

# The SJTU X-LANCE Lab System for CNSRC 2022

Zhengyang Chen, Bei Liu, Bing Han, Leying Zhang, Yanmin Qian

MoE Key Lab of Artificial Intelligence, AI Institute  
X-LANCE Lab, Department of Computer Science and Engineering  
Shanghai Jiao Tong University, Shanghai, China

{zhengyang.chen, beiliu, hanbing97, zhangleying, yanminqian}@sjtu.edu.cn

## Abstract

This technical report describes the SJTU X-LANCE Lab system for the three tracks in CNSRC 2022. In this challenge, we explored the speaker embedding modeling ability of deep ResNet (Deeper r-vector). All the systems are only trained on the Cnceleb training set and we use the same systems for the three tracks in CNSRC 2022. In this challenge, our best single system and fusion system achieve 0.3164 and 0.2975 minDCF respectively on the speaker verification task. Besides, we submit the result of ResNet221 to the speaker retrieval track and achieve 0.4626 mAP.

## 1. Data Usage

For all three tracks in the CNSRC 2022, we follow the same data usage setup. 797 speakers from CN-Celeb1 dev [1] and 1996 speaker from CN-Celeb2 [2] are used as the training data.

### 1.1. Data Processing

As shown in [2], there are many short utterances less than 2s in CN-Celeb dataset. In our experiment, we first concatenate the short utterances from the same genre and same speaker to make them longer than 5s. It should be noted that we only do this operation on the training set. After doing this operation, the training utterance number is reduced from 632,740 to 508,228.

### 1.2. Data Augmentation

In our experiments, three different augmentations are applied:

- Additive Noise: We use the audios from MUSAN [3] as the additive noise in our experiment to do data augmentation.
- Reverberation: The impulse response from RIR<sup>1</sup> is used to do reverberation data augmentation.
- Speed Perturbation: We randomly speech up or slow down an utterance with ratio 1.1 and 0.9 to do speed perturbation. The utterance with a new speed will be considered from a new speaker.

We do all the data augmentations online. For each utterance in the training process, we independently decide whether to do each data augmentation with a probability 0.6.

### 1.3. Acoustic Feature Extraction and Processing

In our experiment, we extract the 80-dimensional fbank feature for each utterance and then do the utterance-wise mean normalization along the time dimension. We did not apply voice activity detection (VAD) in our experiment.

<sup>1</sup><https://www.openslr.org/28/>

## 2. System Architecture

### 2.1. Revisiting r-vector

r-vector is introduced in [4] and is the winning system of VoxSRC 2019 [5]. Although different ResNet variants [6, 7, 8] are proposed for the speaker embedding learning task, they differ in various aspects such as the convolutional type (1D v.s.2D), network width, kernel size, pooling methods, etc. In our experiences, we found the r-vector [4, 9] is the most stable one considering of the performance and scalability. The 34-layer r-vector architecture used in this work is shown in Table 1, which is nearly the same as the original r-vector, while we use 80 dimensional Fbank instead of the 40 dimensional one. The ResNet backbone is used to transform the fbank feature into deep feature representations. Then, statistic pooling [10] is used to map the variable-length feature sequence to fix-dimensional representation.

Table 1: Model architecture for ResNet in r-vector

Layer name	Structure	Output
Input	–	$80 \times \text{Frame Num} \times 1$
Conv2D-1	$3 \times 3, \text{Stride } 1$	$80 \times \text{Frame Num} \times 32$
ResNetBlock-1	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 3, \text{Stride } 1$	$80 \times \text{Frame Num} \times 32$
ResNetBlock-2	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 4, \text{Stride } 2$	$40 \times \text{Frame Num} // 2 \times 64$
ResNetBlock-3	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 6, \text{Stride } 2$	$20 \times \text{Frame Num} // 4 \times 128$
ResNetBlock-4	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 3, \text{Stride } 2$	$10 \times \text{Frame Num} // 8 \times 256$
StatsPooling	–	$20 \times 256$
Flatten	–	5120
Emb Layer	–	256

### 2.2. Make r-vector deeper

It has been shown in [11] that impressive improvement can be achieved via deepening the speaker embedding learner, to make the r-vector more powerful, we extend it to a deeper version. The general configuration for these “deeper r-vector” is shown in Table 2.

We can change parameters  $(N_1, N_2, N_3, N_4)$  in Table 2 to get deep ResNet with different layers and we list the configuration for the deep ResNet used in our experiment in Table 3. Besides, we also use the DF-ResNet in [11] and make it deeper to 287 layers.

Table 2: **Model architecture for deep ResNet.**

Layer name	Structure	Output
Input	—	$80 \times \text{Frame Num} \times 1$
Conv2D-1	$3 \times 3$ , Stride 1	$80 \times \text{Frame Num} \times 32$
ResNetBlock-1	$\begin{matrix} 1 \times 1, 32 \\ 3 \times 3, 32 \\ 1 \times 1, 128 \end{matrix} \times N_1$ , Stride 1	$80 \times \text{Frame Num} \times 128$
ResNetBlock-2	$\begin{matrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{matrix} \times N_2$ , Stride 2	$40 \times \text{Frame Num} // 2 \times 256$
ResNetBlock-3	$\begin{matrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{matrix} \times N_3$ , Stride 2	$20 \times \text{Frame Num} // 4 \times 512$
ResNetBlock-4	$\begin{matrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{matrix} \times N_4$ , Stride 2	$10 \times \text{Frame Num} // 8 \times 1024$
StatisticPooling	—	$20 \times 1024$
Flatten	—	20480
Emb Layer	—	256

Table 3: **Configuration for different deep ResNet.**

Deep ResNet Name	$(N_1, N_2, N_3, N_4)$
ResNet152	(3, 8, 36, 3)
ResNet221	(6, 16, 48, 3)
ResNet293	(10, 20, 64, 3)

### 3. System Training

In our experiment, we split the training process into two stages. In the first stage, we do the general classification training. In the second stage, we do the large margin finetuning, which is first proposed in [12]. At each stage, SGD is used as the optimizer to update our models. The learning rate is exponentially decreased from an initial value to a final value.

#### 3.1. Stage I

In stage I, we use the additive angular margin (AAM) [13, 14] loss as the training objective. As mentioned in section 1.2, after applying speed perturbation, the speaker number is changed from 2793 to 8379. The scale and margin in the AAM loss are set to 32 and 0.2 respectively. We trained all the systems for 165 epochs. In each epoch, we go through the whole training set and randomly sample 2s segment from each utterance to build the training batch. In this stage, the initial value and final value of the learning rate are set to 0.1 and 0.00005 respectively.

#### 3.2. Stage II: Large Margin Finetuning

In stage II, we do the large margin finetuning [12] based on the model from stage I. In this stage, we abandon the speed perturbation augmentation and the speaker classification number is 2793. All the systems are trained for another 5 epochs. Besides, we randomly sampled 6s segments from each utterance to construct a training batch and change the margin in AAM to 0.5. In this stage, the initial value and final value of the learning rate are set to 0.0001 and 0.000025 respectively.

### 4. System Scoring

#### 4.1. Scoring for Speaker Verification

In our experiment, we use cosine similarity as the scoring method. Besides, we also add the adaptive score normalization [15] and we set the imposter cohort size to 600. The imposter cohort is estimated from the training set by average the embeddings for each training speaker. In the cnceleb evaluation trial, there are multiple utterances for each enrollment speaker.

We tried three different ways to leverage the multiple utterances during the scoring for each trial pair.

**Utt-Concat:** In this strategy, we concatenate multiple utterances, which belong to the same enrollment speaker, to one long enrollment utterance. Then, we can directly get the score between enrollment utterance and test utterance using the scoring method mentioned above.

**Emb-Avg:** In this strategy, we first extract the speaker embedding for each utterance. Then, we average the embeddings belonging to the same enrollment speaker to get enrollment embedding.

**Score-Avg:** In this strategy, for each utterance from the same enrollment speaker, we compute its score with the test utterance. Then, the scores are averaged to be used as the score between the enrollment speaker and the test utterance.

#### 4.2. Scoring for Speaker Retrieval

In the speaker retrieval task, we score each enrollment utterance against all the utterances in the large data pool. The scoring method is the same as the method used in the speaker verification task. Then, the utterances in the data pool with top 10 scores are considered as the retrieval results.

### 5. Results

#### 5.1. Speaker Verification

##### 5.1.1. Results comparison between different scoring strategies.

In this section, we will compare different ways to score the speaker verification system. The corresponding results are listed in Table 4. The results show that applying asnorm after cosine scoring can consistently bring further improvement. Actually, we also did the score calibration [12] in our experiment. However, the score calibration can only improve the EER but degrade the minDCF and we abandon it finally. Besides, we also compare the different ways to leverage the multiple utterances belonging to one enrollment speaker and we have given a detailed description of these methods in section 4.1. From the results, we find the Emb-Avg achieves the best result on the Cnceleb evaluation trial. The asnorm and Emb-Avg are applied to all the systems in the following sections.

Table 4: **Results comparison between different scoring methods and different strategies to combine multiple utterances within one enrollment speaker.** The results are from the ResNet34 model after stage I training.

Scoring Method	Enroll Comb	minDCF (0.01)	EER (%)
Cosine	Utt-Concat	0.4391	7.305
Cosine	Emb-Avg	0.4004	6.922
Cosine + ASnorm	Utt-Concat	0.4035	7.085
Cosine + ASnorm	Emb-Avg	<b>0.3707</b>	<b>6.590</b>
Cosine + ASnorm	Score-Avg	0.4419	6.759

##### 5.1.2. Results for different systems

Here, we listed results for all the systems in Table 5. As we expected, the deep ResNet has a strong modeling ability and can avoid overfitting problem at the same time. For the submission of the speaker verification task, we fuse all the systems with large margin finetuning based on their performance. Besides, we also list the false negative rate (FNR) and false positive rate (FPR) in Table 5. We get these two values by setting the score

Table 5: **Results for different systems.** The FNR and FPR denotes the false negative rate and false positive rate. We get the FNR and FPR by setting the score threshold to the same threshold when we get the minDCF (0.01). LM denotes the large margin finetuning.

System	Params #	minDCF (0.01)	EER (%)	FNR (%)	FPR (%)
ResNet34 *	6.63M	0.3958	7.981	35.29	0.043
ResNet34	6.63M	0.3707	6.590	31.73	0.054
ResNet152	19.8M	0.3386	5.762	29.34	0.045
ResNet221	23.8M	0.3270	<b>5.543</b>	28.08	0.046
ResNet293	28.6M	<b>0.3202</b>	5.553	<b>27.92</b>	<b>0.041</b>
DF-ResNet	14.8M	0.3361	6.279	28.83	0.048
ResNet34 + LM	6.63M	0.3543	6.221	30.06	0.054
ResNet152 + LM	19.8M	0.3251	5.452	28.66	0.039
ResNet221 + LM	23.8M	0.3179	5.284	28.27	<b>0.035</b>
ResNet293 + LM	28.6M	<b>0.3164</b>	<b>5.227</b>	27.82	0.038
DF-ResNet + LM	14.8M	0.3185	6.117	<b>27.46</b>	0.044
Fusion	-	0.2975	4.911	25.28	0.045

We didn't apply speed perturbation on ResNet34 \* as a comparison.

threshold to the same threshold value when we get minDCF (0.01). The results in Table 5 show that the minDCF is positively correlated with FNR and we can interpret the minDCF (0.01) as how the system performs on FNR in the case of very low FPR.

## 5.2. Speaker Retrieval

For the retrieval, we use the ResNet221+LM system in Table 5 to score each enrollment utterance against all the test utterances. Then we get the results with top 10 scores following the strategy described in section 4.2.

## 6. References

- [1] Yue Fan, JW Kang, LT Li, KC Li, HL Chen, ST Cheng, PY Zhang, ZY Zhou, YQ Cai, and Dong Wang, "CN-Celeb: a challenging chinese speaker recognition dataset," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7604–7608.
- [2] Lantian Li, Ruiqi Liu, Jiawen Kang, Yue Fan, Hao Cui, Yunqi Cai, Ravichander Vippera, Thomas Fang Zheng, and Dong Wang, "CN-Celeb: multi-genre speaker recognition," *Speech Communication*, 2022.
- [3] David Snyder, Guoguo Chen, and Daniel Povey, "MUSAN: A Music, Speech, and Noise Corpus," 2015, arXiv:1510.08484v1.
- [4] Hossein Zeinali, Shuai Wang, Anna Silnova, Pavel Matějka, and Oldřich Plchot, "But system description to voxceleb speaker recognition challenge 2019," *arXiv preprint arXiv:1910.12592*, 2019.
- [5] Joon Son Chung, Arsha Nagrani, Ernesto Coto, Weidi Xie, Mitchell McLaren, Douglas A Reynolds, and Andrew Zisserman, "Voxsrc 2019: The first voxceleb speaker recognition challenge," *arXiv preprint arXiv:1912.02522*, 2019.
- [6] Weidi Xie, Arsha Nagrani, Joon Son Chung, and Andrew 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.
- [7] Weicheng Cai, Jinkun Chen, and Ming Li, "Exploring the encoding layer and loss function in end-to-end speaker and language recognition system," *arXiv preprint arXiv:1804.05160*, 2018.
- [8] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman, "Voxceleb2: Deep speaker recognition," *arXiv preprint arXiv:1806.05622*, 2018.
- [9] Shuai Wang, Yexin Yang, Zhanghao Wu, Yanmin Qian, and Kai Yu, "Data augmentation using deep generative models for embedding based speaker recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2598–2609, 2020.
- [10] David Snyder, Daniel Garcia-Romero, Daniel Povey, and Sanjeev Khudanpur, "Deep neural network embeddings for text-independent speaker verification.," in *Interspeech*, 2017, pp. 999–1003.
- [11] Bei Liu, Zhengyang Chen, Shuai Wang, Haoyu Wang, and Yanmin Qian, "Df-resnet: Boosting speaker verification performance with depth-first design," *submitted to Interspeech 2022*, 2022.
- [12] Jenthe Thienpondt, Brecht Desplanques, and Kris Demuynck, "The idlab voxceleb speaker recognition challenge 2020 system description," *arXiv preprint arXiv:2010.12468*, 2020.
- [13] 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.
- [14] Xu Xiang, Shuai Wang, Houjun Huang, Yanmin Qian, and Kai Yu, "Margin matters: Towards more discriminative deep neural network embeddings for speaker recognition," in *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (AP-SIPA ASC)*. IEEE, 2019, pp. 1652–1656.
- [15] Sandro Cumani, Pier Domenico Bazu, Daniele Colibro, Claudio Vair, Pietro Laface, and Vasileios Vasilakakis,

“Comparison of speaker recognition approaches for real applications,” in *Twelfth annual conference of the international speech communication association*, 2011.