The SpeakIn System Description for CNSRC2022

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

SpeakIn Technologies Co. Ltd.

{zhengyu, chenyihao, pengjinghan, zhangyajun, liumin, xuminqiang}@speakin.ai

Abstract

This report describes our submission to the task 1 and task 2 of the CNSRC2022. In fixed track, we only used CN-Celeb.T as training set. For open track, we added Wechat data. The subsystems, including ResNet74, ResNet101, RepVGG A2, ECAPA-TDNN were developed in this evaluation. We also used large-margin-based fine-tuning strategy. In backend, submean and asnorm were used. In task1 fixed track, our system was a fusion of 5 models, and 2 models were fused in task1 open track. And we used single system in task2.

Index Terms: speaker verification, CNSRC

1. Data

1.1. Training dataset

For Task 1 SV *fixed track*, ONLY *CN-Celeb.T* was used to perform system development. We here adopted a 3-fold speed augmentation [2, 3] at first to generate extra twice speakers. Each speech segment in this dataset was perturbed by 0.9 or 1.1 factor based on the SOX speed function. Then we obtained 8,235 speakers.

For Task 1 SV *open track* and Task 2 SR *open track*, in total, there are 245497 speakers in this dataset. The datasets used for training included:

1. Our own Chinese Wechat data (CHI-wechat).
2. VoxCeleb 1+2.
3. CN-Celeb.T.

We applied the following techniques to augment each utterance:

- Adding reverberation: artificially reverberation using a convolution with simulated RIRs from the AIR dataset
- Adding music: taking a music file (without vocals) randomly selected from MUSAN[1], trimmed or repeated as necessary to match duration, and added to the original signal (5-15dB SNR).
- Adding noise: MUSAN noises were added at one second intervals throughout the recording (0-15dB SNR).
- Adding Babble: MUSAN speech was added to the original signal (13-20dB SNR).

1.2. Feature

We extracted both 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 features were extracted without extra voice activation detection (VAD). All features were cepstral mean normalized in both our training modes.

2. Models

2.1. ResNet

As one of the most classical ConvNets, ResNet[2] has proved its power in speaker verification. In our systems, bottleneck-block-based ResNet (deeper structures:ResNet-74, ResNet-101) are adopted. All base channels of these ResNets are 64.

2.2. RepVGG

In our previous work, we have proved that the repvgg structure[3] is very effective in speaker recognition.We select RepVGG-A2 as our backbones. All models adopt 64 base channels.

2.3. Ecapa-tdnnL

The Ecapa-tdnn[4] structure is a modified Time Delay Neural Network structure and has been shown to perform well in speaker recognition. We use 2048 base channels in this competition.

2.4. Pooling method

The pooling layer aims to aggregate the variable sequence to an utterance level embedding. In addition to statistics pooling layer, was also used multi-query multi-head attention pooling mechanism layer (MQMHA) [3] to aggregate along the time across.

2.5. 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 [5, 6] and AAM-Softmax [7] loss functions by two methods.

First, the subcenter method [8] 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.

Secondly, 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 examples and a number of classes of C, the formulation with adding extra Inter-TopK penalty based on the AM-Softmax is:

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

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:

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

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 $cos(\theta_{i,j} - m')$.

2.6. Training Protocol

We used Pytorch [9] to conduct our experiments. All of our models were trained through two stages.

In the first stage, the SGD optimizer with a momentum of 0.9 and weight decay of 1e-3 (4e-4 for online training) was used. We used 8 GPUs with 1,024 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. Furthermore, the margin gradually increases from 0 to 0.2 [10].

In the large-margin-based fine-tuning stage [4], settings are slightly different from the first stage. Firstly, we removed the speed augmented part from the training set to avoid domain mismatch. Secondly, we changed the frame size from 200 to 1200 and increased the margin exponentially from 0.2 to 0.8. The AM-Softmax loss was replaced by AAM-Softmax loss. Thirdly, we choose the center with the largest vector length as the "dominated center" and discard the others. We only use this dominated center throughout the whole finetuning process. Finally, we adopted a smaller finetuning learning rate of 8e-5 and a 256 batch size. The learning rate scheduler is almost the same while the decay factor became 0.5.

3. Results

3.1. SV task

All our SV systems were described in Tabel 1 and Tabel2.

For the fixed track, our best single system is Resnet101, which has a 6.13% EER and 0.335 minDCF after submean and asnorm calibration. The fused result used all 5 single system and get a 5.95% EER and 0.3185 minDCF.

Tabel 1: Performance on CN-Celeb.E set in fixed track.

5***System**	**CN-Celeb.E**		**CN-Celeb.E (submean asnorm)**	
	eer	minc	eer	minc
ResNet34	7.8231	0.3755	7.4571	0.3518
Ecapa-tdnnL	8.8257	0.4086	8.6623	0.3914
ResNet74	7.7274	0.3785	7.3162	0.3475
RepVGG_A2	7.7387	0.3681	7.4233	0.3460
ResNet101	6.2518	0.3619	6.1335	0.3358
fused	-	-	5.9530	0.3185

For the open track, we only train 2 single system and best single system result is resnet101, which has a 5.02% EER and 0.2418 minDCF. The fused result is 4.66% EER and 0.2384 minDCF.

Tabel 2: Performance on CN-Celeb.E set in open track.

2***System**	**CN-Celeb.E**		**CN-Celeb.E (submean asnorm)**	
	eer	minc	eer	minc
RepVGG_A2	5.8519	0.2925	6.0321	0.2711
ResNet101	5.0577	0.2574	5.0183	0.2418
fused	-	-	4.6630	0.2384

3.2. SR task

Our SR systems are described is Table3. We developed 2 single system for SR task and Repvgg_A2 get a 52% mAP and Resnet101 has a 61% mAP. We used the Resnet101 result as our final submission.

Tabel 3: Performance on SR.eval set

2***System**	**SR.eval**
	mAP
RepVGG_A2	0.5203
ResNet101	0.6106

4. Conclusion

We experimented multiple models on the SV and SR tasks, and resnet achieved the best results on both tasks. Data augmentation, parameter changes in the finetune stage, and score normalization in the backend have all brought improvement.

5. References

[1] David Snyder, Guoguo Chen, and Daniel Povey, "Musan: A music, speech, and noise corpus," *arXiv preprint arXiv:1510.08484*, 2015.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[3] Miao Zhao, Yufeng Ma, Min Liu, and Minqiang Xu, "The speakin system for voxceleb speaker recognition challange 2021," *arXiv preprint arXiv:2109.01989*, 2021.

[4] Jenthe Thienpondt, Brecht Desplanques, and Kris Demuynck, "The idlab voxceleb speaker recognition challenge 2020 system description," *arXiv preprint arXiv:2010.12468*, 2020.

[5] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu, "Additive margin softmax for face verification," *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 926–930, 2018.

[6] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei 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.

[7] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos 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.

[8] Jiankang Deng, Jia Guo, Tongliang Liu, Mingming Gong, and Stefanos Zafeiriou, "Sub-center arcface: Boosting

face recognition by large-scale noisy web faces," in *European Conference on Computer Vision*. Springer, 2020, pp. 741–757.

[9] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al., "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8026–8037, 2019.

[10] Yi Liu, Liang He, and Jia Liu, "Large margin softmax loss for speaker verification," *arXiv preprint arXiv:1904.03479*, 2019.