Strategies for End-to-End Text-Independent Speaker Verification

Weiwei Lin[1]*, Man-Wai Mak*[1] *and Jen-Tzung Chien*[2]

[1] Dept. of Electronic and Information Engineering,
The Hong Kong Polytechnic University
[2]Dept. of Electrical and Computer Engineering,
National Chiao Tung University

weiwei.lin@connect.polyu.hk, man.wai.mak@polyu.edu.hk, jtchien@nctu.edu.tw

Abstract

State-of-the-art speaker verification (SV) systems typically consist of two distinct components: a deep neural network (DNN) for creating speaker embeddings and a backend for improving the embeddings' discriminative ability. The question which arises is: Can we train an SV system without a backend? We believe that the backend is to compensate for the fact that the network is trained entirely on short speech segments. This paper shows that with several modifications to the x-vector system, DNN embeddings can be directly used for verification. The proposed modifications include: (1) a mask-pooling layer that augments the training samples by randomly masking the frame-level activations and then computing temporal statistics, (2) a sampling scheme that produces diverse training samples by randomly splicing several speech segments from each utterance, and (3) additional convolutional layers designed to reduce the temporal resolution to save computational cost. Experiments on NIST SRE 2016 and 2018 show that our method can achieve state-of-the-art performance with simple cosine similarity and requires only half of the computational cost of the x-vector network.

Index Terms: Speaker verification; end-to-end speaker embedding; deep neural network; x-vector

1. Introduction

Speaker verification has become an increasingly popular choice for biometric authentication [1]. Depending on whether the transcriptions of the enrollment and the test speech are the same or not, speaker verification can be divided into text-dependent and text-independent. Text-independent SV (TI-SV) has broader applications and is more challenging. This paper focuses on TI-SV.

Recently, deep neural network (DNN) based approaches have gained a lot of attention and showed superior performance compared to i-vector/PLDA [2, 3]. In [4], the authors proposed to use a fully-connected network to process contextual filterbank features. The averaged activation at the last layer was used for cosine-distance scoring. An advanced DNN architecture for SV was proposed in [5], where a network comprising several inception blocks was trained by minimizing the triplet loss.

Among the DNN-based approaches, the x-vector approach [6] is considered as state-of-the-art front-end method. Compared with previous DNN-based approaches, several features of x-vector approach stand out: (1) the use of extensive data augmentation with real-life noise and reverberation, (2) the use of short training segments randomly sampled from training utterances, and (3) the production of segment-level representations by concatenating the mean and the standard deviation of the frame-level activations. Recently, research has shown that replacing the TDNNs or CNNs in an x-vector network by the ResNets [7] or DenseNets [8] can improve performance [9–11]. However, even with these advanced architectures, the above three strategies of the x-vector still play critical roles in achieving good performance. It is worth noting that the embeddings extracted from the x-vector network using full-length utterances are not good enough for speaker verification. A backend model that takes the embeddings as input is required during scoring to account for the non-speaker variability.

An end-to-end system using only an integrated neural network is attractive in several aspects. Firstly, hyper-parameter optimization is easier in an end-to-end system. In an x-vector system, the DNN and the backend are optimized separately, which complicates the hyper-parameter search as validation has to be done for the network and the backend separately. Secondly, although training the backend itself is reasonably fast, preparing the training data (i.e., the x-vectors) for it can be time-consuming. Besides, it is not clear which part of the dataset should be used for backend training. Thirdly, an end-to-end system is easier to deploy and debug. In this paper, we propose three modifications to improve DNN embeddings' discriminative ability without relying on a backend. Firstly, instead of randomly sampling a single segment out of an utterance, multiple segments are sampled from an utterance and spliced together to form a long training segment. Secondly, we introduce additional convolutional layers as learnable pooling layer. This strategy can save computation cost by reducing temporal resolution, which is especially desirable when working with long speech segments. Thirdly, we introduce a mask-pooling layer as a special form of data augmentation inside the network. The mask-pooling layer produces multiple utterance-level representations out of a single speech segment by randomly masking out frame-level activations and then computing the temporal statistics of the remaining activations across time.

2. X-Vector System

An x-vector network consists of three parts: frame-level time delay neural networks (TDNNs), utterance-level fully-connected (FC) layers, and a statistics pooling layer that bridges the frame-level layers and utterance-level layers [6,12]. TDNNs are a particular form of convolutional neural networks (CNNs). TDNN skips the computation at chosen positions while maintaining the same receptive-field size as a CNN. The statistics pooling layer concatenates the mean and the standard deviation of the activations from the last convolutional layer. The concatenated vectors are passed to two FC layers. The network is to minimize the standard cross-entropy loss. The network is trained using small chunks sampled from the entire utterances. After training, each utterance's embedding can be extracted

Table 1: *Architecture of our x-vector network. N_{spk} denotes the number of speakers*

Layer	Size, Stride	Channel_in × Channel_out
Conv0	5, 1	23 × 512
Conv1	2, 2	512 × 512
Conv2	3, 1	512 × 512
Conv3	3, 1	512 × 512
Conv4	2, 2	512 × 512
Conv5	1, 1	512 × 1536
Stats pooling	–	–
FC0	–	3072 × 512
FC1	–	512 × 128
AM-softmax [13]	–	128 × N_{spk}

from the first affine layer after statistics pooling. A backend consisting of LDA and PLDA is trained using the embeddings as input and speaker identities as the labels.

3. Proposed End-to-End Approach

In an x-vector system, the segments for training the network are densely sampled from the full-length utterances. The sampled segments typically range from 200 ms to 400 ms, which are much shorter than the entire utterances. As a result, the embeddings may not be a good representation of long utterances. A simple solution is to use longer segments or directly use the entire utterances for training. This strategy, however, also has problems. Because the training segments are sampled from the entire utterances, long segments have less sample diversity than short segments. Training with long segments alone may lead to overfitting. Another disadvantage is that long segments require more computation and GPU memory. Ideally, we want to have both short and long segments for training. This approach, however, will lead to substantial computational burden. We propose three techniques that reduce the duration discrepancy between the training segments of the x-vector network and the test segments used for deriving the test x-vectors while maintaining the sample diversity and computation cost at a reasonable level.

3.1. Splice Sampling

When training an x-vector network, one segment is sampled from one utterance. A drawback of this sampling approach is that each training segment can only contains a number of consecutive frames. A good speaker embedding should be able to exploit speaker information across a longer time span. Therefore, we propose to sample several chunks of non-consecutive segments from each training utterance and splice them together to form a single training segment as shown in Fig. 1. This approach can diversify training data when using long segments for training.

3.2. CNN Local Pooling

Subsampling operations, such as max-pooling or mean pooling, can significantly reduce the computation cost of a CNN by decreasing the resolution of the input. In addition, max-pooling or mean pooling introduces invariance to translation. Although translation invariance is not very important to speaker embedding as the statistics pooling layer computes the mean and the standard deviation across time, the computational reduction is still very attractive. In this paper, we consider a pooling operation that operates over the time axis.

Denote $\mathbf{I}$ as the feature map to be input to a pooling layer. The max pooling operation takes the maximum value inside the kernel and shifts the kernel with stride S to produce the output $\mathbf{O}$:

$$\mathbf{O}[i,t] = \max_{k=0,\dots,K-1} \mathbf{I}[i, S\times t + k], \tag{1}$$

where k indexes the elements inside the kernel, K is the length of the kernel, i is the channel index, and t is the time index of the output. The mean pooling can be described in a similar manner:

$$\mathbf{O}[i,t] = \frac{1}{K} \sum_{k=0,\dots,K-1} \mathbf{I}[i, S\times t + k]. \tag{2}$$

It was demonstrated in [14] that the pooling operations in Eq. 1 and Eq. 2 can be replaced by a convolutional layer with increased stride without loss in accuracy. Without the pooling layer, given the input feature map $\mathbf{I}$ and the convolutional filter matrix $\mathbf{W}$, the output of the convolutional layer at channel j and frame t is:

$$\mathbf{O}[j,t] = \sum_{k=0,\dots,K-1} \sum_{i} \mathbf{W}[j,i] \cdot \mathbf{I}[i, S\times t + k], \tag{3}$$

where j indexes the output channel and i indexes the input channel. Here we omit the bias term for simplicity. We can see the similarity between the convolutional layer and the pooling layers from Eq. 1–Eq. 3. In fact, if we use a kernel parameterized by $1/K$ with kernel size K and stride S, the convolutional layer would act like mean pooling [14].

We adopt two 1D convolutional layers with kernel size 2 and stride 2 in our DNN as a way to reduce subsequent computation. The architecture of our network is summarized in Table 1. Because the two convolutional layers with stride 2 reduce the temporal resolution by half for the subsequent convolutional layers, for an input sequence of size 23×3000, our network requires only 4.3 GMac (Giga multiply–accumulate operation) to produce an embedding as compared to 8.0 GMac in the original x-vector network.

3.3. Mask-pooling Layer

In addition to high computational cost, another disadvantage of using long training segments is the lack of sample diversity. We propose a novel mask-pooling layer that augments training data without significantly increasing computational cost.

Assume that the activation at the last convolutional layer is $\mathbf{z}_t$, where t is a time index. The mask-pooling layer involves the following operations. First, a mask r_t is sampled from a Bernoulli distribution parameterized by p. Then, we multiply $\mathbf{z}_t$ with r_t to decide whether we keep this frame or not. The resulting frames are denoted as $\{\hat{\mathbf{z}}_t\}$. Finally, the utterance-level representation $\mathbf{x}$ is obtained by concatenating the mean and the standard deviation of $\{\hat{\mathbf{z}}_t\}$ as follows:

$$r_t \sim \text{Bernoulli}(p), \tag{4}$$

$$\hat{\mathbf{z}}_t = r_t \cdot \mathbf{z}_t, \tag{5}$$

$$\mathbf{x} = \text{Concat}(\text{MEAN}(\{\hat{\mathbf{z}}_t\}), \text{STD}(\{\hat{\mathbf{z}}_t\})). \tag{6}$$

Here, MEAN$(\cdot)$ and STD$(\cdot)$ are operated on non-zero elements in $\{\hat{\mathbf{z}}_t\}$. We denote Eqs. 4–6 in whole as:

$$\mathbf{x} = \text{MaskPooling}(\{\mathbf{z}_t\}, p). \tag{7}$$

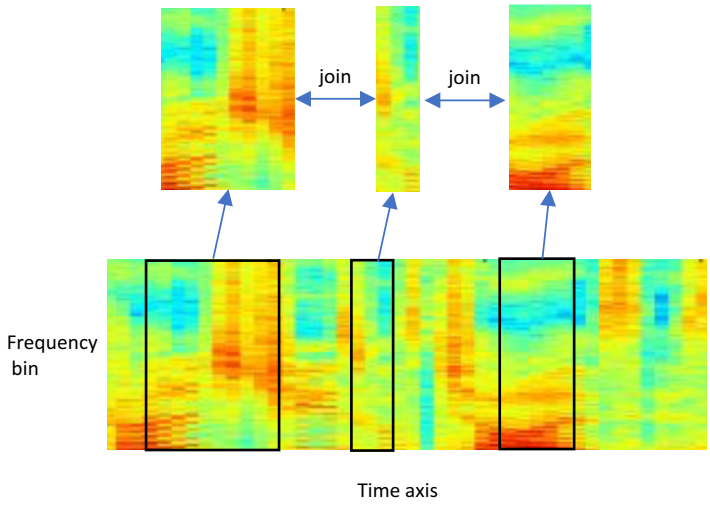

Figure 1: *Splice sampling on a spectrogram. Three chunks are taken out of the spectrogram and spliced together to form a training segment.*

The above operations are very similar to Dropout [15]. However, unlike Dropout, we can repeatedly apply this operations I times with a different p_i sampled from a uniform distribution over 0 to 1:

$$\mathbf{x}_i = \text{MaskPooling}(\{\mathbf{z}_t\}, p_i), \quad i = 1, \ldots, I. \tag{8}$$

Suppose $f(\cdot)$ represents the fully-connected layers and the softmax layer. Assume that the loss function for cross-entropy is denoted by $\mathcal{L}$. Then the loss for these I copies of the augmented data is calculated by

$$\sum_{i=1}^{I} \mathcal{L}(f(\mathbf{x}_i), \mathbf{y}), \tag{9}$$

where $\mathbf{y}$ is the label of all $\mathbf{x}_i$. Different from the data augmentation methods in [6], where noise and reverberation were added to the waveforms, our mask-pooling layer operates on the internal representation of the network. In terms of augmentation effect, mask-pooling produces utterance-level representations with different durations. It is similar to the cutout and spectrum augmentation [16, 17] in that the augmented sampled are produced by withholding information. An important advantage of the proposed method over cutout and spectrum augmentation is that for I augmented samples, there is only one forward propagation to the convolutional layers and I forward propagations through the fully-connected layers.

4. Experiments

4.1. Data Preparation

The training data include NIST SRE 2004–2010 (SRE04–10 in short) and all of the Switchboard data. We followed the data augmentation strategy in the Kaldi SRE16 receipt [6, 18]. The training data were augmented by adding noise, music, reverb, and babble to the original speech files in the datasets. After filtering out utterances shorter than 500 ms and speakers with less than 8 utterances, we were left with 4,808 speakers. 23-dimensional Mel-frequency cepstral coefficients (MFCC) were computed from the 8kHz speech files. Mean normalization was applied to the MFCC using a 3-second sliding window. Non-speech frames were removed using Kaldi's energy-based voice activity detector.

4.2. Training of DNNs and Backend Classifiers

For fair comparisons, all systems under evaluation were trained to minimize the additive margin loss using an Adam optimizer [19] with learning rate set to 0.001. In the x-vector systems, embeddings were extracted from the affine transformation layer after statistics pooling. For the proposed system, the embeddings were extracted from the last fully-connected layer before computing log-softmax. We used a standard backend comprised of LDA, length-normalization, and PLDA. Both LDA and PLDA were trained using the embeddings extracted from full-length utterances. We used correlation alignment [20] for domain adaptation in the PLDA backend. For the end-to-end systems, we applied a whitening transformation to the enrollment data and the test data before cosine-distance scoring. The whitening matrix was estimated using target-domain data. We used AM-softmax with a margin of 0.35 in all the systems.

4.3. Evaluation

All systems were evaluated on the evaluation set of SRE 2016 and 2018. The SRE16 evaluation set is composed of Tagalog and Cantonese telephone conversations. For SRE18, we only conducted the evaluation on the CMN2 portion, which consists of Tunisian Arabic conversations. Both evaluations aim to evaluate the robustness of systems against noise, channel, and language mismatches. We report results in equal error rate (EER) and minimum detection cost function (minDCF). Both metrics were obtained using the scoring tools provided by NIST.

5. Results

We present the performance of the proposed method and x-vector systems on SRE16 and SRE18. We conducted experiments for three different DNNs, namely Xvec_short, Xvec_long and Our_Xvec. Xvec_short refers to the x-vector network trained on 200ms–400ms chunks. Xvec_long refers to the x-vector network trained on 1200ms chunks. Our_Xvec refers to

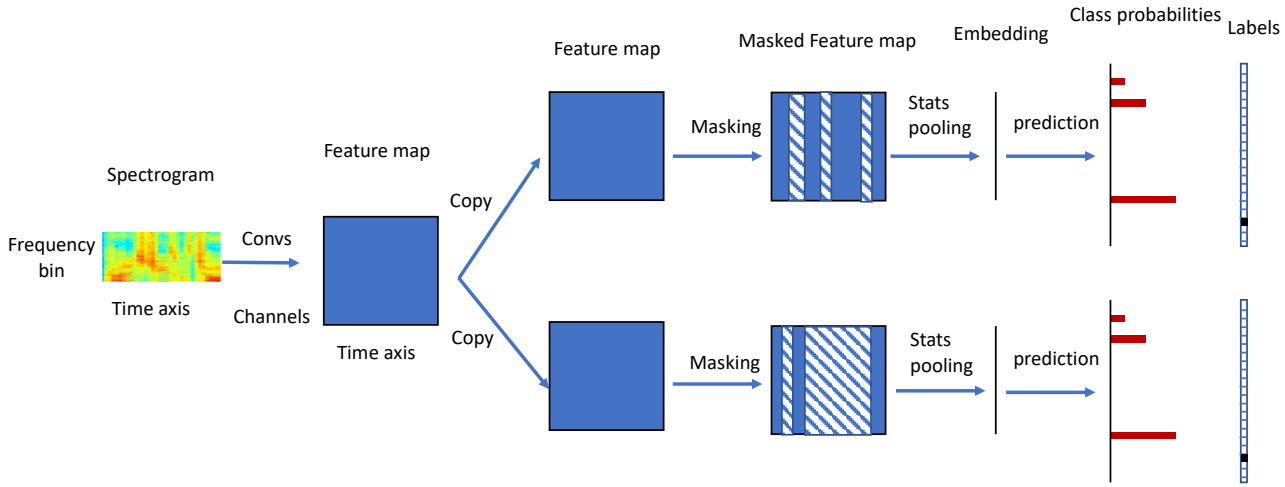

Figure 2: *Illustration of the-mask pooling operation in the proposed x-vector network. For simplicity, only two random masks ($I = 2$ in Eq. 8) are shown.*

Table 2: *Comparison of the performance using x-vector systems and the proposed approach with different scoring methods.*

		SRE16		SRE18	
Front-end	Scoring	EER(%)	minDCF	EER(%)	minDCF
Xvec_short	PLDA	8.34	0.593	8.73	0.556
Xvec_long	PLDA	8.96	0.593	8.83	0.570
Our_Xvec	PLDA	8.90	0.600	8.91	0.598
Xvec_short	Cosine	10.57	0.674	11.98	0.681
Xvec_long	Cosine	9.88	0.653	11.12	0.643
Our_Xvec	Cosine	**8.26**	**0.583**	**8.65**	**0.551**

the x-vector network with the proposed three modifications. We also compared systems that use the PLDA backend with systems that use cosine similarity.

As can be seen from Table 2, when directly using cosine similarity for scoring, Xvec_long outperforms Xvec_short. However, with the PLDA backend, it is the other way around. Still, the best performance for the x-vector systems is obtained by using short training segments with a PLDA backend. The proposed approach (Our_Xvec) outperforms the best x-vector system with simple cosine-distance scoring. Another interesting finding is that the proposed approach does not benefit from the PLDA backend as the x-vector system does, which suggests that a backend is no longer necessary.

6. Conclusions

In this paper, we showed that with three modifications to the x-vector system, we were able to train state-of-the-art speaker verification systems without a backend. The proposed methods not only eliminated the need for the complicated backend but also reduced the x-vector extraction time to about half. In the future, we will further investigate the effect on how the duration mismatch between training and test utterances affects the performance of speaker embeddings.

7. Acknowledgment

This work was supported by RGC of Hong Kong,Grant 152518/16E and 152137/17E, and Taiwan MOST, Grant 109-2634-F-009-024.

8. References

[1] M. Marras, P. A. Marín-Reyes, J. Lorenzo-Navarro, M. C. Santana, and G. Fenu, "Deep multi-biometric fusion for audio-visual user re-identification and verification," in *Proc. ICPRAM*, M. D. Marsico, G. S. di Baja, and A. L. N. Fred, Eds., vol. 11996. Springer, 2019, pp. 136–157.

[2] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[3] P. Li, Y. Fu, U. Mohammed, J. Elder, and S. Prince, "Probabilistic models for inference about identity," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 1, pp. 144–157, 2012.

[4] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *Proc. ICASSP*. IEEE, 2014, pp. 4052–4056.

[5] C. Zhang, K. Koishida, and J. H. Hansen, "Text-independent speaker verification based on triplet convolutional neural network embeddings," *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 26, no. 9, pp. 1633–1644, 2018.

[6] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *Proc. ICASSP*. IEEE, 2018, pp. 5329–5333.

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

[8] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. of IEEE International Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4700–4708.

[9] W. Lin, M. W. Mak, and L. Yi, "Learning mixture representation for deep speaker embedding using attention," in *Proc. Odyssey 2020 The Speaker and Language Recognition Workshop*, 2020, pp. 210–214.

[10] N. N. An, N. Q. Thanh, and Y. Liu, "Deep CNNs with self-attention for speaker identification," *IEEE Access*, vol. 7, pp. 85 327–85 337, 2019.

[11] A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman, "Voxceleb: Large-scale speaker verification in the wild," *Comput. Speech Lang.*, vol. 60, 2020.

[12] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Proc. Interspeech*, 2015.

[13] 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.

[14] J. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," in *Proc. of ICLR*, 2015.

[15] 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.

[16] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[17] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[18] 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. Workshop on Automatic Speech Recognition and Understanding*, 2011.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[20] B. Sun, J. Feng, and K. Saenko, "Correlation alignment for unsupervised domain adaptation," in *Domain Adaptation in Computer Vision Applications*. Springer, 2017, pp. 153–171.