THE BGU NIST 2020 CTS SPEAKER RECOGNITION CHALLENGE SYSTEM

Shani Klein, Bar Madar, Haim Permuter

Department of Electrical Engineering, Ben-Gurion University, Beer-Sheva, Israel
shanikle@post.bgu.ac.il, haimp@bgu.ac.il

ABSTRACT

This document briefly describes the systems submitted by BGU team of Ben-Gurion University for the NIST SRE 2020 CTS challenge. All systems based on deep neural network (DNN) speaker embeddings. In this paradigm, a DNN maps variable-length speech segments to speaker embeddings, called x-vectors . We used several x-vector-based systems that differed primarily in the DNN architecture, temporal pooling mechanism, and training objective function. After speaker embedding is extracted, we apply domain adaptation and classify the embedding using probabilistic linear discriminant analysis model (PLDA).

On the evaluation set, our best single-system submission used a Resnet architecture (using LDE and Angular Softmax), and achieved EER of 3.55% , min DCF of 0.147 and actual DCF of 0.153 . A fusion of all four x-vector systems was our primary submission, and it obtained EER of 2.85%, min DCF of 0.124 and actual DCF of 0.132 .

1. INTRODUCTION

The main task in the CTS Challenge is speaker detection, i.e., determining whether a specified target speaker is speaking during a given segment of speech .
NIST SRE 2020 CTS challenge uses CTS recordings extracted from multiple data source . Unlike the 2019 CTS Challenge, this challenge containing multilingual speech and no development set was initially released.

In this paper, we analyze the BGU submission to NIST SRE CTS 2020 and the effect of the unknown multilingual evaluation data set on the performance of the systems.

All our systems consisted of a Deep neural network embedding (x-vector)[1], We explored several types of x-vectors differing in network topology and pooling methods inculding TDNN , Extended-TDNN(ETDNN), factorized TDNN, and ResNet. We also tested mean plus standard deviation; learnable dictionary encoder (LDE). The system give scoring PLDA [2] back-end. We adapted the embeddings features and the back-end models to the multilingual conditions using combinations of data sets consist different languages with adaptation algorithms such PLDA adaptation, LDA, whitening and correlation alignment (CORAL).
The rest of the paper is organized as follows. Section 2 describes the data we used to train the models and description of the augmentation algorithms we used. Section 3 describes the acoustic features and VAD. Section 4 discusses the Deep speaker embedding (x-vector) extractor variants. Section 5 describes the back-end and the multilingual adaptations techniques. Section 6 summarizes the calibration, fusion and normalization. Section 8 presents and analyzes the results. Finally, Section 9 shows the conclusions.

We adapted the embeddings features and the back-end models to the multilingual conditions using combinations of data sets consist different languages with adaptation algorithms such PLDA adaptation, LDA, whitening and correlation alignment (CORAL). These systems considered the current state-of-the-art in text-independent speaker recognition technology.

2. DATA SETS

Table 1 lists the different datasets that we used for training and optimizing the system. Four different corpus are used: SRE+SWBD+LRE+MX6 corpus that Combines Switchboard, LRE, SRE' 04, 05, 06, 08 and Mixer6, used for the PLDA training and Backend components. Voxceleb corpus that Combines Voxceleb 1 and 2 for the embedding extractor training and PLDA training. SRE18+19 that Combines SRE18 eval, SRE18 Unlabeled and SRE19 eval, for backend components. While the SRE, SWBD and Mixer6 data sets are sampling at 8Khz, the Voxceleb data set was downsampled from 16Khz to 8 Khz using SOX.

2.1. Data augmentation

Augmentation increases the amount and diversity of the existing training data. Our strategy employs additive noises and reverberation. Reverberation involves convolving room impulse responses (RIR) with audio. We use the simulated RIRs described in [3], and the reverberation itself is performed with the multi-condition training tools in the Kaldi recipe [4]. For additive noise, we use the MUSAN dataset, which consists of over 900 noises, 42 hours of music from various genres and 60 hours of speech from twelve languages [5]. We use a 3-fold augmentation that combines the original "clean" training list with two augmented copies. To augment a recording, we choose between one of the following randomly:

- Babble: Three to seven speakers are randomly picked from MUSAN speech, summed together, then added to the original signal (13-20dB SNR).
- Music: A single music file 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).
- Reverb: The training recording is artificially reverberated via convolution with simulated RIRs.

Name	Corpora
SRE_SWBD_LRE_MX6	SRE'04, 05, 06, 08 Switchboard-2 Phase I Switchboard-2 Phase II Switchboard-2 Phase III Switchboard Cellular Part 1 Switchboard Cellular Part 2 LRE 2011 test set Mixer 6
VOXCELEB	Voxceleb 1 Voxceleb 2
SRE18_19	SRE18 eval + unlabeled SRE19 eval

Table 1. Corpora used for training the systems.

3. ACOUSTIC FEATURES AND VAD

3.1. feature extraction

for the TDNN based system the features are 23 MFCCs with a frame length of 25ms every 10ms using a 23 channel mel-scale filterbank spanning the frequency range 20Hz-3700Hz. for the system based on ResNet we used 40 log-Mel filter-banks. All the feature vectors are mean-normalized over a sliding window of up to 3 seconds.

3.2. VAD and selecting frames

An energy-based with configuration in kaldi recipe - sre16/v1 is employed to filter out non-speech frames from the utterances. Features that are too short after removing silence were removed (at least 5s(500 frames) per utterance).

4. DEEP SPEAKER EMBEDDINGS (X-VECTOR)

speaker embedding is a fixed vector that represent each speaker regardless of the length of the speaker's utterance. To extract the embedding we used transfer learning method. We trained a model to classified speakers from a close data set (Voxceleb). The models are divided into an encoder (DNN + pooling layer + Fully connected layers) and classifier (softmax or angular-softamx layers). After the model is well training, we cut the classifier part, and the FC layer become to be the output that extract the speaker embedding (X-vector) of a new utterance. Each system is used different DNN encoder and classifier as follow:

4.1. TDNN based Encoder networks

4.1.1. TDNN X-vector system

We used Time delay neural network connected to a statistics pooling layer that aggregates the frame-level TDNN outputs and calculates the mean vector as well as the second-order statistics as the standard deviation vector over the features to output a fixed 512-dimensional utterance-level x-vectors. Specifically, we trained the TDNN architecture mention in [1].

4.1.2. Extended TDNN X-vector System

Etdnn system is an extended version of tdnn. It has wider context and interleaving dense layers between each two tdnn layers. The ETDNN configuration is outlined in table 2.

4.1.3. F-TDNN with skip connections

The factorized TDNN (F-TDNN)[6], reduces the number of parameters of the network by factorizing the weight matrix of each TDNN layer into the product of two low-rank matrices. The first matrix is constrained to be semi-orthogonal in order to retain the main information. To further reduce the risk of gradient vanishing of deeper networks,FTDNN introduces skip connections between the low-rank interior layers, where previous layers are concatenated to form the input of the current layer. In general, skip connections is performed by using a vector addition between the connected layers as performs in the "Resnet" architecture [7]. In our case the skip connections are performed via concatenation of the vectors so as to ensure maximum information flow between layers in the network, inspired by the "Densnet" architecture[8]. The FTDNN configuration is outlined in table 3 In summary, our F-TDNN consisted of a TDNN layer of kernel size 5 and 512 channels; 8 F-TDNN layers with 1024 channels and bottelneck dimension of 256; and a fully connected layer with dimension 2048. The time offsets in the splicing for the F-TDNN layers are (2,0,2,0,2,2,2,0). Layer 5 receives skip connections from layer 3; Layer 7 receives skip connections from layers 2,4; Layer 9 receives skip connections from layers 4,6. The pooling layer and the classifier are the same as the TDNN system.

4.2. Resnet34-LDE encoder network

TDNN layers are replaced by a residual network with 2D convolutions. We used a residual network with 34 layers (ResNet34) and the pooling layer is replaced by a learnable dictionary encoding (LDE)layer[9]. The original x-vector framework assumes that the frame-level TDNN representations before the pooling layer are unimodal. Thus, to pool those representations, we just compute their mean and standard deviation to obtain a single vector per utterance. Instead, the LDE pooling assumes that frame-level representations are GMM distributed in C clusters and it learns a dictionary with the centers of those clusters. In more details, Given a set of L frames feature sequence $\{\mathbf{x}_1, \mathbf{x}_2..., \mathbf{x}_L\}$ and a learned dictionary center $\boldsymbol{\mu} = \{\boldsymbol{\mu_1}, ..., \boldsymbol{\mu_C}\}$. We define the smoothing factor Sc for each dictionary center $\boldsymbol{\mu_c}$ to be learnable:

$$w_{tc} = \frac{\exp\left(-s_c||\mathbf{r_{tc}}||^2\right)}{\sum_{m=1}^{C}\exp\left(-s_m||\mathbf{r_{tm}}||^2\right)} \quad (1)$$

Now, each frame-feature x_t can be assigned with a weight w_{tc} to each component µc and the corresponding residual vector is denoted by $\boldsymbol{r_{tc}} = \boldsymbol{x_t} - \boldsymbol{\mu_c}$, where $t = 1, ...L$ and $c = 1, ...C$. The residual encoding model applies an aggregation operation for every dictionary component center µc:

$$\boldsymbol{E_c} = \sum_{t=1}^{L} \boldsymbol{e_{tc}} = \frac{\sum_{t=1}^{L}\left(-S_c||\boldsymbol{r_{tc}}||^{\mathbf{2}}\right)}{\sum_{t=1}^{L} w_{tc}} \quad (2)$$

The LDE layer concatenates the aggregated residual vectors with assigned weights. The resulted encoder outputs a fixed dimensional representation $\boldsymbol{E} = \{\boldsymbol{e_1}, ..., \boldsymbol{e_C}\}$ (independent of the sequence length L). Instead of using the cross-entropy loss function for training,we used the angular softmax loss [10]. The angular softmax loss has stronger requirements for correct classification when $m \geq 2$ (an integer that controls the angular margin), which generates an angular classification margin between embeddings of different classes. Table 4, shows our basic Resnet based speaker embedding extractor topology. The pooling method on this architecture is mean and std so its double the size of the input vector ($128 \longrightarrow 256$). The Resnet based encoder was implemented in Pytorch while the other systems were implemented in Kaldi.

5. BACKEND

In order to deal with the multilingual challenge, we used adaptation algorithms(LDA, Whitening, CORAL) but instead of adapt out-of-domain data to in-domain, we generate a data set consists portions from augmented VOXCELEB, SRE, SRE18+19,Mixer6, LRE, SWBD (1M utterances in total). This data set is multilingual, and we used all the adaptation algorithms to adapt the classifier(PLDA) training data to this multilingual data. It makes our systems to be more robustness to a multilingual data.

5.1. CORAL

Correlation Alligment (CORAL) is used to adapt the x-vectors of the out-of-domain data to in-domain. Let $\boldsymbol{C_O}$ and $\boldsymbol{C_I}$ be the covariance matrices of the OOD and InD data, respectively. Denote $\boldsymbol{X}$ as the OOD x-vector, the CORAL is performed by first whitening and then re-coloring, as follows:

$$\boldsymbol{X_{coral}} = \boldsymbol{C_I^{\frac{1}{2}}} \boldsymbol{C_O^{-\frac{1}{2}}} \boldsymbol{X} \quad (3)$$

To robust our systems for a multilingual data, we build a data set consist of portions from voxceleb, SRE, SRE18-19 and swbd (1M utterances at all) and adapt the x-vectors to this multilingual combined data-set in order to reduce the difference between the different domains.

5.2. LDA and whitening

Dimensional reduction (LDA) from 512 to 150 is performed using linear discriminant analysis (LDA). Whitening and centering processes applying on all the x-vectors extracted from the DNN

5.3. PLDA

As a clssifier, we used a Gaussian PLDA model as describe in [11] with a full-rank Eigenvoice subspace is used, trained with the data mention in 2.

5.4. PLDA adaptation

The PLDA was adapted using the SRE18+19 augmented data as describe in [12] Although the multilingual test set and instead of the backend adaptation data that describes in 5, we found that PLDA adaptation with this data improve the systems performances the most.

6. CALIBRATION, FUSION AND NORMALIZATION

- Fusion and calibration: Fusion and calibration was performed using linear logistic regression with the Bosaris toolkit[13]. To select the best fusion, we implemented a greedy fusion scheme. First, we calibrated all the systems and select the one with the lowest actual cost. Then, we evaluated all the two-system fusions that include that best system. Thus, we got the best two systems fusion. We fixed those two systems and then add a third system, and so on.

- Normalization: we create our own cohorts based on previews competitions , then we used AS-NORM to normalize and calibrate.

7. RESULTS

We list the performance of all single systems on SRE20Eval in Table 2. All results were computed by the official NIST scoring tool. The primary measure metric is the act-Cost, which is an average of detection cost functions (DCFs) at priori probabilities 0.05. The best single system is the Resnet based,which achieves 3.40% EER, 0.153 min-DCF, and 0.336 act-DCF on SRE20 evaluation set. Our primary submission is the fusion of 3 systems as describes on 6, including ETDNN, FTDNN and Resnet. The primary system yields 2.67% EER, 0.134 min-DCF, and 0.209 act-DCF on the evaluation set of SRE20.

8. CPU EXECUTION TIME

All tasks were performed on 64bit linux with 512G RAM and two Intel Xeon Silver 4114 2.20GHz. All CPU times are counted based on one core CPU. The time it took to process a single trial is 28 seconds.

Layer	Layer Type	Layer context	Total context	Input x output
1	TDNN-ReLU	[t-2,t+2]	5	120x512
2	Dense-ReLU	{t}	5	512x512
3	TDNN-ReLU	{t-2,t,t+2}	9	1536x512
4	Dense-ReLU	{t}	9	512x512
5	TDNN-ReLU	{t-3,t,t+3}	15	1536x512
6	Dense-ReLU	{t}	15	512x512
7	TDNN-ReLU	{t-4,t,t+4}	23	1536x512
8	Dense-ReLU	{t}	23	512x512
9	Dense-ReLU	{t-5,t,t+5}	33	536x512
10	Dense-ReLU	{t}	33	512x512
11	Dense-ReLU	{t}	33	512x512
12	Dense-ReLU	{t}	33	512x1500
13	Stats-Pooling	[0,T)	T	1500Tx3000
14	**Dense-ReLU**	**{0}**	**T**	**3000x512**
15	Dense-ReLU	{0}	T	512x512
16	Dense-softmax	{0}	T	512xN

Table 2. The x-vector speaker embedding extractor ETDNN architecture. The speaker embedding vector, extracted from Layer 14 (in bold) The size dimensions are adjusted for input utterance consists of T frames, and close trainig set consists of N speakers

Layer	Layer Type	Context factor 1	Context factor 2	Skip conn. from layer	Size	Inner size
1	TDNN-ReLU	[t-2,t+2]	-	-	120x512	-
2	F-TDNN-ReLU	{t-2,t}	{t,t+2}	-	1536x1024	256
3	F-TDNN-ReLU	{t}	{t}	-	1024x1024	256
4	F-TDNN-ReLU	{t-3,t}	{t,t+3}	-	3072x1024	256
5	F-TDNN-ReLU	{t}	{t}	3	2048x1024	256
6	F-TDNN-ReLU	{t-3,t}	{t,t+3}	-	3072x1024	256
7	F-TDNN-ReLU	{t-3,t}	{t,t+3}	2,4	5120x1024	256
8	F-TDNN-ReLU	{t-3,t}	{t,t+3}	-	3072x1024	256
9	F-TDNN-ReLU	{t}	{t}	4,6,8	4096x1024	256
10	Dense-ReLU	{t}	-	-	1024x2048	-
11	Stats-Pooling	full-seq	-	-	2048Tx4096	-
12	**Dense-ReLU**	**{0}**	-	-	**4096x512**	-
13	Dense-ReLU	-	-	-	512x512	-
14	Dense-Softmax	-	-	-	512xN.spks	-

Table 3. The speaker embedding extractor FTDNN architecture. The speaker embedding vector, extracted from Layer 12 (in bold)

Layer	**Layer Type**	**Output Size**	**Downsample**	**Channels**	**Blocks**
1	Conv1	$F \times T$	No	16	-
2	ResBlock	$F \times T$	No	16	3
3	ResBlock	$F/2 \times T/2$	Yes	32	4
4	ResBlock	$F/4 \times T/4$	Yes	64	6
5	ResBlock	$F/8 \times T/8$	Yes	128	3
6	Pooling	256×1	-	1	-
7	**Dense-ReLu**	1024×1	-	-	-
8	Dense-ReLu	1×1024	-	-	-
9	Dense-Softmax	$1 \times N.spks$	-	-	-

Table 4. The speaker embedding extractor Resnet architecture. ”F” is the size of the frame-level feature vector, ”T” is the sequence length(number of frames). The speaker embedding vector, extracted from Layer 7 (in bold)

System	Approach	Speaker Embedding Training Data	Back-end Training Data	Set	EER(%)	min_C	act_C
BGU 1	TDNN	Voxceleb	SRE18/19	SRE20 Test	8.27	0.521	0.644
BGU 2	ETDNN	Voxceleb SRE_SWBD_LRE_MX6	SRE_SWBD_LRE_MX6 SRE18/19	SRE20 Test	3.73	0.164	0.181
BGU 3	FTDNN + skip connections	Voxceleb SRE_SWBD_LRE_MX6	SRE_SWBD_LRE_MX6 SRE18/19	SRE20 Test	3.84	0.159	0.178
BGU 4	Resnet+LDE + Angular-softmax	Voxceleb SRE_SWBD_LRE_MX6	SRE_SWBD_LRE_MX6 SRE18/19	SRE20 Test	3.55	0.147	0.153
FUSION	**BGU 2-4**			**SRE20 Test**	**2.85**	**0.124**	**0.134**

Table 5. EER: Equal Error Rate, min_C:Minimum Decision Cost Function, act_C: actual Decision Cost Function. all the x-vector extractors trained with augmented data. the ”with augmentation” refer to the PLDA adaptation in-domain data.