The JHU-MIT System Description for NIST SRE19 AV

Jesús Villalba[1], *Daniel Garcia-Romero*[2], *Nanxin Chen*[1], *Gregory Sell*[2], *Jonas Borgstrom*[3], *Alan McCree*[2], *David Snyder*[1,2], *Saurabh Kataria*[1], *L. Paola García-Perera*[1], *Fred Richardson*[3], *Pedro A. Torres-Carrasquillo*[3], *Najim Dehak*[1]

[1]Center for Language and Speech Processing, Johns Hopkins University, Baltimore, MD, USA
[2]Human Language Technology Center of Excellence, Johns Hopkins University, Baltimore, MD, USA
[3] MIT Lincoln Laboratory, Lexington, MA, USA

{jvillalba,dgromero,bobchennan,gsell,dsnyder,alan.maccree,paola.garcia,ndehak3}@jhu.edu,
{jonas.borgstrom,frichard,ptorres}@ll.mit.edu,

Abstract

This document represents the SRE19 AV submission by the team composed of JHU-CLSP, JHU-HLTCOE and MIT Lincoln Labs. All the developed systems for the audio and video conditions consisted of Neural network embeddings with some flavor of PLDA/cosine back-end. Primary fusions obtained Actual DCF of 0.250 on SRE18 VAST eval, 0.183 on SRE19 AV dev audio, 0.140 on SRE19 AV dev video and 0.054 on SRE19 AV multi-modal.

1. Introduction

The JHU-MIT submission is the joint effort of the teams at Johns Hopkins CLSP and HLTCOE, and MIT Lincoln Laboratory (MIT-LL). We submitted systems for the audio, video, and audio-visual conditions.

For the audio condition, the systems consisted of F-TDNN, E-TDNN or ResNet x-vectors followed by some form of PLDA/cosine scoring classifier plus adaptive score normalization. More in detail, all systems followed these steps:

1. Acoustic feature extraction (MFCC).
2. Voice activity detection.
3. Diarization.
4. Embedding extraction.
5. LDA dimensionality reduction.
6. Centering, whitening and length normalization.
7. PLDA log-likelihood ratio evaluation/Cosine scoring.
8. Adaptive score-normalization (all system used AS-Norm unless said otherwise).
9. Fusion/calibration.

For the video condition, we followed these steps:

1. Face detection on video frames.
2. Extract one embedding per each detected face.
3. Back-end based on cosine scoring.
4. Optional adaptive score normalization.
5. Fusion/calibration.

Multi-modal systems (primary and contrastive), were obtained by adding the scores of the corresponding audio and video systems.

2. Training datasets

2.1. Individual datasets

The datasets used for training included:

- NIST SRE12 phonecalls recorded through far-field microphone (SRE12-micphn). We did not use interviews to avoid dealing with the interviewer removal.
- MIXER6 microphone phonecalls (MX6-micphn).
- VoxCelebCat 1+2: the original distribution of VoxCeleb split each video into multiple short excerpts. We concatenated all excerpts from the same video into one file. This makes the dataset more appropriate for PLDA training and also helps to balance the weight of each video in the embedding training.
- SITW-dev-core: single speaker segments from the Speakers in the Wild development set.
- DihardII: 10-60 seconds segments extracted from the DihardII dev and eval distributions. The ground truth RTTMs where used to create ground truth VAD to get the speech of a single speaker per segment.
- SITW-dev-test-diarized. Segments obtained from diarizing the SITW dev test set.
- SRE18-dev-VAST-diarized: SRE18 VAST development set. For enrollment segments we used diarization marks provided by the organizer. For the test segments, we used the segments obtained by our diarization system.

2.2. JHU-CLSP Training data

The dataset combinations used in the systems from the JHU-CLSP-MITLL team were:

- CLSP-Train-SRE18: This is the training set that we used to train x-vector in the previous SRE18 eval. This set included SRE12-micphn, MX6-micphn, VoxCelebCat and SITW-dev-core. It contained 436815 recordings from 7936 speakers. This data was augmented with noise or/and reverberation $\times 2$ and added to the original dataset to get $\times 3$ the original dataset size.
- CLSP-Train-JSALT: This is the set we used to train x-vectors during JSALT2019 workshop and included VoxCelebCat 1+2. We augmented with noise or reverberation $\times 4$ and added to the original dataset to the $\times 5$ the original dataset size. Finally, it contained 833k utterances from 7185 speakers.

- CLSP-Train-SRE19: This set only included VoxCeleb-Cat 1+2. We augmented with noise or/and reverberation $\times 5$ and added to the original dataset to the $\times 6$ the original dataset size. Finally, it contained 1M utterances from 7185 speakers.
- CLSP-PLDA: This is CLSP-Train-SRE19 where we removed speakers with less than 40 recordings (after augmentation) and we removed utterances from speakers with more than 80 utterances. We did this to balance the number of utterances by speaker. We also removed utterances with less than 15 seconds and more than 800 seconds, to match SRE18 VAST eval durations. This dataset had 400k utterances from 5728 speakers.
- SITW-SRE18-dev-diarized: SITW-dev-test-diarized + SRE18-dev-VAST-diarized. It was used to center SRE8 and SRE19 video data.
- SITW-SRE18-DihardII-diarized: SITW-SRE18-dev-diarized + DihardII. It was used for adaptive S-Norm.

Impulse responses for augmentation were obtained from the Aachen impulse response database (AIR)[1]. Noises were from the MUSAN corpus[2]. We used the same SNR levels as in the Kaldi recipes.

2.3. JHU-HLTCOE Training data

For our speaker system submission we trained our DNN on VoxCeleb-1-2. In total, there are 7185 speakers in this dataset, with 7 million utterances after augmentation.

2.3.1. Data Augmentation

To augment an utterance, we randomly pick from one of the following strategies:

- **Reverb:** Artificially reverberate via convolution with simulated RIRs from the AIR dataset
- **Music:** A single music file (without vocals) is randomly selected from MUSAN, trimmed or repeated as necessary to match duration, and added to the original signal (5-15dB SNR).
- **Noise:** MUSAN noises are added at one second intervals throughout the recording (0-15dB SNR).
- **Babble:** Three to seven speakers are randomly picked from MX6-micphn, summed together, then added to the original signal (13-20dB SNR)
- **Codec:** If the file is from VoxCelebCat, simulate GSM AMR phone encoding[3]

2.4. MITLL Training data

The x-vector DNN was trained on data from VoxCeleb1 and VoxCeleb2. After removing utterances with less than 5 seconds of speech and speakers with less than 8 utterances, this set included 1.1M utterances from 7.2k speakers. Data augmentation was then performed by adding noise and reverberation, resulting in 3.9M utterances. Specifically, data augmentation was performed according to the following recipe:

- **Reverb:** Room impulse responses were randomly chosen from the *small room* and *medium room* subsets of the MUSAN corpus.
- **Noise:** Noise signals were randomly chosen from the MUSAN corpus, and added at 1 second intervals. The noise sigals were mixed at SNRs randomly chosen from the set $\{0, 5, 8, 10, 13, 15\}$ dB.
- **Music:** A single music signal was randomly chosen from the MUSAN corpus, and mixed at an SNR randomly chosen from the set $\{5, 8, 10, 15\}$ dB.
- **Babble:** Babble signals were created by first randomly selecting speech files from the MUSAN corpus, and then mixing 3-7 individual signals. The resulting babble noise was mixed at an SNR randomly selected from the set $\{10, 13, 15, 17, 20\}$ dB.

A separate training set was designed for training the backend. This set was comprised of VoxCelebCat, with data augmentation, resulting in 290k utterances from 7.2k speakers.

3. Development datasets

The development datasets were used to train fusion and calibration; and measure performance. For the audio condition we used SRE18 VAST eval, since it had much more trials than the SRE19 AV dev (only 1 false alarm at $P_{\mathcal{T}} = 0.05$). For the video part, we used SRE19 AV dev. We didn't use Janus dataset for calibration, since we observed domain mismatch between JANUS and SRE19 AV dev.

4. Acoustic features

JHU-CLSP and MITLL used 40 dimension MFCC (40 Mel filters) for x-vectors based on time-delay networks. For embeddings based on 2D convolutions (ResNet34), we used 40 log-Mel filter banks. Features were short-time centered before silence removal with a 3 seconds sliding window.

JHU-HLTCOE processed the audio at 16 KHz sampling rate. We used 80 Mel filter bank log-energies with a 25 ms window every 10 ms over the spectral band of 20–7600 Hz. Mean normalization was applied using a moving window of 3 seconds. A DNN VAD was used to detect speech segments.

5. Voice activity detection

5.1. Kaldi energy VAD

The Kaldi energy VAD makes frame-level decisions, classifying a frame as speech or non-speech based on the average log-energy in a given window. JHU-CLSP used this VAD for all datasets except SRE18 VAST and SRE19 AV. MIT-LL and JHU-HLTCOE used this VAD for all datasets.

5.2. TDNN VAD

The voice activity detection (VAD) is based on a Time-Delay Neural Network (TDNN) which special characteristic is the statistics pooling for long context information [4]. The process starts by training a GMM-HMM with 3 target classes: speech, non-speech and noise (garbage). All the acoustic phones are mapped to these classes. The TDNN is trained using the alignments from the GMM-HMM. At test time, we perform a usual

[1]http://www.openslr.org/resources/28

[2]http://www.openslr.org/resources/17

[3]`http://www.3gpp.org/ftp/Specs/archive/26_series/26.073/26073-800.zip`

[4]More details can be found in [1] egs/aspire/s5/local/segmentation/tuning/train_stats_asr_sad_1a.sh

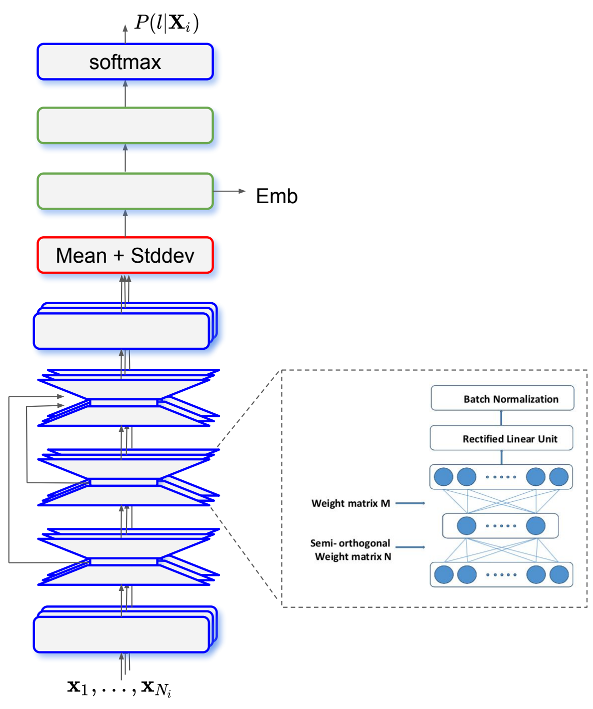

Figure 1: Factorized TDNN x-vector architecture.

Table 1: *Factorized TDNN 3A x-vector architecture*

Layer	Layer Type	Context factor1	Context factor2	Skip conn. from layer	Size	Inner size
1	TDNN-ReLU	t-2:t+2			512	
2	F-TDNN-ReLU	t-2, t	t, t+2		1024	256
3	F-TDNN-ReLU	t	t		1024	256
4	F-TDNN-ReLU	t-3, t	t, t+3		1024	256
5	F-TDNN-ReLU	t	t	3	1024	256
6	F-TDNN-ReLU	t-3, t	t, t+3		1024	256
7	F-TDNN-ReLU	t-3, t	t, t+3	2, 4	1024	256
8	F-TDNN-ReLU	t-3, t	t, t+3		1024	256
9	F-TDNN-ReLU	t	t	4, 6, 8	1024	256
10	Dense-ReLU	t	t		2048	
11	Pooling (mean+stddev)	full-seq			2x2048	
12	Dense-ReLU				512	
13	Dense-ReLU				512	
14	Dense-Softmax				N. spks.	

decoding transforming the estimating probabilities into likelihoods and assign every frame a speech or non-speech label. This VAD was trained using AVA-Speech corpus [2].

6. Audio embeddings

6.1. JHU-CLSP F-TDNN x-vectors

We used Kaldi x-vectors [3, 4] based on factor TDNN (F-TDNN) encoder with skip connections as described in [5, 6]. These used mean+stddev pooling and were trained by softmax cross-entropy. We have 3 variants of the F-TDNN encoder:

- 3A: This is the architecture that we used in the previous evaluation. This model has around 17M parameters. We have two models with this architecture:
  - 3A-SRE18: This is last year's model. It was trained on Train-CLSP-SRE18 dataset for 3 epochs.
  - 3A-JSALT: This was trained on the Train-CLSP-JSALT dataset for 6 epochs.

Table 2: *Factorized TDNN 4A x-vector architecture*

Layer	Layer Type	Context factor1	Context factor2	Skip conn. from layer	Size	Inner size
1	TDNN-ReLU	t-2:t+2			512	
2	F-TDNN-ReLU	t-2, t	t, t+2		725	180
3	F-TDNN-ReLU	t	t		725	180
4	F-TDNN-ReLU	t-3, t	t, t+3		725	180
5	F-TDNN-ReLU	t	t	3	725	180
6	F-TDNN-ReLU	t-3, t	t, t+3		725	180
7	F-TDNN-ReLU	t-3, t	t, t+3	2, 4, 6	725	180
8	F-TDNN-ReLU	t-3, t	t, t+3		725	180
9	F-TDNN-ReLU	t	t	3, 5, 7	725	180
10	F-TDNN-ReLU	t-3, t	t, t+3		725	180
11	F-TDNN-ReLU	t	t	6, 8, 10	725	180
12	F-TDNN-ReLU	t-3, t	t, t+3		725	180
13	F-TDNN-ReLU	t	t	7, 9, 11	725	180
14	TDNN-ReLU	t	t		1800	
15	Pooling (mean+stddev)	full-seq			2x1800	
16	Dense(x-vector)-ReLU				512	
17	Dense-ReLU				512	
18	Dense-Softmax				N. spks.	

Table 3: *Factorized TDNN 5A x-vector architecture*

Layer	Layer Type	Context factor1	Context factor2	Skip conn. from layer	Size	Inner size
1	TDNN-ReLU	t-2:t+2			512	
2	F-TDNN-ReLU	t-2, t	t, t+2		2048	512
3	F-TDNN-ReLU	t-3, t	t, t+3		2048	512
4	F-TDNN-ReLU	t	t	2	2048	512
5	F-TDNN-ReLU	t-3, t	t, t+3		2048	512
6	F-TDNN-ReLU	t	t	4	2048	512
7	F-TDNN-ReLU	t-3, t	t, t+3		2048	512
8	F-TDNN-ReLU	t	t	6	2048	512
9	F-TDNN-ReLU	t-3, t	t, t+3		2048	512
10	F-TDNN-ReLU	t	t	8	2048	512
11	Pooling (mean+stddev)	full-seq			2x2048	
12	Dense-ReLU				512	
13	Dense-ReLU				512	
14	Dense-Softmax				N. spks.	

Table 4: *Extended TDNN x-vector architecture*

Layer	Layer Type	Context	Size
1	TDNN-ReLU	t-2:t+2	512
2	Dense-ReLU	t	512
3	TDNN-ReLU	t-2, t, t+2	512
4	Dense-ReLU	t	512
5	TDNN-ReLU	t-3, t, t+3	512
6	Dense-ReLU	t	512
7	TDNN-ReLU	t-4, t, t+4	512
8	Dense-ReLU	t	512
9	Dense-ReLU	t	512
10	Dense-ReLU	t	1500
11	Pooling (mean+stddev)	Full-seq	2x1500
12	Dense(Embedding)-ReLU		512
13	Dense-ReLU		512
14	Dense-Softmax		Num. spks.

- 4A: This is shallower and deeper F-TDNN compared to 3A model. This model has around 14M parameters. We have one model (4A-SRE19) with this architecture trained on Train-CLSP-SRE19 data for 6 epochs.
- 5A: This is a wider model compared to 3A. It has around 40M parameters. We have one model (5A-SRE19) with this architecture on Train-CLSP-SRE19 data for 3 epochs (because of deadline constrains).

6.2. JHU-HLTCOE x-vectors

We used a modified version of ResNet-34 presented in [7] for our speaker embedding system. A statistics pooling layer is used to aggregate across time. The number of channels in the first block was 64. We double the channels as we progress between blocks. The channels in the last layers were 512. The embedding dimension was 256.

6.3. JHU-CLSP ResNet-LDE embeddings

The ResNet-LDE system uses a residual network with 2D convolutions (ResNet34) [8] and the pooling layer is Learnable dictionary encoding (LDE) layer [9, 10].

Instead of using categorical cross-entropy for training, we used additive angular softmax loss [11, 12]. Additive angular softmax loss has stronger requirements for correct classification which generates a margin between embeddings of different classes.

In summary, we had three models of this kind:

- ResNet34-SRE18: This was trained on the CLSP-Train-SRE18 dataset.
- ResNet34-SRE19: This is ResNet34-SRE fine-tuned on the CLSP-Train-SRE19 dataset.
- ResNet34-Large-SRE19: This one uses larger Resnet size: blocks contain 64, 128, 256, 512 channels respectively. Due to memory issue we only use 8 clusters for this network.

6.4. MITLL x-vectors

The MIT LL x-vector DNN is an extended version of that described in TDNN in [4]. Specifically, the network is extended to include an additional TDNN layer with a dilation factor of 4. Additionally, the TDNN layers are interleaved with dense layers. The network, illustrated in Fig. 4, was trained by minimizing the multi-class cross-entropy of the output from the final softmax layer.

7. Audio Back-ends

7.1. JHU-CLSP back-end

The JHU-CLSP back-end consisted of LDA, centering, whitening, length normalization and generative Gaussian SPLDA.

We trained LDA, centering, whitening and SPLDA on the CLSP-PLDA data. LDA dimension was 200 and SPLDA had 150 eigenvoices. The centering for the VAST was MAP adapted from SITW-dev-diarized to SRE18-dev-VAST-diarized with relevance factor $r = 14$.

As there may be several speakers in the test segment, we used diarization to obtain several speaker clusters. Also, we combined the x-vectors obtained from diarization with the x-vectors coming from full-length recordings. We scored the enrollment segment against all the test segment clusters and selected the maximum score.

We used adaptive S-Norm with SITW-SRE18-DihardII-diarized as cohort. We selected the top 1000 cohort segments. We observed consistent improvement by adding the DihardII data to the AS-Norm. We didn't observe any improvement adding DihardII to PLDA or x-vector training.

7.2. JHU-HLTCOE back-end

The embeddings from our DNN were centered using the SITW-core enrollment data and compared using cosine distance. As there may be several speakers in the test segment, we used diarization to obtain several speaker clusters. We scored the enrollment segment against all the test segment clusters and selected the maximum score.

We used adaptive S-Norm with SITW-SRE18-DihardII-diarized as cohort. We selected the top 10% cohort segments. We observed consistent improvement by adding the Dihard data to the AS-Norm.

7.3. MIT Lincoln Laboratory back-end

The MIT LL back-end included LDA dimension reduction, centering/whitening and length normalization, PLDA scoring, and adaptive score normalization. The back-end relied on the training set described in Sec. 2.4. In the back-end pipeline, LDA was first applied to all x-vectors, reducing their dimension to 150, followed by whitening, centering, and length normalization. PLDA scoring was then performed. Adaptive S-Norm was applied for score normalization, where the SITW set was used as a cohort, and the 200 top-scoring utterances were used during normalization. The PLDA model was trained in a discriminative manner by selecting random x-vector pairs from the back-end training set, and minimizing the cross-entropy of the PLDA posteriors. A total of 1M trials was used during discriminative training.

8. Fusion and Calibration

Fusion and Calibration were performed using linear logistic regression with liblinear optimizer [13]. To select the best fusion combination, we used a greedy fusion scheme as last year [5]. First, we calibrate all the systems and select the best one given the lowest actual cost. We fix that one as the best system and evaluate all the two system fusions that include the best system. Thus, we select the best fusion of two systems. We fix those two system and then add a third system, and so on. To reduce the chances of over-fitting, in each step, we prioritize fusions with only positive weights. Logistic regression was trained on operating point $P_{\mathcal{T}} = 0.05$.

9. Diarization

9.1. JHU-CLSP diarization

For diarization of the SITW multi and VAST test data, we used a similar setup to the Kaldi x-vector callhome diarization recipe [5], which is based on [14].

We used of F-TDNN x-vector 5A, to compute embeddings using a sliding window with 1.5 seconds of frame-length and 0.75 seconds of frame-shift. We obtained sliding window embeddings for the dev, eval and VoxCelebCat without augmentation. We used VoxCeleb x-vectors to train LDA dimensionality reduction to 120, centering and PLDA. We scored all x-vectors

[5] `https://github.com/kaldi-asr/kaldi/tree/master/egs/callhome_diarization/v2`

Table 5: *Audio systems results on SRE18 VAST eval and SRE19 dev*

System	SRE18 VAST eval			SRE19 AV dev		
	EER	Min Cp	Act Cp	EER	Min Cp	Act Cp
Single systems						
COE-ResNet34	**8.04**	**0.258**	**0.262**	4.69	**0.172**	**0.185**
F-TDNN 5A-SRE19	10.18	0.33	0.339	5.84	0.205	0.239
F-TDNN 4A-SRE19	10.35	0.349	0.351	5.87	0.214	0.239
F-TDNN 3A-JSALT	10.35	0.347	0.352	6.28	0.198	0.223
F-TDNN 3A-SRE18	11.2	0.363	0.366	**4.63**	0.21	0.225
ResNet34-SRE18	9.36	0.356	0.359	6.66	0.207	0.246
ResNet34-SRE19	10.21	0.358	0.367	6.64	0.25	0.277
ResNet34-Large-SRE19	10.59	0.322	0.329	5.09	0.198	0.201
E-TDNN-MITLL	13.04	0.460	0.469	9.21	0.328	0.344
Submissions						
Primary	**7.64**	0.247	0.250	**4.50**	0.177	**0.183**
Single (5A-SRE19)	10.18	0.33	0.339	5.84	0.205	0.239
Contrastive	7.65	**0.243**	**0.243**	4.51	**0.171**	0.186

in a given recording against each other and applied AHC on the score matrix. We tuned the stopping threshold on SRE18 eval.

We assumed that the target speaker would have a significant amount of speech in the test segment. For that reason, we discarded all the speaker clusters with less than 10 seconds duration unless all clusters in the segment are shorter than that.

9.2. JHU-HLTCOE diarization

Speaker diarization was performed by clustering segment embeddings with the leave-one-out PLDA GMM approach from [15]. Given inputs of the length-normalized segment embeddings, initial segment posteriors over speakers from k-means, and PLDA parameters (within-class and across-class covariance), this algorithm alternates between updating the speaker models and generating segment speaker posteriors.

The diarization embeddings were trained to optimize this clustering using Bayesian enrollment from 1 to 20 previous segments. Training data was LDC corpora Switchboard, Fisher, Mixer6, SRE2004-10 plus narrow-band versions of VoxCeleb1 and VoxCeleb2. Various TDNN architectures were explored, but the submitted version uses a modified version of ResNet34. Diarization was performed on test cuts only, and initialized with five clusters per cut; the PLDA GMM naturally reduced the number of speakers within 30 iterations. The speaker recognition score was generated as the maximum over embeddings generated from no diarization and each of the diarization output speakers.

10. Audio Single Systems

Table 5 presents the results of the individual systems on SRE18 VAST eval and SRE19 dev audio conditions.

11. Audio Submissions

The submissions for the audio conditions where:

- Primary: Best fusion of 4 systems composed of: COE, F-TDNN-5A-SRE19, F-TDNN-4A-SRE19, ResNet34-SRE19.
- Single: Best JHU-CLSP single system: F-TDNN-5A-SRE19.
- Contrastive: Best fusion of 4 systems composed of: COE, F-TDNN-5A-SRE19, F-TDNN-4A-SRE19, ResNet34-SRE19, MITLL.

Table 5 presents the results of our submissions for audio.

12. Video Embeddings

12.1. JHU-CLSP embeddings

To extract embeddings for face recognition, we used original InsightFace [11] embeddings and RetinaFace [12] face detection implementations[6]. Pre-trained models for RetinaFace were computed, [7] and 4 pretrained models for InsightFace were obtained [8]:

- r100: LResNet100.
- r50: LResNet50.
- r34: LResNet34
- mobile: MobileFaceNet

InsightFace models were trained on MS1M-Arcface dataset. RetinaFace was trained on WiderFace dataset[9].

Face detection procedure:

- Enrollment: We applied RetinaFace detector at the frames given in reference bounding boxes ± 2 frames. Then we keep the faces that overlap with the reference bounding boxes. If we don't detect any faces in the reference bounding boxes we apply face detection over the full video and keep all the faces.
- Test: We detect faces with RetinaFace over the full video extracting frames at a rate of 1 frame per second.

After face detection, we align the faces with the landmarks obtained with Multi-task Cascaded Convolutional Network (MT-CNN) [16] (included with the InsightFace embedding models) and extract embeddings with the InsightFace

[6]https://github.com/deepinsight/insightface
[7]https://github.com/deepinsight/insightface/tree/master/RetinaFace
[8]https://github.com/deepinsight/insightface/wiki/Model-Zoo
[9]http://shuoyang1213.me/WIDERFACE/WiderFace_Results.html

model. If the MT-CNN fails to detect the landmarks, we use the landmarks provided by the RetinaFace detector.

12.2. JHU-HLTCOE embeddings

For face embeddings, we utilized an implementation of InsightFace available for download[10] that follows the methods outlined in [11]. Our face embedding model was similarly found in the same repository[11]. The downloaded ResNet-101 model was trained with the MS-Celeb-1M dataset (3.8M faces, 85K faces).

Face detection was performed with a Multi-task Cascaded Convolutional Network [16], also included in the downloaded implementation. The default settings were used for an initial detection pass run on frames extracted every quarter of a second from all videos. Faces with a final confidence score less than 0.95 were subsequently excluded in order to minimize false alarms or low-resolution faces that might corrupt downstream processing.

Enrollment models were created by averaging embeddings from any detected boxes that overlap with the provided enrollment box. Frames were searched for overlapping boxes within a second before or after the enrollment box's frame (a total of 9 frames searched). If no overlapping boxes were found with any of the given enrollment faces, that model was left empty and scores for its trials were added simply as zero after calibration.

Test faces were clustered using Agglomerative Hierarchical Clustering to 21 clusters in all videos.

The choice of 21 clusters (as well as the confidence threshold and search range for enrollment faces) was determined with experiments on the provided Janus dataset.

13. Video Back-ends

13.1. JHU-CLSP back-end

We evaluated different back-end strategies all based on cosine scoring. These back-end for each trial of enrollment video i versus test video j do:

- be1: Score all detected enrollment versus all detected test face embeddings and take the maximum score.
- be2: Average all the enrollment embeddings and score versus all detected test face embeddings and take the maximum score.
- be3: Compute the median of all the enrollment embeddings and score versus all detected test face embeddings and take the maximum score.
- be4: Compute the median of all the enrollment embeddings, perform agglomerative clustering (AHC) on the test embeddings with stopping threshold=0.8, score the enrollment embedding versus all the test clusters and take the maximum score.
- be5: Perform AHC in the enrollment and test sides, score all enrollment clusters versus all test clusters and take the maximum.
- be6: Compute the median of all the enrollment embeddings, apply self-attention procedure on the test side to enhance the test embeddings. This self-attention method consisted in, for each test embedding $\mathbf{x}_{jt}$ in the test video, we computed a new enhanced embedding $\mathbf{y}_{jt}$ as

$$\mathbf{p}_{jt} = \mathrm{softmax}_k(a\cos(\mathbf{x}_t, \mathbf{x}_k)) \quad t = 1, \dots, T \quad (1)$$

$$\mathbf{y}_{jt} = \sum_{k=1}^{T} p_{jtk}\mathbf{x}_k \qquad t = 1, \dots, T \quad (2)$$

where we set $a = 2$ empirically. The idea is to improve each embedding by doing a weighted average of the embeddings that are closer to it. Finally, we score the enrollment embedding versus all the enhanced test embeddings and take the maximum.

- be7: Compute the median of all the enrollment embeddings, apply self-attention procedure on the test side to enhance the test embeddings, apply attention procedure between median enrollment embedding $\mathbf{e}_j$ of video j and the test enhanced embeddings $\mathbf{y}_i$ to produce a single test embedding $\mathbf{z}_j$ closer to the enrollment, that is

$$\mathbf{q}_{ij} = \mathrm{softmax}_k(b\cos(\mathbf{e}_i, \mathbf{y}_{jk})) \quad (3)$$

$$\mathbf{z}_{ij} = \sum_{k=1}^{T} q_{ijk}\mathbf{y}_{jk} \quad (4)$$

where we set $b = 7$. Then $\mathbf{e}_i$ is scored against a single test embedding $\mathbf{z}_{ij}$. In this case, the final test embedding used in each trial depends on the enrollment embedding.

- be9: We apply self-attention procedure to the enrollment and test side to obtain enhanced enrollment and test embeddings. Then, we score all enrollment vs all test enhanced embeddings and take the maximum.

[10]https://github.com/foamliu/InsightFace-v3

[11]https://github.com/foamliu/InsightFace-v3/releases/download/v1.0/BEST_checkpoint.tar

Table 6: *Video systems results on SRE19 AV dev.*

Systems	SRE19 AV dev		
	EER	Min Cp	Act Cp
Single systems			
COE	**6.02**	**0.185**	**0.2**
r100-be2-snorm	11.29	0.228	0.249
r100-be4	8.03	0.360	0.384
r100-be5	10.96	0.364	0.375
r100-be7-snorm	10.79	0.238	0.245
r100-be9-snorm	9.73	0.219	0.236
r50-be3	12.64	0.454	0.475
r50-be3-snorm	14.55	0.316	0.334
r50-be5-snorm	13.10	0.284	0.308
r50-be7-snorm	14.54	0.347	0.347
r34-be1	14.09	0.441	0.456
r34-be2	12.67	0.501	0.522
r34-be4-snorm	14.69	0.328	0.334
r34-be5	13.78	0.435	0.462
r34-be5-snorm	15.49	0.255	0.281
r34-be7-snorm	13.28	0.377	0.377
mobile-be1	15.52	0.760	0.775
mobile-be1-snorm	16.20	0.356	0.360
mobile-be3-snorm	17.10	0.330	0.363
mobile-be4-snorm	17.34	0.337	0.345
mobile-be6-snorm	14.35	0.332	0.346
Submissions			
Primary	5.12	**0.140**	**0.140**
Single	9.73	0.219	0.236
Contrastive	**5.00**	**0.140**	0.149

Table 7: *Multi-modal systems results on SRE19 AV dev.*

Systems	SRE19 AV dev		
	EER	Min Cp	Act Cp
Submissions			
Primary	1.89	**0.037**	0.054
Single	2.63	0.056	0.070
Contrastive	**1.75**	**0.037**	**0.042**

For each back-end flavour, we have versions with and without adaptive S-Norm. We used the top 1000 element from the cohort. The cohort for score normalization was obtained from JANUS dev test set.

13.2. JHU-HLTCOE back-end

For a given model/test pair, the enrollment model was scored against all 21 face clusters using cosine scoring of the embeddings, and the maximum score was kept for the trial.

Face scores were calibrated using logistic regression learned on the SRE19 Development set, as calibration parameters learned on the Janus data were not found to perform well on the development data.

14. Video single systems

Table 6 presents the results of the individual systems on SRE19 AV dev video condition. We include only embedding plus back-end combinations that were included in one of the submissions.

15. Video submissions

The submissions for the video conditions where:

- Primary: Fusion at $P_{\mathcal{T}} = 0.05$ of
  - COE system.
  - Best fusion of 3 JHU-CLSP systems at $P_{\mathcal{T}} = 0.2$, composed of: r100-be9-snorm, r100-be4, mobile-be1.
- Single: Best JHU-CLSP single system: r100-be9-snorm.
- Contrastive:

  Fusion at $P_{\mathcal{T}} = 0.05$ ofCOE system. Best fusion of 20 JHU-CLSP systems at $P_{\mathcal{T}} = 0.2$, composed of: r100-be9-snorm+r100-be4+mobile-be1+r50-be7-snorm+mobile-be4-snorm+mobile-be1-snorm+r34-be1+r100-be7-snorm+r50-be3+r34-be7-snorm+r100-be2-snorm+r50-be3-snorm+r100-be5+mobile-be3-snorm+mobile-be6-2-snorm+r50-be5-snorm+r34-be5+r34-be4-snorm+r34-be5-snorm.

Table 6 presents the results of our submissions for audio.

16. Audio Visual submissions

Multi-modal submissions (Primary, single and contrastive) were obtained by adding the score of the corresponding audio and video submissions. We added the scores instead of averaging because both modalities are independent.

Table 8: *Computational resources for face embeddings.*

System	Seconds per item	Memory (GB)
Detection (MT-CNN)	0.27	0.0024
Embedding (InsightFace)	1.85	1

Table 9: *Computational resources x-vectors*

System	Real time factor	Memory (GB)
F-TDNN 3A x-vector	3.5	3
F-TDNN 4A x-vector	4.3	3
F-TDNN 5A x-vector	3	3
E-TDNN x-vector	7	2
ResNet34 embedding	26.3	0.2
ResNet34LDE-Large	11.7	1.0

17. Computation resources

Processing times were measured in Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz. Most of the processing time is dedicated to the embedding extraction. MFCC, VAD and back-end processing time are negligible comparison.

18. References

[1] 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 *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU2011*, Waikoloa, HI, USA, dec 2011, pp. 1–4, IEEE.

[2] Sourish Chaudhuri, Joseph Roth, Daniel PW Ellis, Andrew Gallagher, Liat Kaver, Radhika Marvin, Caroline Pantofaru, Nathan Reale, Loretta Guarino Reid, Kevin Wilson, et al., “Ava-speech: A densely labeled dataset of speech activity in movies,” *arXiv preprint arXiv:1808.00606*, 2018.

[3] David Snyder, Daniel Garcia-Romero, Daniel Povey, and Sanjeev Khudanpur, “Deep Neural Network Embeddings for Text-Independent Speaker Verification,” in *Proceedings of the 18th Annual Conference of the International Speech Communication Association, INTERSPEECH 2017*, Stockholm, Sweden, aug 2017, pp. 999–1003, ISCA.

[4] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur, “X-Vectors : Robust DNN Embeddings for Speaker Recognition,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018*, Alberta, Canada, apr 2018, pp. 5329–5333, IEEE.

[5] Jesús Villalba, Nanxin Chen, David Snyder, Daniel Garcia-Romero, Alan McCree, Gregory Sell, Jonas Borgstrom, Fred Richardson, Suwon Shon, Francois Grondin, Reda Dehak, Leibny Paola Garcia-Perera, Daniel Povey, Pedro Torres-Carrasquillo, Sanjeev Khudanpur, and Najim Dehak, “State-of-the-art Speaker Recognition for Telephone and Video Speech: the JHU-MIT Submission for NIST SRE18,” in *Proceedings of the 20th Annual Conference of the International Speech*

Communication Association, INTERSPEECH 2019, Graz, Austria, sep 2019.

[6] Jesús Villalba, Nanxin Chen, David Snyder, Daniel Garcia-Romero, Alan McCree, Gregory Sell, Jonas Borgstrom, Leibny Paola García-Perera, Fred Richardson, Réda Dehak, Pedro A. Torres-Carrasquillo, and Najim Dehak, "State-of-the-art Speaker Recognition with Neural Network Embeddings in NIST SRE18 and Speakers In The Wild Evaluations," *Computer Speech & Language*, p. 101026, oct 2019.

[7] Hossein Zeinali, Shuai Wang, Anna Silnova, Pavel Matejka, and Oldrich Plchot, "But system description to voxceleb speaker recognition challenge 2019," in *VoxSRC Challenge workshop*, 2019.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep Residual Learning for Image Recognition," dec 2015.

[9] Weicheng Cai, Zexin Cai, Xiang Zhang, Xiaoqi Wang, and Ming Li, "A Novel Learnable Dictionary Encoding Layer for End-to-End Language Identification," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, Canada, apr 2018, pp. 5189–5193, IEEE.

[10] Weicheng Cai, Jinkun Chen, and Ming Li, "Exploring the Encoding Layer and Loss Function in End-to-End Speaker and Language Recognition System," in *Odyssey 2018 The Speaker and Language Recognition Workshop*, Les Sables d'Olonne, France, jun 2018, pp. 74–81, ISCA.

[11] Jiankang Deng, Jia Guo, Xue Niannan, and Stefanos Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," *arXiv:1801.07698*, 2018.

[12] Jiankang Deng, Jia Guo, Yuxiang Zhou, Jinke Yu, Irene Kotsia, and Stefanos Zafeiriou, "Retinaface: Single-stage dense face localisation in the wild," *arXiv preprint arXiv:1905.00641*, 2019.

[13] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin, "Liblinear: A library for large linear classification," *Journal of machine learning research*, vol. 9, no. Aug, pp. 1871–1874, 2008.

[14] Daniel Garcia-Romero, David Snyder, Gregory Sell, Daniel Povey, and Alan Mccree, "Speaker Diarization Using Deep Neural Network Embeddings," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017*, New Orleans, LA, USA, mar 2017, pp. 4930–4934, IEEE.

[15] Alan McCree, Gregory Sell, and Daniel Garcia-Romero, "Speaker Diarization using Leave-one-out Gaussian PLDA Clustering of DNN Embeddings," in *Interspeech*, 2019.

[16] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao, "Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks," *IEEE Signal Processing Letters*, 2015.