System description of team RRI MRC for FFSVC 2020

Anonymous Author[1]

[1]Anonymous Affiliation

author@mail.com

Abstract

In this report, we describe the submission of team RRI MRC to the FFSVC 2020.

Index Terms: speaker verification, speaker embeddings

1. Introduction

The FFSVC 2020 includes three tasks: two text-dependent far-field speaker verification tasks and one text-independent far-field speaker verification task. However, our solutions for all tasks are based on text-independent system. Therefore, we will focus on the core text-independent scenario and discuss the differences between three individual solutions.

2. Experimental setup

2.1. Training data

We used several datasets during the challenge: FFSVC 2020 challenge dataset, HI-MIA (SLR85) [1], CN-Celeb (SLR82) [2], Aishell (SLR33) [3], VoxCeleb1 [4] and VoxCeleb2 [5]. Specifically, for all three tasks we've started with a model, trained on VoxCeleb1 and VoxCeleb2. For task 1 we fine-tuned the model on FFSVC 2020 and HI-MIA datasets. For task 2, the fine-tuning was done on FFSVC 2020, HI-MIA, CN-Celeb and Aishell datasets. Finally, we fine-tuned the model on FFSVC2020, HI-MIA and CN-Celeb datasets for task 3.

2.2. Development data

Initially we used the development set trial lists, provided by the organizers of the challenge. However, we found them to be quite "simple" and not comparable to the public part of test trials in terms of evaluation metrics. Therefore, we generated our own trial lists in accordance with original structure and containing 1567500 / 2585900 / 1567500 trials for tasks 1 to 3 respectively. Evaluation metrics acquired using new development set trial lists were more comparable to actual evaluation metrics on public part of the test set.

3. System description

Our approach is based on recent advances in the field of speaker recognition where deep neural networks (DNNs) play an important role.

3.1. Pre-processing

We followed the Voxceleb recipe[1] from Kaldi for training DNNs used to extract utterance-level speaker embeddings. The training data was augmented with reverbed additive noise using samples from Musan music[2] and RIRs from Room Impulse Response and Noise Database[3]. In addition we added extra far-field samples, processed via beamforming (BeamformIt[4]) and weighted prediction error (NARA-WPE [6]) algorithms to the training data.

We tried to use other kinds of augmentations during model selection as well, such as using other kinds of noises from Musan, reverberation for near-field samples, volume augmentation, tempo change, but it did not lead to noticeable improvements on both original and extended trial lists. Voice activity detection (VAD) was omitted for the same reason.

3.2. Features

We used 40-dimensional log filterbanks as feature representations of audio signal to train different speaker embedder networks. Our implementation is based on the open-source `python_speech_features`[5] package. Both features were extracted from signal frames of 25ms length with 10ms shift. Frequency limits were set to 20-7600 Hz. Both features were mean normalized within the entire feature sequence.

3.3. Embedding extractors

We used ResNet34 topology from [7] as the basis from our solution due to its superior peformance compared to the x-vector approach in the recent VoxCeleb SRC Challenge 2019 [8]. It uses statistical pooling which accumulates mean and standard deviation statistics for the frame-level outputs to get a fixed-dimensional utterance-level representation. The ResNet34 model is trained entirely on single channel data.

We compared a few alternative ResNet architectures including ResNet50 and found that the one used in [9] yields the best performance on the development data. We selected it for our experiments. Our implementations are based on the PyTorch framework [10].

Each network was initially trained on VoxCeleb1 and VoxCeleb2 datasets, and then fine-tuned on datasets, corresponding to a given task. It was done mostly to save computational resources and time spent on individual experiments.

We followed the two-stage training strategy described in [9]. First, the network was trained on VoxCeleb1 and VoxCeleb2 datasets with the standard Softmax loss. Second, on the fine-tuning stage, the additive angular margin loss (further referred to as AAM-Softmax) was used after replacing all the layers following the embedding layer. We used the AAM-Softmax loss with scale $s = 30$ and margin $m = 0.25$.

We used chunks of 600 frames for training the ResNet embedder. These chunks were obtained by random cropping of the training segments. In the testing stage, embeddings were extracted from the full-length feature sequences without any crop-

[1] https://github.com/kaldi-asr/kaldi/blob/master/egs/voxceleb/v2/

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

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

[4] https://github.com/xanguera/BeamformIt

[5] https://pypi.org/project/python_speech_features/

ping.

The network was trained using SGD with momentum=0.9 and weight decay=0.0001. In addition, we used different learning rate for layers during fine-tuning. Starting from the last layer, the learning rates were reducing by a factor of 10 for each following layer.

3.4. Backend

For the ResNet based embeddings we used cosine similarity based scoring. No additional score normalization was done. Previously we've also found out that LDA-PLDA based scoring does not help when embeddings are extracted with ResNet, therefore, we used raw cosine similarity-based scores. To acquire embeddings of multichannel utterances we've averaged embeddings for individual channels.

4. References

[1] X. Qin, H. Bu, and M. Li, "Hi-mia : A far-field text-dependent speaker verification database and the baselines," 2019.

[2] Y. Fan, J. Kang, L. Li, K. Li, H. Chen, S. Cheng, P. Zhang, Z. Zhou, Y. Cai, and D. Wang, "Cn-celeb: a challenging chinese speaker recognition dataset," 2019.

[3] X. N. B. W. H. Z. Hui Bu, Jiayu Du, "Aishell-1: An open-source mandarin speech corpus and a speech recognition baseline," in *Oriental COCOSDA 2017*, 2017, p. Submitted.

[4] A. Nagrani, J. S. Chung, and A. Zisserman, "VoxCeleb: A large-scale speaker identification dataset," in *Proc. Interspeech 2017*, 2017, pp. 2616–2620.

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

[6] L. Drude, J. Heymann, C. Boeddeker, and R. Haeb-Umbach, "Nara-wpe: A python package for weighted prediction error dereverberation in numpy and tensorflow for online and offline processing," in *Speech Communication; 13th ITG-Symposium*. VDE, 2018, pp. 1–5.

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

[8] J. S. Chung, A. Nagrani, E. Coto, W. Xie, M. McLaren, D. A. Reynolds, and A. Zisserman, "Voxsrc 2019: The first voxceleb speaker recognition challenge," *ISCA Challenges*, 2019.

[9] H. Zeinali, S. Wang, A. Silnova, P. Matejka, and O. Plchot, "BUT system description to voxceleb speaker recognition challenge 2019," *CoRR*, vol. abs/1910.12592, 2019. [Online]. Available: http://arxiv.org/abs/1910.12592

[10] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.