The SpeakIn System for Far-Field Speaker Verification Challenge 2022

Yu Zheng[*], *Jinghan Peng*[*], *Yihao Chen*[*], *Yajun Zhang, Min Liu, Minqiang Xu*[†]

SpeakIn Technologies Co. Ltd.

{zhengyu, liumin, xuminqiang}@speakin.ai

Abstract

This paper describes speaker verification (SV) systems submitted by the SPEAKIN team to the Task 1 and Task 2 of the Far-Field Speaker Verification Challenge 2022 (*FFSVC2022*)[1]. SV tasks of the challenge focus on the problem of fully supervised far-field speaker verification (Task 1) and semi-supervised far-field speaker verification (Task 2).

In Task 1, we used *VoxCeleb* dataset and *FFSVC2020* dataset(train, dev and supplementary set) as training dataset. And for Task 2, we only used *VoxCeleb* dataset as training set. The subsystems, including fwSE-ResNet34-D, ResNet74, fwSE-ResNet101-D, ResNet152, ResNet221, RepVGG-A1, RepVGG-A2, RepVGG-B1 were developed in this evaluation. We also used Large-Margin Fine-tuning strategy. Sub-Mean and AS-Norm backend methods were used to solve the problem of domain mismatch. In the fusion stage, three models were fused in Task1 and two models were fused in Task2. On the *FFSVC2022* leaderboard, the EER of our submission is 3.0049% and the corresponding minDCF is 0.2938 in Task1. For Task2, EER and minDCF are 6.2060% and 0.5232 respectively.

Index Terms: speaker verification, FFSVC, domain adaptation

1. Introduction

All of our systems are deep-learning-based. Two different networks are used as encoders, namely ResNet and RepVGG. Based on two-dimensional (2D) convolution layer, the networks get state-of-the-art performance for far-field speaker recognition in the reverberant and noisy environment. The following sections describe the details of our systems.

2. Datasets

2.1. Training dataset

For Task 1, *VoxCeleb*[2, 3] and *FFSVC2020* dataset(train, dev and supplementary set) were used to perform system development. We here adopted a 3-fold speed augmentation at first to generate extra twice speakers. Each speech segment in this dataset was perturbed by 0.9 and 1.1 factor based on the SOX speed function. Then we obtained 18447 speakers.

For Task 2, only *VoxCeleb* data with speed perturbation was used to perform system development. There are 17982 speakers in this dataset. *FFSVC2020* dataset without speaker label information was used in the backend stage.

We applied the following techniques to augment each utterance:

- Reverberation: artificially reverberation using a convolution with simulated RIRs[4] from the AIR dataset
- Music: taking a music file (without vocals) randomly selected from MUSAN[5], trimmed or repeated as necessary to match duration, and added to the original signal (5-15dB SNR).
- Noise: MUSAN noises were added at one second intervals throughout the recording (0-15dB SNR).
- Babble: MUSAN speech was added to the original signal (13-20dB SNR).

We extracted 81-dimensional log Mel filter bank energies based on Kaldi. The window size is 25 ms, and the frame-shift is 10 ms. 200 frames of each feature were extracted without extra Voice Activation Detection (VAD). All features were cepstral mean normalized (CMN) in our training modes.

2.2. Development & Evaluation dataset

The development (Dev) data has the same data distribution as evaluation (Eval) data. All trial pairs of the development set and the evaluation set are single-channel speech segments. All enrollment utterances are close-talking speech segments recorded by telephone, while the test segments are close-talking or far-field audio recorded by tablet or telephone[1].

3. Systems

3.1. ResNet

As one of the most classical ConvNets, ResNet[6] has proved its power in speaker verification. In our systems, bottleneck-block-based ResNet (deeper structures:ResNet-74, ResNet-101, ResNet-152) are adopted. Base channels of all these ResNets are 64. We also implemented a deep and thin ResNet-221 structure which used ResNet_v2[7] bottleNeck with 32 base channels.

3.1.1. ResNet-D

ResNet-D[8] is a modification on the ResNet architecture that utilises an average pooling tweak for downsampling. The motivation is that in the unmodified ResNet, the 1×1 convolution for the downsampling block ignores 3/4 of input feature maps. Such modification will not lead to omission of information.

3.1.2. fwSE

A frequency-wise Squeeze-Excitation (fwSE) block[9], which injects global frequency information across all feature maps, is used in our system.

3.2. RepVGG

In our previous work, we have proved that the RepVGG, as one of the re-parameterized models, shows competitive performance in speaker recognition[10, 11] not only in computer vision field. We select RepVGG-A1, RepVGG-A2, RepVGG-B1

[*]These authors share equal contribution to this work.

[†]Corresponding author.

as our backbones in this challenge. All RepVGG models adopt 64 base channels.

3.3. Pooling Method

The pooling layer aims to aggregate the variable sequence to an utterance level embedding. In addition to global statistics pooling layer (GSP), we also used multi-query multi-head attention pooling mechanism layer (MQMHA) [10]. RepVGG-A1 and RepVGG-A2 is followed by GSP, and other backbones are followed by MQMHA.

3.4. Loss Function

Recently, margin based softmax methods have been widely used in speaker recognition works. To make a much better performance, we strengthen the AM-Softmax[12, 13] and AAM-Softmax[14] loss functions by two methods.

First, the subcenter method [15] was introduced to reduce the influence of possible noisy samples. The formulation is given by:

$$cos(\theta_{i,j}) = \max_{1 \le k \le K}(||x_i|| \cdot ||W_{j,k}||) \tag{1}$$

where the $\max$ function means that the nearest center is selected and it inhibits possible noisy samples interfering the dominant class center. K means the number of sub-centers for each speaker class, and k is the index of the sub-center.

Second, we proposed the Inter-TopK penalty to pay further attention to the centers which obtain high similarities comparing samples that do not belong to them. Therefore, it adds an extra penalty on these easily misclassified centers. Given a batch with N samples and C classes, the formulation with adding extra Inter-TopK penalty based on the AM-Softmax is:

$$\mathcal{L}_{AM'} = -\frac{1}{N}\sum_{i=1}^{N} log \frac{e^{s\cdot(cos\theta_{i,y_i}-m)}}{e^{s\cdot(cos\theta_{i,y_i}-m)} + \sum\limits_{j=1,j\neq y_i}^{C} e^{s\cdot\phi(\theta_{i,j})}} \tag{2}$$

where m is the original margin of AM-Softmax and s is the scalar of magnitude. We use the $\phi(\theta_{i,j})$ to replace the $cos\theta_{i,j}$ in the denominator:

$$\phi(\theta_{i,j}) = \begin{cases} cos\theta_{i,j} + m' & j \in \underset{1\le n\le C}{arg\,topK}(cos\theta_{i,n}) \\ cos\theta_{i,j} & Others. \end{cases} \tag{3}$$

where m' is an extra penalty which focuses on the closest K centers to the example x_i. And it is just the original AM-Softmax case when $m' = 0$. Similarity, the Inter-TopK penalty could be also added for AAM-Softmax loss function by replacing $cos\theta_{i,j} + m'$ by $cos(\theta_{i,j} - m')$.

4. Training Protocol

We used Pytorch[16] to conduct our experiments. For Task 1, some of our models were trained through two stages, and the others were trained through an additional stage. And on Task 2, all of our systems were trained through the first stage.

4.1. Stage 1: Pre-Training

In the first stage, we used all *VoxCeleb* data with speed perturbation, consisting of 17982 speakers. It should be noted that we do not use any *FFSVC2020* data at this stage. The number of classes is set to 17982 or 18447, depending on whether weights of speakers from *FFSVC2020* dataset are reserved. The SGD optimizer with a momentum of 0.9 and weight decay of 1e-3 was used. We used 8 GPUs with 128 mini-batch and an initial learning rate of 0.08 to train all of our models. 200 frames of each sample in one batch were adopted. We adopted ReduceLROnPlateau scheduler with a frequency of validating every 2,000 iterations, and the patience is 2. The minimum learning rate is 1.0e-6, and the decay factor is 0.1. All the models were trained with AM-Softmax in the first stage. Furthermore, the margin gradually increases from 0 to 0.2 [17].

We propose a novel training method, that is reserving the speaker weights in pre-training stage which needs to be learned for the fine-tuning stage. In the pre-training phase, there are no positive samples to train these reserved weights. When the number of classes is set to 17982, the weights of speakers of *FFSVC2020* dataset will be generated by random initialization for the second-stage training. However, preserving the speaker weights of *FFSVC2020* dataset in the stage 1, we could use the trained weights to initialize in the next fine-tuning stage. The experimental results confirm that this training method greatly improves the performance of the model, compared with the method of randomly initializing the weights of the last layer in traditional transfer learning. For specific experimental results, we can refer to the results of whether fwSE-ResNet34-D reserves weights in Table 1.

4.2. Stage 2: Fine-Tuning

We used *VoxCeleb* without speed perturbation and *FFSVC2020* dataset with speed perturbation to fine-tune all systems in the second stage. The training dataset consists of 6459 speakers. All the sets and hyper-parameters were the same as the first stage except the initial learning rate, which was set to 2e-5. We removed speed augmented part from the *VoxCeleb* data, discarded the corresponding weights at the same time. For this reason, the number of classes is changed to 6459.

4.3. Stage 3: Large-Margin Fine-Tuning

Large-Margin Fine-Tuning (LM-FT)[18] helps to further improve model performance for some of our models. We chose the second-stage model to fine-tune for an additional epoch. In the LM-FT stage, settings are slightly different from the second stage. Firstly, we only used the *FFSVC2020* dataset set as the training data, removing the speed augmented part from the training set to avoid domain mismatch. Secondly, we changed the chunk size from 200 to 400 and increased the margin exponentially from 0.2 to 0.5. The AM-Softmax loss was replaced by AAM-Softmax loss. We found that the large-margin-based fine-tuning in the third stage is not stable. For some large models, t extra large-margin-based fine-tuning after the second stage may make the model performance worse. As a result, we only do the third stage fine-tuning on some models, such as RepVGG-A1 and RepVGG-A2.

5. Backend

We used cosine distance for scoring in both Task 1 and Task 2. In addition, adaptive symmetric score normalization (AS-norm)[19] was used for Task 1, and Sub-Mean was used only for Task 2.

Table 1: *Performance on* FFSVC2022 *Dev&Eval set in Task 1. All systems used AS-Norm backend method. Reserved Weight is T means the number of classes is set to 18447 in the first stage, otherwise the number of classes is set to 17982. S7 and S8 systems were trained after the third stage, the others were only trained after the second stage.*

System Index	System	Reserved Weight	Dev		Eval	
			EER(%)	minDCF	EER(%)	minDCF
S1	fwSE-ResNet34-D	T	3.4278	0.3830	-	-
S2	fwSE-ResNet34-D	F	5.8806	0.5379	-	-
S3	ResNet74	T	3.3361	0.3621	-	-
S4	fwSE-ResNet101-D	T	3.0694	0.3319	-	-
S5	ResNet152	T	2.6667	0.2939	3.1897	0.3108
S6	ResNet221	T	2.9861	0.3237	3.3333	0.3307
S7	RepVGG-A1	T	3.8417	0.3910	4.1109	0.3933
S8	RepVGG-A2	T	3.5389	0.3536	3.7178	0.3651
S9	RepVGG-B1	T	3.2861	0.3247	-	-
Fusion						
S5,S6,S8			**2.5000**	**0.2735**	**3.0049**	**0.2938**

5.1. AS-Norm

For Task 1, AS-Norm was used for all of the models. For AS-Norm, we selected the original *VoxCeleb* and *FFSVC2020* dataset without any augmentation. The cohort was created by using speaker's utterance embedding vector as speaker center, consisted of 6149 speaker centers. Only part of the cohorts are selected to compute mean and standard deviation for normalization, and top-300 highest scores are selected for Task 1.

5.2. Sub-Mean

Sub-Mean was used for models trained in the first stage for Task 2. We randomly chose 40000 utterances from the *FFSVC2022* dataset, then extracting the embeddings to compute the mean embedding. The enrollment and test embeddings were both subtract the mean embedding before scoring:

$$s(\boldsymbol{x_e}, \boldsymbol{x_t}) = cos(\boldsymbol{x_e} - \bar{\boldsymbol{x}}, \boldsymbol{x_t} - \bar{\boldsymbol{x}}) \tag{4}$$

where $\boldsymbol{x_e}$, $\boldsymbol{x_t}$ are enrollment and test speaker embedding vectors respectively, and $\bar{\boldsymbol{x}}$ is the mean embedding vector of 40000 utterances randomly chosen from the *FFSVC2022* dataset.

5.3. Fusion

The results of all systems were fused using Logistic Regression on *FFSVC2022* Dev set. We got the weight of each system and then selected the dominant systems to assign weights artificially. In the end, fusion was performed by computing the weighted average of the scores of selected individual systems.

6. Results

Results of experiments on all our systems developed for the Task 1 and Task 2 of the challenge are displayed in Table 1 and Table2 respectively. The performance is measured on the *FFSVC2022* development and evaluation set in terms of Equal Error Rate (EER) and Minimum Detection Cost (minDCF) with a prior target probability, P_{tar} of 0.01. All systems in Table 1 are the results of models trained in stage 2 or stage 3 for Task 1, while all systems in Table 2 are the results of models trained in stage 1 only using *VoxCeleb* dataset for Task 2.

On Task 1, ResNet152 gets the best performance by both EER and minDCF, which has a 6.6924% EER and 0.5374 minDCF after AS-Norm calibration. The fused result used three single systems and get a 3.0049% EER and 0.2938 minDCF. The S1 system that reserves the speaker classification weights of *FFSVC2020* dataset in the first training stage gets a much lower minDCF value than the S2 system that does not.

Table 2: *Performance on* FFSVC2022 *Dev&Eval set in Task 2. All models were trained only using* VoxCeleb *dataset and all systems used Sub-Mean backend method.*

System Index	System	Dev		Eval	
		EER(%)	minDCF	EER(%)	minDCF
S1	fwSE-ResNet34-D	7.2444	0.5650	-	-
S2	fwSE-ResNet101-D	7.1472	0.5617	-	-
S3	ResNet152	6.4639	0.5132	6.6924	0.5374
S4	ResNet221	6.4417	0.5399	-	-
S5	RepVGG-A1	7.4306	0.5929	-	-
S6	RepVGG-A2	6.9583	0.5600	-	-
S7	RepVGG-B1	7.0861	0.5556	-	-
Fusion					
S3,S4		**5.9833**	**0.5004**	**6.2060**	**0.5232**

For Task 2, all the models were trained in the stage 1. Same as Task 1, the best individual system is ResNet152, which has a 6.6924% EER and 0.5374 minDCF. The fused result is 6.2060% EER and 0.5232 minDCF. *FFSVC2020* dataset was used in all systems to process Sub-Mean method.

7. Conclusions

We experimented multiple models on the SV, and ResNet achieved the best results on both tasks. Data augmentation, hyper-parameter changes in the fine-tune stage, and score normalization in the backend have all brought improvement. Experimental results show that the larger model outperforms the small model on both tasks of *FFSVC2022* challenge.

7.1. Task 1

Training in stages and using different training data and strategies in different stages can greatly improve model performance. The model trained using only *VoxCeleb* data will perform slightly worse than the model trained using *VoxCeleb* and *FFSVC2020* data in the first stage. But the former will perform far better than the latter after fine-tuning in the second stage.

From the experimental results we draw conclusions that reserving the weights of the speakers from *FFSVC2020* dataset and only using *VoxCeleb* dataset to train in the first stage makes the model perform much more better after fine-tuning in the sec-

ond stages. We speculated that the randomly initialized weights of speakers of *FFSVC2020* dataset are hard to converge possibly due to the small learning rate in the second stage.

We found that an additional large-margin-based fine-tuning after the second-stage fine-tuning can further improve the model performance, such as RepVGG-A1 and RepVGG-A2. However, our experimental results show that this method is unstable. For some large models, such as RepVGG-B1 and ResNet152, the refinement may make the model performance worse.

7.2. Task 2

For Task 2, the speaker label of far-field *FFSVC2020* dataset (in Mandarin) cannot be used in training. We only use close-talking *VoxCeleb* dataset (mostly in English) to train our models. Instead, the development set and evaludation set are far-field audio dataset. Therefore, there is a domain mismatch between training data and evaluation data. We focused on the solution of domain mismatch, and used Sub-Mean method to solve the problem. Our experimental results show that Sub-Mean is pretty useful for domain mismatch.

8. References

[1] X. Qin, M. Li, H. Bu, S. Narayanan, and H. Li, "Far-field speaker verification challenge (ffsvc) 2022 : Challenge evaluation plan," 2022.

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

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

[4] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5220–5224.

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

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[7] ——, "Identity mappings in deep residual networks," *CoRR*, vol. abs/1603.05027, 2016.

[8] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, "Bag of tricks for image classification with convolutional neural networks," *CoRR*, vol. abs/1812.01187, 2018.

[9] J. Thienpondt, B. Desplanques, and K. Demuynck, "The idlab voxceleb speaker recognition challenge 2021 system description," 2021. [Online]. Available: https://arxiv.org/abs/2109.04070

[10] M. Zhao, Y. Ma, M. Liu, and M. Xu, "The speakin system for voxceleb speaker recognition challange 2021," *arXiv preprint arXiv:2109.01989*, 2021.

[11] Y. Ma, M. Zhao, Y. Ding, Y. Zheng, M. Liu, and M. Xu, "Rep works in speaker verification," *CoRR*, vol. abs/2110.09720, 2021.

[12] 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.

[13] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, "Cosface: Large margin cosine loss for deep face recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5265–5274.

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

[15] J. Deng, J. Guo, T. Liu, M. Gong, and S. Zafeiriou, "Sub-center arcface: Boosting face recognition by large-scale noisy web faces," in *European Conference on Computer Vision*. Springer, 2020, pp. 741–757.

[16] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8026–8037, 2019.

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

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

[19] S. Cumani, P. Batzu, D. Colibro, C. Vair, P. Laface, and V. Vasilakakis, "Comparison of speaker recognition approaches for real applications." 08 2011, pp. 2365–2368.