The HYU Speaker Recognition System for the SdSV Challenge 2020

Joon-Young Yang, Jeong-Hwan Choi and Joon-Hyuk Chang

Department of Electronics and Computer Engineering
Hanyang University, Seoul, Republic of Korea
dreadbird06@gmail.com, brent1104@hanyang.ac.kr, jchang@hanyang.ac.kr

Abstract

In this document, we describe the speaker recognition system of Hanyang University (HYU) team submitted to the text-independent speaker verification (SV) task of the Short-duration Speaker Verification (SdSV) Challenge 2020. The description includes the composition of the training dataset, topologies and training strategy of the neural speaker embedding networks, and the evaluation results. In a nutshell, we train three neural networks that share the same backbone architecture of a 34-layered residual convolutional network yet mainly differ in pooling methods. The networks are first pre-trained using a large-scale out-of-domain dataset, and then fine-tuned using the in-domain DeepMine dataset. Finally, a simple cosine similarity scoring method is employed to evaluate the speaker verification trials. In addition to the evaluation results of the SdSV Challenge, we also report the performance of the pre-trained speaker embedding models on the VoxCeleb1 SV benchmark.
Index Terms: speaker verification, short-duration speaker verification challenge

1. Introduction

This document describes the speaker recognition system of Hanyang University (HYU) team submitted to the text-independent speaker verification (SV) task of the Short-duration Speaker Verification (SdSV) Challenge 2020. The challenge imposed a fixed training condition on the SV systems; the VoxCeleb1 [1], VoxCeleb2 [2], and LibriSpeech [3] corpus can be used as out-of-domain datasets, and the DeepMine [] dataset is provided as the in-domain dataset. As for the evaluation setup, the durations of the enrollment and the test utterances comprising the trial set are considerably different; the former is uniformly distributed between 3 to 120 s after a silence removal, whereas the latter between 1 to 8 s. Moreover, one of the two partitions of the trial set consisted of Persian-English cross-language trials.

We build three neural speaker embedding networks that share the same backbone architecture of a 34-layered residual convolutional network (ResNet-34) [4]. The three ResNet-34 networks exploit different pooling strategies, while some of those also adopt the recently proposed input feature augmentation method [5]. Basically, we use a two-stage approach to train the neural speaker embedding networks: they are first pre-trained using the large-scale out-of-domain datasets, and then fined-tuned using the in-domain dataset.

2. Dataset

The out-of-domain speech dataset was comprised of the VoxCeleb1, VoxCeleb2, and LibriSpeech corpora. We applied the standard data augmentation method to the entire set of utterances based on the Kaldi [6] recipe, which created four more copies of the original speech dataset with the following types of corruptions:

- reverberation using simulated room impulse responses
- a variety of foreground noises
- simulated babbling background noise
- background music noise.

Next, from the augmented set of utterances, we filtered out those with durations less than 4 s, and also excluded the speakers with the number of utterances less than 8. Finally, after cutting out a small portion for validation purposes, we obtained a training set comprising 6,728,250 utterances from 9,658 speakers and a validation set of 5,000 utterances from 200 speakers. Note that the 40 speakers in the VoxCeleb1 dataset, whose name starts with 'E', were not included in both the training and validation datasets, and later used to examine the performance of the SV systems on the VoxCeleb1 SV benchmark.

To prepare the in-domain training dataset, the abovementioned filtering and augmentation processes were also applied to the DeepMine Task2 Train partition. As a result, we obtained a training set comprising 404,729 utterances from 588 speakers and a validation set of 5,000 utterances from 200 speakers

Finally, we extracted 56-dimensional log mel-filterbank energies (MFBEs) from the entire dataset, and the MFBEs were used as the input acoustic features for training the neural speaker embedding networks. The window and hop size for the acoustic feature extraction were set to 25 ms and 10 ms, respectively. The energy-based voice activity detection algorithm provided in Kaldi [6] was used to remove the silence frames from the sequence of MFBEs.

3. Neural speaker embedding networks

3.1. Backbone architecture

Since the neural networks designed for the challenge have an identical structure before the pooling layer, we only describe the shared architecture here and leave the specifications of the rest for the following subsections. Table. 1 presents the layer configurations of the ResNet-34 architecture. Each residual convolution block (ResBlock) consists of a series of batch normalization (BN) [7], rectified linear unit (ReLU) nonlinearity, and 2D convolution (Conv2D) operations. To be more specific, given the input feature maps, a ResBlock performs BN–ReLU–Conv2D–BN–ReLU–Conv2D operations followed by an additive shortcut connection of the input. In the forward path, only the first Conv2D and ResBlock keep the size of the feature maps, while the rest of the ResBlocks downsize the feature maps in the first Conv2D layer within each block. All the Conv2D operations are performed with a kernel of size 3×3, except for the 1×1 Conv2D layers that downsample the feature maps prior to the additive shortcut connections. On top of

each output of the Conv2D-0 and ResBlocks, an average pooling operation is performed over the frequency axis (AverageFreq). The frequency-pooled feature maps are passed to the pooling layers, which will be described in the next subsection.

It can be noted that the sliding-window mean subtraction (SWMS) operation is implemented as a part of the network to normalize the input features. Identical to the Kaldi's implementation, the SWMS operation subtracts the mean of the windowed input features in a sliding-window manner if the number of time frames is greater than 300; otherwise, it simply applies a global mean subtraction.

Table 1: *Backbone architecture of the ResNet-34-based neural speaker embedding model. (T is the number of time frames in an input feature sequence.)*

Name	Conv. Ops.	Output
Input	-	$1 \times 56 \times T$
SWMS	-	$1 \times 56 \times T$
Conv2D-0	3×3, 32	$32 \times 56 \times T$
AverageFreq-0	-	$32 \times T$
ResBlock-1	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 3$	$32 \times 56 \times T$
AverageFreq-1	-	$32 \times T$
ResBlock-2	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 4$	$64 \times 28 \times T/2$
AverageFreq-2	-	$64 \times T/2$
ResBlock-3	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 6$	$128 \times 14 \times T/4$
AverageFreq-3	-	$128 \times T/4$
ResBlock-4	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 3$	$256 \times 7 \times T/8$
AverageFreq-4	-	$256 \times T/8$

3.2. Model specifications

Three different neural speaker embedding models are contructed, each of which employs a unique pooling strategy. Based on the observations from the recent studies that successfully boosted the performance of the TI SV systems [8, 9, 10], we also applied pooling operations to the output feature maps of the intermediate-level ResBlocks. The types of the pooling operations we considered are the learnable dictionary encoding (LDE) [11, 12], temporal attentive statistics (TAS) pooling [13], and multi-head LDE (MHLDE) which was devised as an analogue of the multi-head TAS pooling [13] for the LDE. Below are the specifications of the three neural speaker embedding networks.

3.2.1. ResNet-34 with hierarchical LDE pooling

The first neural speaker embedding model employs the LDE pooling to each of the frequency-averaged intermediate-level output features. Table. 2 summarizes the pooling strategy for the first model. As shown in the table, the LDE learns to extract a pre-defined number of code vectors given a variable-length sequence of input features. Note that the dimension of the LDE output is determined as the multiplication of that of the input and the number of codes in a dictionary. Finally, the outputs of the LDE pooling layers are concatenated to a 22,016-dimensional code vector and passed to a dense layer with 144 hidden units. The dense layer compises an affine transform followed by a BN, and the speaker embedding is extracted from this layer, immediately after the BN. As multiple LDE poolings are applied to the intermediate-level features of different hierarchies, we simply denote the employed pooling strategy as hierarchical LDE (HLDE) and the corresponding model as *ResNet34-HLDE*.

Table 2: *Specifications of the hierarchical LDE pooling operations.*

Name	#codes	Input	Output
LDEpool-0	8	$32 \times T$	256
LDEpool-1	8	$32 \times T$	256
LDEpool-2	16	$64 \times T/2$	1,024
LDEpool-3	32	$128 \times T/4$	4,096
LDEpool-4	64	$256 \times T/8$	16,384
Concat	-	22,016	
Dense-1	22,016	144	

Table 3: *Specifications of the hierarchical TAS pooling operations.*

Name	Input	Output
TASpool-0	$32 \times T$	256
TASpool-1	$32 \times T$	256
TASpool-2	$64 \times T/2$	128
TASpool-3	$128 \times T/4$	256
TASpool-4	$256 \times T/8$	512
Concat	-	1,024
Dense-1	1,024	256

Table 4: *Specifications of the BLSTM-based input feature augmentation method [5].*

Name	Input	Output
BLSTM-0	$1 \times 56 \times T$	$1 \times 256 \times T$
Dense-0	$1 \times 256 \times T$	$1 \times 840 \times T$
Reshape	$1 \times 840 \times T$	$15 \times 56 \times T$
Concat	$\begin{bmatrix} 1 \times 56 \times T \\ 15 \times 56 \times T \end{bmatrix}$	$16 \times 56 \times T$

3.2.2. BLResNet34 with hierarchical TAS pooling

In the second model configuration, the LDE poolings used in the first model are replaced by the TAS poolings. Table. 3 summarizes the pooling strategy for the second model, where the dimension of the output is twice of that of the input due to the concatenation of the mean and the standard deviation vectors. Herein, we applied length normalization to the mean and the standard deviation vectors, which contrained both vectors to have unit norms. The statistics computed from multiple intermediate layer outputs are concatenated to a 1,024-dimensional vector and passed to a dense layer with 256 hidden units. In addition to the replacement of the pooling strategy, we also adopted the recently proposed bi-directional long short-term memory (BLSTM)-based input feature augmentation method [5]. Table. 4 presents the adopted augmentation method. First, the sequence of 56-dimensional MFBEs are processed by a single BLSTM layer with 128 hidden units for each direction, and then the output sequence is linearly transformed to a sequence of 840-dimensional feature vectors. Subsequently, each of the

840-dimensional feature vectors in a sequence is reshaped to be treated as 56-dimensional features stacked along the channel axis, and then concatenated to the input MFBEs. Finally, these channel-augmented stacked representations are directly used as the inputs for the ResNet34 model. We denote this model as *BLResNet34-HTAS*.

3.2.3. BLResNet34 with hierarchical MHLDE pooling

For the last model configuration, we simply replaced the TAS poolings with the MHLDE poolings. We denote this model as *BLResNet34-HMHLDE*.

4. Training and evaluation

Since the provided in-domain DeepMine dataset is somewhat language-specific, we choose to first pre-train the speaker embedding networks using the large-scale out-of-domain datasets, and then fine-tune the networks using the in-domain dataset. All the neural networks were implemented using PyTorch [14], and trained on a single NVIDIA RTX 2080 Ti GPU.

4.1. Pre-training

In the pre-training stage, all the networks were trained to classify the 9,658 speaker identities using the additive margin softmax (AMSoftmax) [15] loss function defined as follows:

$$L_{AM} = \frac{1}{N}\sum_{i=1}^{N} -\log \frac{e^{cos(\theta_{y_i,i})-m}}{e^{cos(\theta_{y_i,i})-m} + \sum_{j\neq y_i}^{C} e^{cos(\theta_{j,i})}}, \quad (1)$$

where C is the number of classes, N is the number of samples in a mini-batch, i is the sample index, $y_i \in \{1, 2, \ldots, C\}$ is the class label of the i-th sample, and $\theta_{j,i}$ denotes the angle between the j-th column vector of the weight matrix and the output of the penultimate dense layer computed from the i-th sample. We set the value of the margin term, m, to 0.2.

We used the hyperparameter settings and training schemes similar to those described in [16]. A single mini-batch was composed by gathering 64 randomly cropped chunks of MFBEs with a duration of 200 frames, and a single training epoch was defined as the iterations over 20,000 mini-batches. The neural networks were trained using the stochastic gradient descent (SGD) optimizer with an initial learning rate of 0.1, which was annealed by half whenever the validation loss did not improve for three consecutive epochs. The training stopped either if the number of epoch reaches 100 or the validation loss did not improve for eight consecutive epochs. The weights of the networks were $l2$-regularized with the scaling factor of 0.01 [16], and the dropout [17] was applied for every third mini-batch with a rate of 0.2. We also used the same logit annealing strategy described in [16], which gradually applies the AMSoftmax loss from the scratch to the end of the training.

4.2. Fine-tuning

In the fine-tuning stage, we first re-initialized the output layer of the embedding network to random values, freezed the rest of the network layers, and then fine-tuned the output layer using the in-domain dataset. The training was carried out in a manner similar to that described in the pre-training stage, yet we used the standard softmax loss without applying the additive margin term. After that, we unfreezed the entire network and further proceeded the fine-tuning with the AMSoftmax loss. We fixed the learning rate to 10^{-6} and trained the entire network until the validation loss converged.

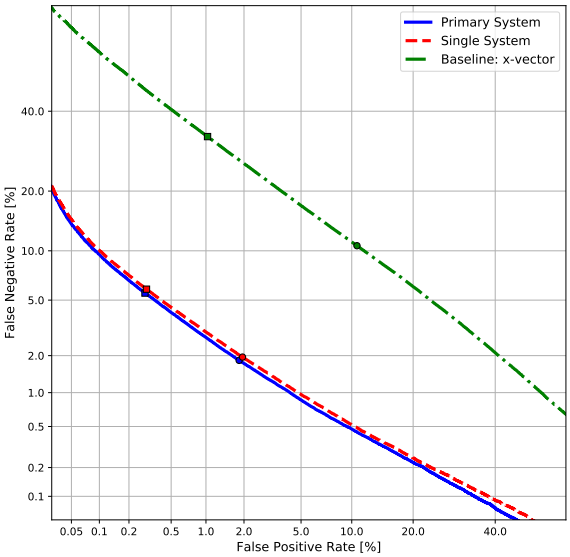

Figure 1: *DET curves of the submitted systems on the progress set.*

4.3. Evaluation

The speaker embeddings were centered using the in-domain global mean statistic and directly used to compute the cosine similarity score for each trial. Fusion of the systems was accomplished by a simple score-level averaging. The performance of the SV systems were evaluated in terms of the equal error rate (EER) and minimum detection cost function (minDCF).

5. Results

Table 5 summarizes the SV results of the trained systems for the SdSV Challenge. The first three rows present the performance of the single systems, and the rest shows the results of score fusion. Note that the submitted single and primary systems are presented with a check mark on their left side. In the first two columns, we also report the performance of our systems on the VoxCeleb1 benchmark. The trial set consisted of pairs of utterances that belong to the speakers whose name starts with 'E', and the number of evaluated trials was 35,061 due to some missing files.

Comparing the first three rows, the single systems showed comparable performance on the VoxCeleb1 benchmark, while the systems with BLSTM-based feature augmentation strategy were superior to the system without it on the SdSV Challenge dataset. Since the BLResNet34-HTAS system slightly outperformed the BLResNet34-HMHLDE on the progress set of the SdSV Challenge, we submitted the former as our single system. As for the fusion of the systems, the combination of the systems that adopted the BLSTM-based feature augmentation method was slightly superior to the others, when evaluated on the progress set. Therefore, we submitted it as our primary system. The detection error tradeoff (DET) curves, provided by the challenge organizers, are shown in Fig. 1, 2, and 3.

Table 5: *Speaker verification results for the VoxCeleb1 benchmark and the SdSV Challenge 2020.*

System	VoxCeleb1		SdSVC progress set		SdSVC evaluation set	
	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF
ResNet34-HLDE (1)	1.56	0.1613	2.06	0.0941	2.07	0.0944
✓ BLResNet34-HTAS (2)	1.60	0.1598	1.93	0.0877	1.95	0.0881
BLResNet34-HMHLDE (3)	1.52	0.1718	1.96	0.0889	1.96	0.0888
1+2	1.48	0.1555	1.90	0.0865	1.89	0.0864
1+3	1.39	0.1624	1.89	0.0865	1.88	0.0863
✓ 2+3	1.47	0.1512	1.84	0.0839	1.83	0.0836
1+2+3	1.41	0.1508	1.84	0.0842	1.83	0.0839

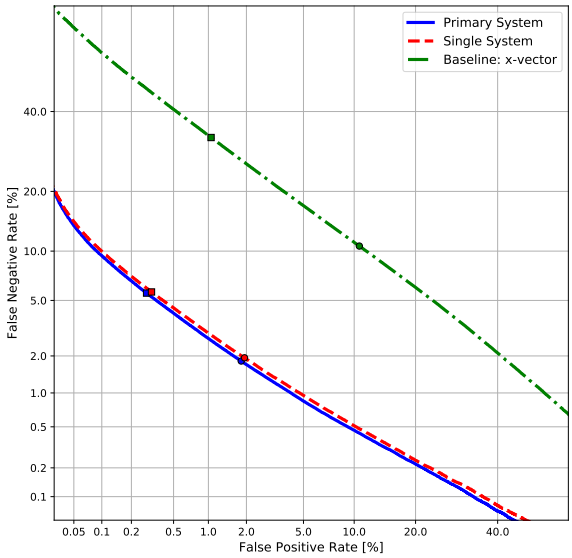

Figure 2: *DET curves of the submitted systems on the evaluation set.*

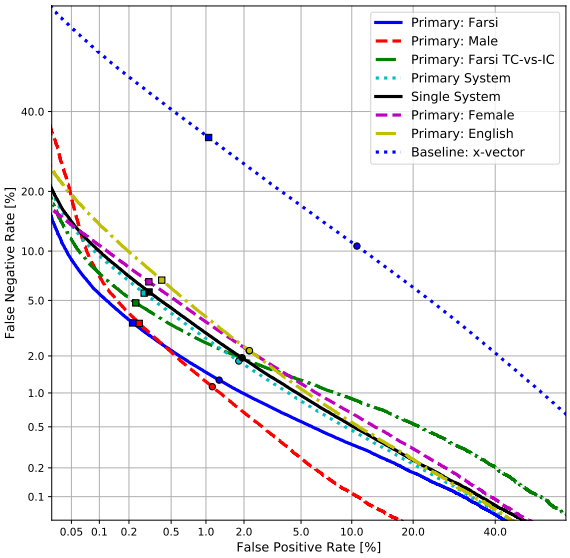

Figure 3: *DET curves of the submitted systems on the sub-conditioned evaluation sets.*

6. References

[1] H. Zeinali, H. Sameti, and T. Stafylakis, "DeepMine speech processing database: Text-dependent and independent speaker verification and speech recognition in Persian and English." in *Proc. Odyssey*, 2018, pp. 386–392.

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

[3] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2015, pp. 5206–5210.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[5] Y. Zhao, T. Zhou, Z. Chen, and J. Wu, "Improving deep cnn networks with long temporal context for text-independent speaker verification," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020, pp. 6834–6838.

[6] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The Kaldi speech recognition toolkit," in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, 2011.

[7] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[8] Z. Gao, Y. Song, I. McLoughlin, P. Li, Y. Jiang, and L. Dai, "Improving aggregation and loss function for better embedding learning in end-to-end speaker verification system," in *Proc. Interspeech*, 2019, pp. 361–365.

[9] S. Seo, D. J. Rim, M. Lim, D. Lee, H. Park, J. Oh, C. Kim, and J.-H. Kim, "Shortcut connections based deep speaker embeddings for end-to-end speaker verification system," in *Proc. Interspeech*, 2019, pp. 2928–2932.

[10] A. Hajavi and A. Etemad, "A deep neural network for short-segment speaker recognition," *arXiv preprint arXiv:1907.10420*, 2019.

[11] H. Zhang, J. Xue, and K. Dana, "Deep ten: Texture encoding network," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 708–717.

[12] W. Cai, Z. Cai, X. Zhang, X. Wang, and M. Li, "A novel learnable dictionary encoding layer for end-to-end language identification," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2018, pp. 5189–5193.

[13] Y. Zhu, T. Ko, D. Snyder, B. Mak, and D. Povey, "Self-attentive speaker embeddings for text-independent speaker verification." in *Proc. Interspeech*, 2018, pp. 3573–3577.

[14] 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," in

Advances in Neural Information Processing Systems, 2019, pp. 8024–8035.

[15] 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.

[16] Y. Liu, L. He, and J. Liu, "Large margin softmax loss for speaker verification," in *Proc. Interspeech*, 2019, pp. 2873–2877.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.