DAIC-JHU SPEAKER RECOGNITION SYSTEM DESCRIPTION FOR NIST CTS2020

Kousuke Itakura[1], Ryota Hatakenaka[1], Shintatou Okada[2], Katsunori Daimo[1],
Jesús Villalba[3,4], Najim Dehak[3,4]

[1] Digital&AI Technology Center, Panasonic Corporation, Japan
[2] Connected Solutions Company, Panasonic Corporation, Japan
[3] Center for Language and Speech Processing, Johns Hopkins University, Baltimore, MD, USA
[4] Human Language Technology Center of Excellence, Johns Hopkins University, Baltimore, MD, USA

ABSTRACT

This document describes our system description for the SRE20 CTS Challenge. We developed several types of DNN embeddings with PLDA backend. Performance of our system on in the SRE20 progress set is as follows: the equal error rate (EER) 2.55%, Cmin=0.087 and Cact=0.101.

1. INTRODUCTION

The DAIC-JHU submission is the joint effort of the teams at Panasonic (Panasonic-DAIC) and Johns Hopkins CLSP(JHU-CLSP). Each team explored different x-vector extractors and PLDA backends with different combination of training sets.

2. SYSTEM COMPONENT

2.1 Acoustic Features and Voice Activity Detection

The acoustic features were 64 log-Mel filter banks for all our systems. These features were short-time mean normalized with a 3 seconds window. Silence frames were removed using 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 each window.

2.2 Audio Embeddings

All the x-vector architectures follow the x-vector scheme [1, 2]. In essence, the embedding network consists of an encoder that extracts frame-level discriminant embeddings, a pooling mechanism, and a classification head. In our case, we tried several encoder architectures and used either statistics pooling (mean+stddev) [1] or channel-wise attentive statistics pooling [3]. The network is trained to minimize the categorical cross-entropy loss of the predicted speaker posteriors. We used additive angular margin softmax loss [4] in all our networks. We describe the encoder architectures in the following paragraphs.

2.2.1 Panasonic-DAIC system

- *SE-ResNeXt50*

SE-ResNeXt50[5] is a squeezing and excitation block applied to ResNeXt50, and ResNeXt50 is a branching of the input in each block of ResNet50 into several paths shown in Figure 1. Table 1 shows SE-ResNeXt50 topology.

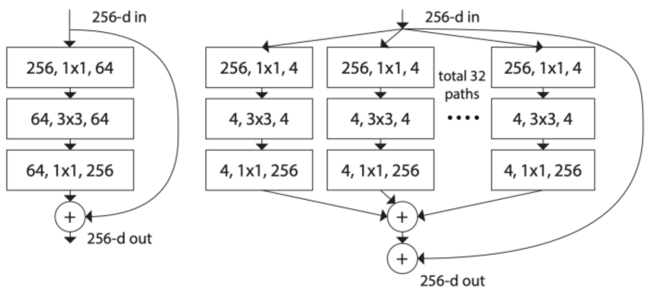
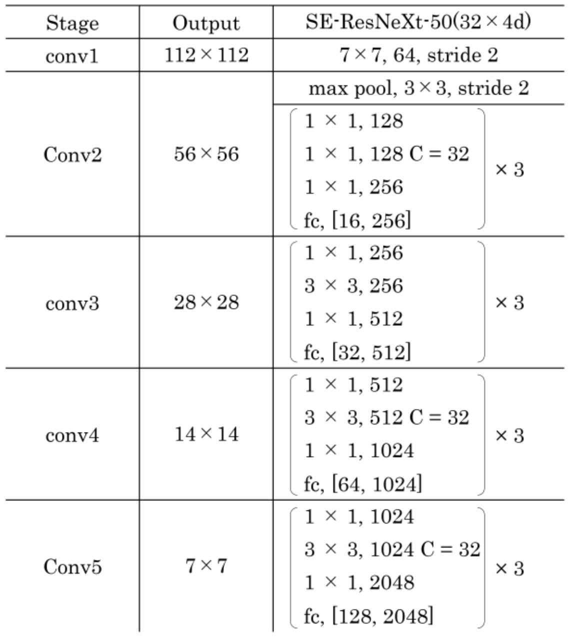


Figure 1: basic brock in ResNet50(left) and basic brock in ResNeXt50(right).

Table 1: the architecture of SE-ResNeXt50



Stage	Output	SE-ResNeXt-50(32×4d)	
conv1	112×112	7×7, 64, stride 2	
Conv2	56×56	max pool, 3×3, stride 2	
		1 × 1, 128 1 × 1, 128 C = 32 1 × 1, 256 fc, [16, 256]	× 3
conv3	28×28	1 × 1, 256 3 × 3, 256 1 × 1, 512 fc, [32, 512]	× 3
conv4	14×14	1 × 1, 512 3 × 3, 512 C = 32 1 × 1, 1024 fc, [64, 1024]	× 3
Conv5	7×7	1 × 1, 1024 3 × 3, 1024 C = 32 1 × 1, 2048 fc, [128, 2048]	× 3

2.2.2 JHU-CLSP system

- *ResNet34*

This encoder is based on the original ResNet34 architecture proposed in [6]. ResNet34 has an input stem layer followed by 16 2D convolutional residual blocks. This architecture downsamples the feature maps 3 times with a stride of 2 (8× total downsampling), at the same time as it multiplies the number of channels in the convolutions.
The output of this network is a four-dimensional tensor
(B, C, F/8, T/8), where B is batch-size, C is number of channels, F is the number of Mel filters and T is time. Channel and frequency dimensions are flattened to (B, C × F/8, T/8) before passing the features to the pooling layer [7].

- *Transformer*

We also tried the Transformer Encoder architecture [8] as Encoder for x-vectors. We used an encoder with 8 self-attention blocks. The input stem uses a two 2D Conv layers that downsample time dimension ×4. We also implemented a local attention procedure that limits the self-attention receptive field to 6 time steps (25 msecs) in each layer. This is similar to the Longformer architecture [9].

- *EfficientNet-b4*

EfficientNet architecture was proposed in [10] for images. This is a residual network that used 2D separable convolutions to reduce the number of multiplications of the network. The work in [10] proposes a base architecture, denoted as EfficientNetb0. Then larger networks EfficientNet-bn are obtained by scaling up number of channels and network depth in such way that EfficientNet-bn is 2^n times more computationally expensive than b0. We found that b4 was needed to improve ResNet34. We also needed to remove the first two feature map downsamplings from the original EfficientNet architecture.

- *Res2Net50*

This encoder is based on the original ResNet50 architecture proposed in [6]. ResNet50 has an input stem layer followed by 16 Bottleneck residual blocks as the one in Figure 2(left). These blocks are based on 2D convolutions. This architecture downsamples the feature maps 3 times with a stride of 2 (8×total downsampling), at the same time that multiplies the number of channels in the convolutions. Res2Net, proposed in [7] replaces the standard bottleneck blocks by Res2Net blocks in Figure 2(right). Res2Net divides the channels in the bottleneck layer into several groups–this is known as the scale parameter. Each group (except the first one, which is just copied in the output) passes through a 3×3 convolution and is added to the input of the convolution of the next group. Hence, each group observed a different receptive field. In our networks, we set the scale to 4 or 8; and the number of channels in each group (width) was set to 26 for the first Res2Net block, following [7]. Each time the network downsamples the feature maps, we duplicate the width of the Res2Net groups.

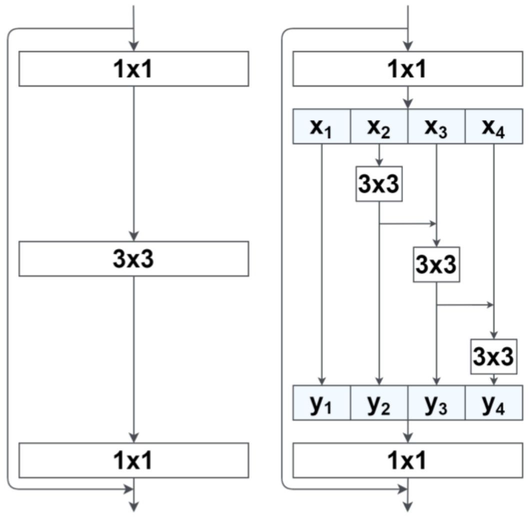


Figure 2: ResNet50 standard bottleneck blocks (left) and Res2Net50 bottleneck block (right)

2.3 Back-end

- *PLDA backend*

The pipeline for this back-end included LDA, centering, whitening, length normalization, and generative Gaussian SPLDA.
At first, we train LDA and the Whitening step, and then, we apply length normalization.
PLDA is trained on length normalized embeddings.

- *kNN-PLDA*

The idea of this back-end consists of training a back-end model adapted to each trial. The motivation is that we do not know the number of domains in our eval data and also, we do not know if all of those domains match any of the domains in our training and adaptation data. Thus, a PLDA mixture may not work since the eval data may not match any of the components of the mixture. We simplify the problem by assuming that enrollment and test segments belong to the same domain, as indicated in the eval plan. The method consists of training a back-end (including PCA/LDA/centering/whitening/PLDA) model using the k Nearest training speakers to the enrollment segments (1 or 3) of the trial. The enrollment segments are also included in the back-end training. In this manner, even if the trial's domain is not included in the training, the corresponding back-end can be trained using the closest speakers from multiple domains. We also think that this method can benefit from domain adaptation. The number of in-domain neighbors may be too small to train PLDA. Instead, we can train the back-end on a larger number of speaker neighbors k1 and adapt to a smallest (closest) number of speakers neighbors k2.

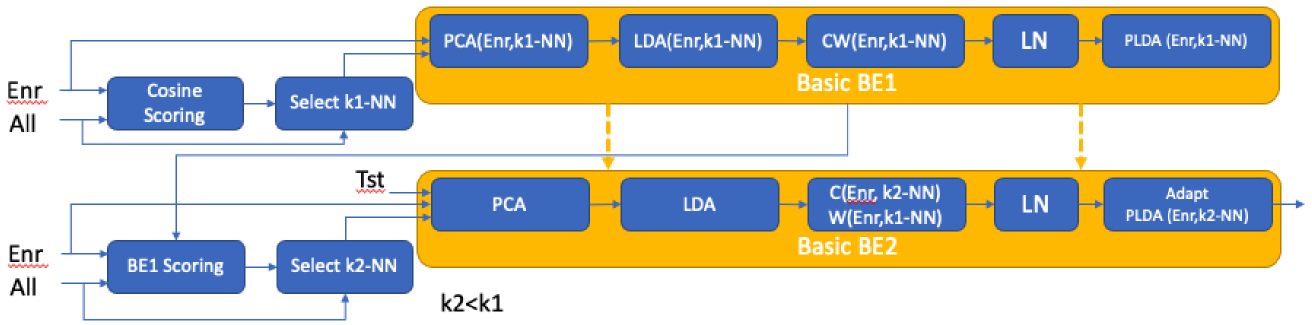


Figure 3: JHU kNN-PLDA-v3 back-end. Enr denotes the enrollment segments, Tst denotes the test segment, and All denotes the full training data.

The procedure is depicted in Figure 3. For each enrollment side, we use cosine scoring to find the k1 closest training speakers. Then, we pool the enrollment segments and all the recordings from those k1 speakers and we train a Basic backend (PCA/LDA/centering/whitening/LNorm), denoted as BE1. Then, we score again the enrollment model versus the training speakers, but this time using BE1 back-end, to find a refined set of in-domain speakers k2 < k1. Then, we use those speakers to adapt BE1's centering and PLDA and produce BE2. In this manner, we train a back-end for each enrollment model, and use that back-end for all the trials that involve that model.
This back-end does not require S-Norm.

2.4 Training datasets

2.4.1 Panasonic-DAIC system

We used the following training datasets:

- Switchboard phase1-3 and cellular1-2.
- NIST SRE04-10.
- MIXER6 telephone phonecalls (MX6-tel).
- NIST SRE18 Dev and Eval.
- NIST SRE19 Eval.
- Fisher Spanish.
- VoxCelebCat 1 + 2: we make VoxCelebCat 1 + 2 by concatenating utterances in VoxCeleb 1 + 2 because the utterances are very short.
- LibriSpeech ASR corpus.
- VCTKcat: we make VCTKcat by concatenating utterances in CSTR VCTK Corpus because the utterances are very short.

We made 2 combinations of the above datasets to train 2 models on SE-ResNeXt50. One combination (named Pana-C1) is the combination of all of the above datasets, and we applied GSM and AMR-NB telephone codecs to VoxCelebCat 1 + 2, LibriSpeech, and VCTKcat using SoX. The other combination (named Pana-C2) is the combination of All datasets except Fisher Spanish, and we applied GSM and AMR-NB telephone codecs to VoxCelebCat 1 + 2. All the training data are augmented by babble, noise, reverberation and music.
For PLDA back-end training, we used NIST SRE04-19 and Fisher Spanish. We did not use any data augmentation.

2.4.2 JHU-CLSP system

We used the following training datasets:

- Switchboard phase1-3 and cellular1-2.
- NIST SRE04-10.
- NIST SRE12 telephone data (SRE12-tel).
- MIXER6 telephone phonecalls (MX6-tel).
- NIST SRE16 Dev: This is the NIST SRE16 development set. It contains 668 recordings from 10 Mandarin speakers and 659 recordings from 10 Cebuano speakers.
- NIST SRE16 Eval 60%: This set contains 60% of the speakers in the NIST SRE16 evaluation set. The rest 40% was kept for development. This set contains 3299 recordings from 60 Cantonese speakers and 2904 recordings from 61 Tagalog speakers.
- NIST SRE18 Dev: This set contains 1741 recordings from 25 Tunisian Arabic speakers.
- NIST SRE18 Eval: This set contains 13451 recordings of 188 Tunisian Arabic speakers.
- Fisher Spanish: This set contains 1638 recordings from 136 Spanish speakers. Several Spanish accents are included.
- VoxCeleb 1+2: This dataset contains 7365 speakers audiofrom video. The original distribution of VoxCeleb splits 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. After concatenation, we obtain 173088 recordings. We applied GSM and AMR-NB telephone codecs to this data using SoX.

We trained our x-vectors on the combination of the datasets above with a total of 304k recordings from 13466 speakers. For x-vector training, we augmented speech on the fly with noise and reverberation. Impulse responses for augmentation were obtained from the Aachen impulse response database (AIR). The noises were acquired from the MUSAN corpus. We used the same SNR levels as in the Kaldi recipes.

For PLDA back-end training, we used NIST SRE04-18 and Fisher Spanish. We did not use any data augmentation.

3. FUSION AND CALIBRATION

Fusion and Calibration was performed using linear logistic regression with the original python script. We pre-calibrated the scores for each single system separately. After fusing, we re-calibrate the fusion score. We used the following dataset to train calibration and fusion.

- NIST SRE16 Eval YUE40%: This set contains 40% of the speakers in the NIST SRE16 evaluation set. We kept the same trial list as in the original SRE16 but keeping only the trials involving those 40 speakers. In total, there are 158k YUE trials.

4. PERFORMANCE

The best fusion system of submission for CTS challnge consisted of 7 systems in Table .

Table 2: *Submission systems*

No.	system	Training Set	Backend
1	SE-ResNeXt50	Pana-C2	PLDA
2	SE-ResNeXt50	Pana-C1	Knn
3	SE-ResNeXt50	Pana-C2	Knn
4	ResNet34	JHU	Knn
5	Transformer	JHU	Knn
6	EfficientNet	JHU	Knn
7	Res2Net50	JHU	Knn

5. SUBMISSION SYSTEM

Table shows our submission system results. Our primary system on the SRE20 progress set is: the equal error rate (EER) 2.55%, Cmin=0.087 and Cact=0.101.

Table 3: Our Submission System Results on SRE20 prog.

	EER	minC	actC
fusion	2.55 %	0.087	0.101

6. COMPUTATIONAL 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. Filter banks, VAD and back-end processing time are negligible in comparison. Processing time is shown in Table 4.

Table 4: *Processing times*

Embedding	Real time factor	Memory (GB)
SE-ResNeXt50	0.525289	1.2
ResNet34	0.0048	1
Transformer	0.0063	1
EfficientNet	0.008	2
Res2Net50	0.0064	1.6

7. REFERENCES

[1] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, “Deep Neural Network Embeddings for Text-Independent Speaker Verification,” *in Proc. INTERSPEECH* 2017.

[2] D. Snyder, D. Garcia-Romero, G. Sell, A. McCree, D. Povey, and S. Khudanpur, “X-Vectors: Robust DNN Embeddings for Speaker Recognition,” *in Proc. IEEE ICASSP*, 2018.

[3] B. Desplanques, J. Thienpondt, and K. Demuynck, “Ecapa-tdnn: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification,” *in Proc. INTERSPEECH*, 2020.

[4] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, “Arcface: Additive angular margin loss for deep face recognition,” *in Proc. CVPR*, 2019.

[5] J. Hu, L. Shen, G. Sun, “Squeeze-and-Excitation Networks” *in Proc. CVPR*, 2018.

[6] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” dec 2015.

[7] H. Zeinali, S. Wang, A. Silnova, P. Matejka, and O. Plchot, “But system description to voxceleb speaker recognition challenge 2019,” *in VoxSRC Challenge workshop*, 2019.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *in Advances in neural information processing systems*, 2017

[9] I. Beltagy, M. E Peters, and A. Cohan, “Longformer: The long-document transformer,” arXiv preprint arXiv:2004.05150, 2020.

[10] M. Tan and Q. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks,” in Proceedings of the 36th International Conference on Machine Learning, Kamalika Chaudhuri and Ruslan Salakhutdinov, Eds. 09–15 Jun 2019, vol. 97 of Proceedings of Machine Learning Research, pp. 6105–6114, PMLR.