The Kriston AI System for the VoxCeleb Speaker Recognition Challenge 2022

Qutang Cai$^1$$^†$, Guoqiang Hong$^1$, Zhijian Ye$^1$, Ximin Li$^1$, Haizhou Li$^{1,2,3}$

$^1$Kriston AI Lab, China
$^2$The Chinese University of Hong Kong, Shenzhen, China
$^3$Department of Electrical and Computer Engineering, National University of Singapore, Singapore

{caiqt, hgq, yzj, lixm, haizhou.li}@kuaishang.com.cn

Abstract

This technical report describes our system for track 1, 2 and 4 of the VoxCeleb Speaker Recognition Challenge 2022 (VoxSRC-22). By combining several ResNet variants, our submission for track 1 attained a minDCF of 0.090 with EER 1.403%. By further incorporating three fine-tuned pre-trained models, our submission for track 2 achieved a minDCF of 0.072 with EER 1.119%. For track 4, our system consisted of voice activity detection (VAD), speaker embedding extraction, agglomerative hierarchical clustering (AHC) followed by a re-clustering step based on a Bayesian hidden Markov model and overlapped speech detection and handling. Our submission for track 4 achieved a diarisation error rate (DER) of 4.86%. The submissions all ranked the 2nd places for the corresponding tracks.

Index Terms: speaker verification, speaker recognition, speaker diarisation, ResNet, pre-trained models, VoxSRC-22

1. Introduction

The VoxSRC-22 challenge contains two full supervised speaker verification tracks (track 1 and track 2), and one diarisation track (track 4), where

track 1 is a closed task, and only VoxCeleb2 dev dataset can be used for training models;

track 2 and 4 are both open tasks, and any public data except the challenge test data can be used.

The goal of this challenge is to probe how well current methods can segment and recognize speakers from speech obtained ‘in the wild’.

For track 1, we trained from scratch six models modified from the ResNet architecture, using only VoxCeleb2 dev dataset. For track 2, we additionally fine-tuned three recently proposed pre-trained models [3][4], which are all publicly available, to harness the power of the large-scale pre-trained models. All the models in track 1 and 2 were trained and calibrated individually with the same procedure, and then fused using weighted linear combinations.

For track 4, we built our speaker diarization system by means of VAD, speaker embedding extraction, clustering, overlapped speech detection (OSD) and handling, step by step as shown in Figure 1.

2. Data preparation and augmentation

2.1. Training data

Track 1&2: For training, we used the VoxCeleb2 dev dataset which contains 1,092,009 utterances and 5,994 speakers in total.

Track 4: For validation, four development sets were used, including VoxCeleb1-O, VoxCeleb1-E, VoxCeleb1-H [11] and VoxSRC22-dev$^†$.

2.2. Features

Track 1&2: For track 1, we used mean normalized Kaldi-compliant log Mel-filter bank (FBank) features with energies with a 25 ms window size and a 10 ms frameshift. The feature dimensions were chosen from \{96, 104, 112, 120\} in our experiments. For fine-tuning models in track 2, we directly used the raw waveform. No additional voice activity detection (VAD) was used throughout this report.

Track 4: For VAD and OSD, we used mean normalized Kaldi-compliant 80-dim FBank and 30-dim MFCC features with energies with a 25 ms window size and a 10 ms frameshift.

† Corresponding author.
‡ Main contributor for track 4.

We used the cleaned trial lists of VoxCeleb1-O, -E and -H.

Figure 1: Diarisation system overview.
3. System description for track 1 and 2

3.1. Model architectures: track 1

**ResNet variants:** The models for track 1 were based on the ResNet architecture which is depicted in Figure 2, whose base channels were fixed to 64. Moreover, we only considered the basic Resnet block used in ResNet34 [2]. We modified the ResNet architecture with one or more of the strategies listed in Table 1 to introduce modelling diversity, and the resulting models are listed in Table 2. In Table 1:

- We only applied M3 and M4 to the first two stages of the backbone due to memory limits and the suggestions in [16].
- For M4, we used channel-wise and frequency-wise squeeze-excitation in [17, 16] to the residual connection, simultaneously. It’s worth mentioning that we additionally introduced bias items to the input which also depend on the input like the weights items.
- For M5, we altered the downsampling operation at the beginning of each stage from a 2-stride $2 \times 2$ convolution with a $2 \times 2$ average pooling operation.

![Base ResNet architecture](image)

**Figure 2:** Base ResNet architecture.

### Table 1: Strategies for modifying ResNet.

| Name | Description |
|------|-------------|
| M1   | Changing input feature dimension |
| M2   | Changing model depths |
| M3   | Changing kernel sizes |
| M4   | Using attention mechanisms [17, 16] |
| M5   | Using other downsampling operations [18] |

### Table 2: ResNet variants for Track 1.

| Name | M1 | M2 | M3 | M4 | M5 |
|------|----|----|----|----|----|
| R1   | X | ✓  | ✓  | X  | X  |
| R2   | X | ✓  | ✓  | ✓  | X  |
| R3   | X | ✓  | ✓  | ✓  | ✓  |
| R4   | X | ✓  | ✓  | ✓  | ✓  |
| R5   | X | ✓  | ✓  | ✓  | ✓  |
| R6   | X | ✓  | ✓  | ✓  | ✓  |

3.2. Model architectures: track 2

The models for Track 2 consisted of the models for Track 1 (see also [Model architectures: track 1] and three fine-tuned pre-trained models, including WavLM Large (WavLM-L) [4], Facebook’s Wav2Vec2 XLS-R 300M (XLSR-300M) and 1B (XLSR-1B) [5]. The hidden states of the pre-trained models were extracted using S3PRL1, and then normalized, linear weight combined, and fed to a downstream model similar to [4], where the downstream model was ECAPA-TDNN [21] with 1024 base channels and a 512-dimensional output. The resulting models are listed in Table 3 where STATS means the statistics pooling layer [22].

### Table 3: Fine-tuned pretrained models.

| Name | Upstream model | Pooling layer |
|------|----------------|---------------|
| P1   | WavLM-L        | SMHA          |
| P2   | XLSR-300M      | STATS        |
| P3   | XLSR-1B        | STATS        |

3.3. Training procedure

A two-stage training procedure like [7, 23] was adopted for training the models:

**Stage-1** Train initial models using short utterances to speedup the training process, where the short utterances were randomly cropped from the corresponding original ones with 2 and 2.24 seconds, respectively for track 1 and

[https://github.com/s3prl/s3prl](https://github.com/s3prl/s3prl)
Gradient accumulation technique was used to catch up when we were confronted with the hardware memory limits.

The basic fine-tuning steps are carried out as follows:

3.4. Fine-tuning pre-trained models

For the ResNet variants, the start learning rates were $10^{-4}$, updating frequency $3,000$, patience $4$, and decaying factor $0.4$). For the full model, we used AdamW (with weight decay $0.0001$) as the optimizer, and a ReduceLROnPlateau scheduler as the learning rate scheduler (with updating frequency $3,000$, patience $4$, and decaying factor $0.4$).

The training steps in Stage-1 are described as follows:

Step-1 Freezing the upstream model, train the downstream model, with a start learning rate of $3 \times 10^{-4}$.

Step-2 Train the self attention weights (in the upstream model) and the downstream model alternatively.

In Stage-2, we trained the entire models with a start learning rate of $2 \times 10^{-5}$.

For P3, we were hindered by the hardware memory limits; consequently, we trained only its self attention weights and the downstream model, alternatively. The training steps in Stage-1 are described as follows:

Step-1 Freezing the upstream model, train the downstream model, with a start learning rate of $3 \times 10^{-4}$.

Step-2 Train the self attention weights (in the upstream model) and the downstream model alternatively for two cycles:

Step-2.1 Freezing the model parameters except the self attention parts, train the self attention weights with a start learning rate of $4 \times 10^{-5}$.

Step-2.2 Freezing the upstream model, train the downstream model with a start learning rate of $3 \times 10^{-4}$.

The training steps in Stage-2 were also carried out similarly, training the self attention weights and the downstream model alternatively, except that the start learning rates were all set to $2 \times 10^{-5}$.

However, we had observed the tendency of overfit when fine-tuning the pre-trained models. Therefore, we saved model checkpoints after each epoch finished, and picked the one that performed best on the validation set for the final system.

3.5. Scoring procedure

When extracting the speaker embedding vectors, the $L_2$-normalized 512-dimensional outputs of the last full connected layer of each model were used. When performing single system scoring, we computed the cosine similarity score of the speaker embeddings of each trial, and then used adaptive score normalization (AS-Norm) and quality measure functions.

### Table 4: Single system evaluation results.

| System  | VoxCeleb1-O | VoxCeleb1-E | VoxCeleb1-H | VoxSRC22-dev |
|---------|-------------|-------------|-------------|--------------|
|         | EER(%)      | DCF$_{0.05}$ | EER(%)      | DCF$_{0.05}$ |
| R1      | 0.3510      | 0.0220      | 0.6077      | 0.0321       |
| R2      | 0.3776      | 0.0244      | 0.5860      | 0.0318       |
| R3      | 0.3616      | 0.0241      | 0.6205      | 0.0333       |
| R4      | 0.3457      | 0.0299      | 0.5739      | 0.0312       |
| R5      | 0.3829      | 0.0271      | 0.5788      | 0.0321       |
| R6      | 0.3297      | 0.0272      | 0.5771      | 0.0315       |
| P1      | 0.3615      | 0.0327      | 0.4705      | 0.0278       |
| P2      | 0.5797      | 0.0523      | 0.4977      | 0.0296       |
| P3      | 0.5159      | 0.0434      | 0.4525      | 0.0286       |

**Fusion**

| track1  | 0.2393 | 0.0209 | 0.4974 | 0.0266 | 0.8160 | 0.0452 | 1.3598 | 0.0977 |
| track2  | 0.2021 | 0.0153 | 0.3481 | 0.0286 | 0.6262 | 0.0354 | 1.0468 | 0.0760 |

1. Gradient accumulation technique was used to catch up when we were confronted with the hardware memory limits.
Table 5: The false alarm (FA), miss detection (MISS) and accuracy of the VAD model.

| System   | FA [%] | MISS [%] | Accuracy [%] |
|----------|--------|----------|---------------|
| FB Bank  | 3.49   | 1.49     | 95.00         |
| MFCC     | 4.27   | 0.92     | 94.80         |
| pyannote | 3.22   | 1.62     | 95.15         |
| Fusion   | 3.55   | 1.06     | 95.37         |

4.4.1. Initial Clustering

The speaker embeddings were clustered by means of AHC with cosine similarity. The AHC clustering threshold was tuned on track4-dev2, combined with Variational Bayes hidden Markov model (VB-HMM) diarisation.

4.4.2. Re-clustering

We replaced equation (17) and (18) in VB-HMM by (2) and (3):

$$L_s = I + \frac{F_A}{F_B} \sum_{t} \gamma_{ts}$$ (2)

$$p_t = x_t = F_C E_t$$ (3)

where $\gamma_{ts}$ is the marginal approximate posterior at frame $t$ for speaker $s$; $F_A = 0.3$, $F_B = 17$; $F_C$ is a scale parameter; $E_t$ is the L2-normalized speaker embedding at frame $t$; $I$ is a vector of 1s.

We also considered using AS-Norm for score calibration. For building cohorts used in AS-Norm, we randomly picked 2 utterances for each speaker from the VoxCeleb2 dev dataset, cropped them to 1.5 seconds and extracted their embeddings. We then replaced the $\alpha_t^s p_t$ and $\Phi$ terms in equation (23) in [33] by

$$\alpha_t^s p_t = \frac{F_A F_C^2}{F_B} \frac{1}{\sigma_s} \sum_{t} \gamma_{ts}$$ (4)

$$\Phi = I$$ (5)

where $\beta_s = \frac{1}{\sigma_s} \sum_{t} \gamma_{ts}$, $I_s = 1.0 + \frac{F_A}{F_B} \sum_{t} \gamma_{ts}$, and $\mu_s$ and $\sigma_s$ are mean and standard deviation of $\beta_s$.

Table 6: The DER and JER of the proposed speaker diarization system on track4-dev2.

| System       | DER [%] | JER [%] |
|--------------|---------|---------|
| VB           | 4.42    | 26.43   |
| VB+asnorm    | 4.29    | 26.81   |

4.4. Overlapped speech detection and handling

The overlap detection model, including its training process, were similar to that of the VAD model. We trained two models with the same structure and fused with pyannote 2.0. For each overlapped speech segments, we found the two closest speakers in time.

5. Experimental results

5.1. Track 1&2

We provide in Table 4 the single system results evaluated on the validation trial lists. The results in Table 4 show that although the single system performances are close to each other, the fused system’s can still achieve a considerable improvement, which also indicates the effectiveness of utilizing the diversities of the single systems. On the test trials of this challenge, the fused system achieved a minDCF of 0.090 and an EER of 1.401% for track 1, and achieved a minDCF of 0.072 and an EER of 1.119% for track 2, where the testing results were all closed to the validation results on the VoxSRC22-dev dataset.

5.2. Track 4

The diarisation results of the proposed systems are shown in Table 6. The system VB+asnorm was our best system. Compared with the system VB, DER was improved by 0.13%, but the JER was deteriorated by 0.38%. Our best submission on the evaluation set attained DER 4.86% and JER 25.48%.
6. References

[1] J. S. Chung, A. Nagrani, and A. Zisserman, “VoxCeleb2: Deep speaker recognition,” in *Proc. Interspeech*, 2018, pp. 1086–1090.

[2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *IEEE/CVF CVPR*, 2016, pp. 770–778.

[3] A. Babu, C. Wang, A. Tjandra, K. Lakhotia, Q. Xu, N. Goyal, K. Singh, P. von Platen, Y. Saraf, J. Pinto, A. Baevski, A. Conneau, and M. Auli, “XLNet: Self-supervised cross-lingual speech representation learning at scale,” 2021. [Online]. Available: https://arxiv.org/abs/2111.09296

[4] S. Chen, C. Wang, Z. Chen, Y. Wu, S. Liu, Z. Chen, J. Li, N. Kanda, T. Yoshioha, X. Xiao, J. Wu, L. Zhou, S. Ren, Y. Qian, Y. Qian, J. Wu, M. Zeng, X. Yu, and F. Wei, “Wavlm: Large-scale self-supervised pre-training for full stack: speech processing,” 2021. [Online]. Available: https://arxiv.org/abs/2110.13000

[5] H. Yamamoto, K. A. Lee, K. Okabe, and T. Koshimaka, “Speaker Augmentation and Bandwidth Extension for Deep Speaker Embedding,” in *Proc. Interspeech*, 2019, pp. 406–410.

[6] W. Wang, D. Cai, X. Qin, and M. Li, “The dku-dukeee systems for voxceleb speaker recognition challenge 2020,” 2020. [Online]. Available: https://arxiv.org/abs/2010.12731

[7] M. Zhao, Y. Ma, M. Liu, and M. Xu, “The speakin system for voxceleb speaker recognition challenge 2021,” 2021. [Online]. Available: https://arxiv.org/abs/2109.01969

[8] 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. Veselý, “The kaldiu speech recognition toolkit,” in *IEEE ASRU*, 2011.

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

[10] 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 *Proc. ICASSP*, 2017, pp. 5220–5224.

[11] A. Nagrani, J. S. Chung, and A. Zisserman, “VoxCeleb: A Large-Scale Speaker Identification Dataset,” in *Proc. Interspeech 2017*. 2017, pp. 2616–2620.

[12] J. S. Chung, J. Huh, A. Nagrani, T. Afouras, and A. Zisserman, “Spot the conversation: Speaker diarisation in the wild,” in *Proc. Interspeech*, 2020.

[13] S. O. Sadjadi, “Nist sre cts superset: A large-scale dataset for telephony speaker recognition,” 2021. [Online]. Available: https://arxiv.org/abs/2108.07118

[14] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “Librispeech: An asr corpus based on public domain audio books,” in *Proc. ICASSP*, 2015, pp. 5206–5210.

[15] J. Du, X. Na, X. Liu, and H. Bu, “Aishell: Transforming mandarin asr research into industrial scale,” 2018. [Online]. Available: https://arxiv.org/abs/1808.10583

[16] M. Rouvier and P.-M. Bousquet, “Studying squeeze-and-excitation used in cnn for speaker verification,” in *IEEE ASRU*, 2021, pp. 1110–1115.

[17] J. Thiennpant, B. Desplanques, and K. Demuynck, “Integrating Frequency Translational Invariance in TDNNs and Frequency Positional Information in 2D ResNets to Enhance Speaker Verification,” in *Proc. Interspeech*, 2021, pp. 2302–2306.

[18] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, “Bag of tricks for image classification with convolutional neural networks,” in *IEEE/CVF CVPR*, 2019, pp. 558–567.

[19] M. India, P. Safari, and J. Hernando, “Self Multi-Head Attention for Speaker Recognition,” in *Proc. Interspeech*, 2019, pp. 4305–4309.

[20] X. Zhang, X. Zhou, M. Lin, and J. Sun, “Shufflenet: An extremely efficient convolutional neural network for mobile devices,” in *IEEE/CVF CVPR*, 2018, pp. 6848–6856.

[21] B. Desplanques, J. Thiennpant, and K. Demuynck, “ECAPA-TDNN: Emphasized Channel Attention, Propagation and Aggregation in TDNN Based Speaker Verification,” in *Proc. Interspeech*, 2020, pp. 3830–3834.

[22] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, “X-vectors: Robust dnn embeddings for speaker recognition,” in *Proc. ICASSP*, 2018, pp. 5329–5333.

[23] J. Thiennpant, B. Desplanques, and K. Demuynck, “The IDLab VoxSRC-20 submission: Large margin fine-tuning and quality-aware score calibration in DNN based speaker verification,” in *Proc. ICASSP*, 2021.

[24] 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.

[25] D. Deng, J. Guo, N. Xue, and S. Zafeiriou, “Arcface: Additive angular margin loss for deep face recognition,” in *IEEE/CVF CVPR*, 2019, pp. 4685–4694.

[26] P. Matejka, O. Novotný, O. Plchot, L. Burget, M. Diez, and J. Černocký, “Analysis of score normalization in multilingual speaker recognition,” in *Proc. Interspeech*, 2017, pp. 1567–1571.

[27] S. Cumani, P. Batzu, D. Colibrò, C. Vair, P. Lafaèce, and V. Vasilakakis, “Comparison of speaker recognition approaches for real applications.” in *Proc. Interspeech*, 2011, pp. 2365–2368.

[28] M. I. Mandasari, R. Saeidi, M. McLaren, and D. A. van Leeuwen, “Quality measure functions for calibration of speaker recognition systems in various duration conditions.” *IEEE TASLP*, vol. 21, no. 11, pp. 2425–2438, 2013.

[29] W. Wang, D. Cai, Q. Lin, L. Yang, J. Wang, J. Wang, and M. Li, “The dku-dukeee-levono system for the diarization task of the 2021 voxceleb speaker recognition challenge,” 2021. [Online]. Available: https://arxiv.org/abs/2109.02002

[30] H. Bredin, R. Yin, J. M. Coria, G. Gelly, P. Korshunov, M. Lavechin, D. Fustes, H. Titeux, W. Bouaziz, and M.-P. Gill, “pyannote.audio: neural building blocks for speaker diarization,” 2019. [Online]. Available: https://arxiv.org/abs/1911.01255

[31] X. Xiao, N. Kanda, Z. Chen, T. Zhou, T. Yoshioha, S. Chen, Y. Zhao, G. Liu, Y. Wu, J. Wu, J. Liu, J. Li, and Y. Gong, “Microsoft speaker diarization system for the voxceleb speaker recognition challenge 2020,” 2020. [Online]. Available: https://arxiv.org/abs/2010.11458

[32] F. Landini, O. Glembek, P. Matějka, J. Rohdin, L. Burget, M. Diez, and A. Silnova, “Analysis of the but diarization system for voxconverse challenge,” 2020. [Online]. Available: https://arxiv.org/abs/2010.11718

[33] F. Landini, J. Profant, M. Diez, and L. Burget, “Bayesian hmm clustering of x-vector sequences (vbs) in speaker diarization: theory, implementation and analysis on standard tasks,” 2020. [Online]. Available: https://arxiv.org/abs/2012.14952