaker recognition evaluation," 2019. [Online]. Available: https://www.nist.gov/itl/iad/mig/nist-2019-speaker-recognition-evaluation

[2] M. Todisco, X. Wang, V. Vestman, M. Sahidullah, H. Delgado, A. Nautsch, J. Yamagishi, N. Evans, T. Kinnunen, and K. A. Lee, "Asvspoof 2019: Future horizons in spoofed and fake audio detection," *arXiv preprint arXiv:1904.05441*, 2019.

[3] M. McLaren, Y. Lei, and L. Ferrer, "Advances in deep neural network approaches to speaker recognition," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 4814–4818.

[4] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5329–5333.

[5] D. Snyder, J. Villalba, N. Chen, D. Povey, G. Sell, N. Dehak, and S. Khudanpur, "The jhu speaker recognition system for the voices 2019 challenge," in *Proc. Interspeech*, 2019, pp. 2468–2472.

[6] A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman, "Voxceleb: Large-scale speaker verification in the wild," *Computer Speech & Language*, vol. 60, p. 101027, 2020.

[7] W. Xie, A. Nagrani, J. S. Chung, and A. Zisserman, "Utterance-level aggregation for speaker recognition in the wild," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5791–5795.

[8] X. Qin, H. Bu, and M. Li, "Hi-mia: A far-field text-dependent speaker verification database and the baselines," *arXiv preprint arXiv:1912.01231*, 2019.

[9] K. Okabe, T. Koshinaka, and K. Shinoda, "Attentive statistics pooling for deep speaker embedding," *arXiv preprint arXiv:1803.10963*, 2018.

[10] Y. Zhu, T. Ko, D. Snyder, B. Mak, and D. Povey, "Self-attentive speaker embeddings for text-independent speaker verification." in *Interspeech*, 2018, pp. 3573–3577.

[11] J. Chen, W. Cai, D. Cai, Z. Cai, H. Zhong, and M. Li, "End-to-end language identification using netfv and netvlad," in *2018 11th International Symposium on Chinese Spoken Language Processing (ISCSLP)*. IEEE, 2018, pp. 319–323.

[12] W. Liu, Y. Wen, Z. Yu, and M. Yang, "Large-margin softmax loss for convolutional neural networks." in *ICML*, vol. 2, no. 3, 2016, p. 7.

[13] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 212–220.

[14] 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, 2018.

[15] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4690–4699.

[16] P. Kenny, "Bayesian speaker verification with heavy-tailed priors." in *Odyssey*, 2010, p. 14.

[17] X. Qin, M. Li, H. Bu, R. K. Das, W. Rao, S. Narayanan, and H. Li, "The ffsvc 2020 evaluation plan," *arXiv preprint arXiv:2002.00387*, 2020.

[18] Y. Zhong, R. Arandjelović, and A. Zisserman, "Ghostvlad for set-based face recognition," in *Asian Conference on Computer Vision*. Springer, 2018, pp. 35–50.

[19] S. Li, X. Lu, R. Takashima, P. Shen, T. Kawahara, and H. Kawai, "Improving very deep time-delay neural network with vertical-attention for effectively training ctc-based asr systems," in *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 77–83.

[20] Y. Liu, L. He, and J. Liu, "Large margin softmax loss for speaker verification," *arXiv preprint arXiv:1904.03479*, 2019.

[21] N. Brümmer and E. De Villiers, "The bosaris toolkit: Theory, algorithms and code for surviving the new dcf," *arXiv preprint arXiv:1304.2865*, 2013.

[22] X. Qin, D. Cai, and M. Li, "Far-field end-to-end text-dependent speaker verification based on mixed training data with transfer learning and enrollment data augmentation," *Proc. Interspeech 2019*, pp. 4045–4049, 2019.

[23] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The kaldi speech recognition toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, Dec. 2011, iEEE Catalog No.: CFP11SRW-USB.

[24] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.