INTERSPEECH 2020
October 25–29, 2020, Shanghai, China

Wav2Spk: A Simple DNN Architecture for Learning Speaker Embeddings from Waveforms

Weiwei Lin and Man-Wai Mak

Dept. of Electronic and Information Engineering
The Hong Kong Polytechnic University, Hong Kong, China
weiwei.lin@connect.polyu.hk, man.wai.mak@polyu.edu.hk

Abstract

Speaker recognition has seen impressive advances with the advent of deep neural networks (DNNs). However, state-of-the-art speaker recognition systems still rely on human engineering features such as mel-frequency cepstrum coefficients (MFCC). We believe that the handcrafted features limit the potential of the powerful representation of DNNs. Besides, there are also additional steps such as voice activity detection (VAD) and cepstral mean and variance normalization (CMVN) after computing the MFCC. In this paper, we show that MFCC, VAD, and CMVN can be replaced by the tools available in the standard deep learning toolboxes, such as a stacked of stride convolutions, temporal gating, and instance normalization. With these tools, we show that directly learning speaker embeddings from waveforms outperforms an x-vector network that uses MFCC or filter-bank output as features. We achieve an EER of 1.95% on the VoxCeleb1 test set using an end-to-end training scheme, which, to our best knowledge, is the best performance reported using raw waveforms. What's more, the proposed method is complementary with x-vector systems. The fusion of the proposed method with x-vectors trained on filter-bank features produce an EER of 1.55%.

Index Terms: Speaker verification; end-to-end speaker embedding; deep neural networks; temporal gating

1. Introduction

For years, machine learning had relied on human knowledge to derive useful features from raw input signals such as speech, images, and text. The human-derived features are referred to as handcrafted features in the literature, such as scale-invariant feature transform (SIFT) [1] in computer vision and mel-frequency cepstrum coefficients (MFCCs) in speech. MFCCs utilize the spectra and human perception of sound. It has long been the de facto features for speech and speaker recognition. In speaker recognition, it has been used in conjunction with Gaussian mixture models, i-vectors, x-vectors, and more recently ResNet and DenseNet speaker embeddings [2–6]. Deep neural networks (DNNs) learn to transform the input data into more abstract and composite representations in successive layers [7]. It has been found that DNNs perform better with raw input than with human engineering features [7].

In the speech community, there is also various success in learning from waveforms. In [8], Zeghidour *et al.* initialized the bottom convolutional layers as an approximation of mel-filter-banks and then fine-tuned the layers jointly with the remaining convolutional layers. They showed that models trained on time-domain filter-banks consistently outperform the ones trained on mel-filter-banks on the phone recognition task. In [9], Ravanelli and Bengio argue that to learn useful features, CNNs should behave like a bandpass filter. They derived a form of CNN kernels that behaves like a bandpass filter in the frequency domain. The network is called SincNet and has shown competitive results in the LibriSpeech dataset for speaker verification. In [10], the authors proposed to first train a CNN based identification network and replace the fully connected layers by a binary classification network for speaker verification. They found that the two-stage approach works better and that the CNN can automatically learn the fundamental frequencies. In [11], an end-to-end speaker verification system was proposed. The authors used a learnable pre-emphasis layer before the first convolutional layer. The method was improved in [12], where the authors used a ResNet architecture and removed inefficient aspects of the multi-step training scheme. Different backend classifiers and losses were also investigated in [12]. In [13], the authors proposed a wav2vec architecture to learn speech features in an unsupervised manner. The network consists of a feature encoder and a feature aggregator. The encoder is used to extract features to replace MFCCs. The encoder is designed in such a way that the outputs correspond to 30ms of speech with a 10ms frame shift, which is consistent with the MFCC features.

In this work, we propose a simple DNN architecture for directly learning speaker embeddings from waveforms. We refer to the proposed DNN architecture as "wav2spk" in the rest of the paper. Our wav2spk includes three components that are designed to replaced MFCC, VAD, and CMVN. Firstly, we follow the design of wav2vec [13] using a feature encoder that is composed of convolutional layers with large stride and kernel size. The kernel size and stride are chosen such that each output sample corresponds to 30ms waveform with 10ms frame shift. Secondly, we introduce a temporal gating unit to replace the role of a VAD. The gating unit chooses the output of the previous layer based on the network objective instead of other arbitrary criteria. Thirdly, inspired by CMVN, we use an instance normalization scheme that normalizes a feature along the temporal axis for each batch and channel.

2. Wav2Spk

2.1. Feature Encoder

Typically, the MFCC features for speaker recognition are computed using a 30-ms window with 10-ms frame shift. As a result, the first layer of the x-vector network already has a large temporal span. The x-vector network is not designed for processing waveforms. If we directly use waveforms as input to the x-vector network, the effective receptive field will be too small. To use waveforms as input to a DNN, it is necessary to use the convolutional layer with large strides and kernel sizes. Here we adopt the encoder used in the wav2vec model, which consists of 5 convolutional layers with kernel sizes (10, 8, 4, 4, 4) and strides (5, 4, 2, 2, 2). As a result, the output of the en-

http://dx.doi.org/10.21437/Interspeech.2020-1287

Table 1: *Summary of the wav2spk architecture.*

Group	Layer	Size, Stride	Channel_in × Channel_out
Feature Encoder	Conv0	10, 5	1 × 40
	Conv1	5, 4	40 × 200
	Conv2	5, 2	200 × 300
	Conv3	3, 2	300 × 512
	Conv4	3, 2	512 × 512
Temporal Gating	–	–	–
Frames Aggregator	Conv5	3, 1	512 × 512
	Conv6	3, 1	512 × 512
	Conv7	3, 1	512 × 512
	Conv8	3, 1	512 × 512
Stats pooling	–	–	–
Utt. Layers	FC0	–	1024 × 512
	FC1	–	512 × 128
	AM-softmax [14]	–	128 × N_SPK

coder network corresponds to 30ms of speech with 10ms frame shift (assuming 16kHz sampling rate).

2.2. Temporal Gating

The feature gating mechanism has been very popular in computer vision [15]. Given an input feature vector $\mathbf{x} \in \mathcal{R}^d$, the gating mechanism produces an output feature vector $\mathbf{y} \in \mathcal{R}^d$ as follows:

$$\mathbf{y} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) \odot \mathbf{x}, \tag{1}$$

where $\odot$ denotes element-wise multiplication, $\mathbf{W}$ is a $d \times d$ weight matrix, and $\mathbf{b}$ is a bias term. The idea is that the elements of $\sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$ can be regarded as weights that signify which features are more useful to the task and thus weight the features accordingly. The idea can be extended to other axis besides the channel.

When using waveforms as input, we can weight each sample in the waveform or individual frames in any layers of the CNN. Let $\mathbf{v}$ be a weight vector of d-dimension and $\mathbf{x}_t$ be the hidden activation in a CNN layer at frame *t*; we have

$$\mathbf{y}_t = \sigma(\mathbf{v}^\mathsf{T}\mathbf{x}_t + b) * \mathbf{x}_t, \tag{2}$$

where $\sigma(\mathbf{v}^\mathsf{T}\mathbf{x}_t + b)$ is a scalar term that indicates the importance of frame $\mathbf{x}_t$ and $*$ denotes the multiplications across all of the feature dimensions. If $\sigma(.)$ is a binary unit, Eq. 2 acts as a VAD inside the network. The difference is that the unit is trained to minimize the network loss instead of other arbitrary criterion. Because the binary unit is not differentiable, we use a sigmoid function. This module can be used in any layer in the network. We experimented with putting it after the feature encoder and before statistics pooling.

2.3. Instance Normalization

An important step after computing the MFCC features is cepstral mean and variance normalization (CMVN). CMVN normalizes the samples across the temporal axis within the window. To replace CMVN, we propose to use instance normalization in each layer of the feature encoder. A normalization layer typically computes the following [16, 17]:

$$\hat{\mathbf{x}}_i = \frac{1}{\boldsymbol{\sigma}_i}\left(\mathbf{x}_i - \boldsymbol{\mu}_i\right), \tag{3}$$

where $\mathbf{x}$ is a feature vector produced by the preceding layer, i is a multi-tuple index, and the division is element-wise. For speech signal in a 1-dimensional convolutional layer, $i = (i_N, i_C, i_T)$ is a 3D vector indexing a feature element in a tensor with axes (N, C, T), where N is the batch axis, C is the channel axis, and T is the time axis. The mean $\boldsymbol{\mu}_i$ and standard deviation $\boldsymbol{\sigma}_i$ are computed by:

$$\boldsymbol{\mu}_i = \frac{1}{m}\sum_{k\in\mathcal{S}_i}\mathbf{x}_k \tag{4}$$

$$\boldsymbol{\sigma}_i = \sqrt{\frac{1}{m}\text{Diag}\left(\sum_{k\in\mathcal{S}_i}\left(\mathbf{x}_k\mathbf{x}_k^\mathsf{T} - \boldsymbol{\mu}_i\boldsymbol{\mu}_i^\mathsf{T}\right)\right) + \boldsymbol{\epsilon}}, \tag{5}$$

where $\mathcal{S}_i$ is the set of 3-tuples from which the statistics are computed and m is the size of $\mathcal{S}_i$, and $\boldsymbol{\epsilon}$ is a constant term for numerical stability. In case of batch normalization, the set $\mathcal{S}_i$ is

$$\mathcal{S}_i = \{k | k_C = i_C\}, \tag{6}$$

where k_C is a running index iterating over the channel axis [17]. Eq. 6 implies that normalization is performed across the batch and time axes at channel i_C. In the case of instance norm, the set is

$$\mathcal{S}_i = \{k | k_N = i_N, k_C = i_C\}, \tag{7}$$

where k_N is a running index along the batch axis. Eq. 7 implies that normalization is performed across the temporal axis for each batch and channel and that the batch and channel axes are normalized independently.

2.4. Loss Function

Margin-based loss has been very successful in face recognition and speaker recognition [14]. Additive-margin loss enforces a minimum margin m between the target class and non-target classes:

$$\begin{aligned}\mathcal{L}_{AMS} &= -\frac{1}{n}\sum_{i=1}^{n}\log\frac{e^{s\cdot\left(\cos\theta_{y_i}-m\right)}}{e^{s\cdot\left(\cos\theta_{y_i}-m\right)}+\sum_{j=1,j\neq y_i}^{c}e^{s\cdot\cos\theta_j}} \\ &= -\frac{1}{n}\sum_{i=1}^{n}\log\frac{e^{s\cdot\left(\mathbf{W}_{y_i}^\mathsf{T}\mathbf{x}_i-m\right)}}{e^{s\cdot\left(\mathbf{W}_{y_i}^\mathsf{T}\mathbf{x}_i-m\right)}+\sum_{j=1,j\neq y_i}^{c}e^{s\mathbf{W}_j^\mathsf{T}\mathbf{x}_i}},\end{aligned} \tag{8}$$

where s is a scaling constant, $\mathbf{W}$ is a weight matrix ($\mathbf{W}_j$ is the j-th column of $\mathbf{W}$), and $\mathbf{x}$ is an embedding vector. Both $\mathbf{W}_j$ and $\mathbf{x}_i$ are normalized to have unit length.

3. Experiments

3.1. Data Preparation

The training data include the VoxCeleb1 development set and the VoxCeleb2 development set [5, 18]. We followed the data augmentation strategy in the Kaldi SRE16 recipe. The training data were augmented by adding noise, music, reverb, and babble to the original speech files in the datasets. After filtering out the utterances shorter than 400ms and the speakers with less than 8 utterances, we are left with 7,302 speakers. For the networks that use hand-crafted features as input, we used MFCC and filter-bank features implemented in Kaldi with a frame length of 30ms. The number of mel-scale filters is 40 and the lower and upper cutoff frequencies covered by the triangular filters are 20Hz and 7,600Hz, respectively. Mean normalization was applied to the filter-bank and MFCC features using a 3-second sliding window. Non-speech frames were removed by Kaldi's energy-based voice activity detector.

3.2. DNN Training

The networks were trained using additive-margin softmax with a margin of 0.35 and a scaling factor of 30. The networks were optimized using stochastic gradient descent (SGD). For each mini-batch, we randomly selected 64 utterances from the training set and then randomly cropped a 400ms speech segment from each utterance. We define one epoch as looping through 120,000 segments. We trained the networks for 320 epochs. The learning rate was set to 0.005 and was divided by 10 at Epoch 80, Epoch 120, and Epoch 160. All networks were implemented in PyTorch [19]. After training, we extracted the speaker embedding and used cosine similarity to produce verification scores. No LDA nor PLDA was used.

Table 2: *Performance on the VoxCeleb1 test set. The difference between our x-vector network and Kaldi's is the loss function. We used additive-margin softmax while Kaldi's uses regular softmax. We placed the temporal gating unit after the feature encoder in the wav2spk model.*

Input	Model	EER	minDCF
MFCC	Kaldi's x-vector	3.12	0.325
MFCC	our x-vector	2.54	0.231
Fbank	our x-vector	2.20	0.226
Waveform	wav2spk	1.95	0.203

Table 3: *Effect of temporal gating on the wav2spk network.*

	EER(%)	minDCF
No gating	2.17	0.225
Gating after encoder	1.95	0.203
Gating before stats-pooling	1.97	0.215

3.3. Evaluation

We used the VoxCeleb dataset [20] to evaluate the performance of the proposed wav2spk. VoxCeleb includes VoxCeleb1 and VoxCeleb2. Together, the corpus contains over one million utterances from over 7,000 celebrities extracted from the YouTube videos. We used the whole VoxCeleb2 and VoxCeleb1's training data for training and VoxCeleb1's test data for performance evaluation. The test utterances from VoxCeleb1 were spoken by speakers from a wide range of ethnicities, professions, and ages in a variety of environments, including red carpets, outdoor stadiums, and indoor studios. The recordings contain real-world noise, such as background chatter, laughter, and room acoustics. There are equal numbers of target and impostor trials in the VoxCeleb1 test set. We report results in terms of equal error rate (EER) and minimum cost function (DCF) with P_{target} = 0.01.

4. Results

4.1. Comparing with MFCC and filter-bank Features

In this section, we compare the proposed wav2spk with the x-vector networks trained on MFCC and filter-bank features. When MFCC and filter-bank features were used, VAD and CMVN were applied. For the wav2spk systems, we used the DNN architecture presented in Table 1, with instance norm being applied to every layer in the feature encoder and temporal gating being applied after the encoder. Figure 1 shows the detection error trade-off curves of the three systems and Tables 2 shows the performance of the three systems. We can see that the filter-bank features outperform the MFCC features slightly, while the proposed wav2spk system performs the best among the three systems.

4.2. Effect of the Gating Mechanism

In this section, we investigate the effect of the gating mechanism proposed in Section 2.2. We also investigate two potential places to put the gating unit: after the feature encoder or before the statistics pooling layer. Table 3 shows the performance of the wav2spk system under three configurations: (1) without gating mechanism, (2) using temporal gating after the feature encoder, and (3) using temporal gating before statistics pooling. We can see that using temporal gating indeed improves the performance and placing it after the feature encoder works best.

4.3. Investigation of Fusion Systems

Because the proposed wav2spk system is very different from the x-vector networks trained on MFCC and filter-bank features. We expect that system fusion will improve performance. Table 4 shows the results of the fusion systems. "Our x-vector fbank" refers to the x-vector network with AM-softmax using filter-bank features as input. For the fusion of the same model type, we trained two models of the same type independently. Then, the scores from the two models were linearly combined with equal weights. For the fusion of different model types, two different models were trained using the same data set. Then, their scores were linearly combined with equal weights. The results show that fusion of the x-vector trained on filter-bank features and the wav2spk gives the most significant performance gain, which suggests that they are more complementary than the two identical systems with independent training.

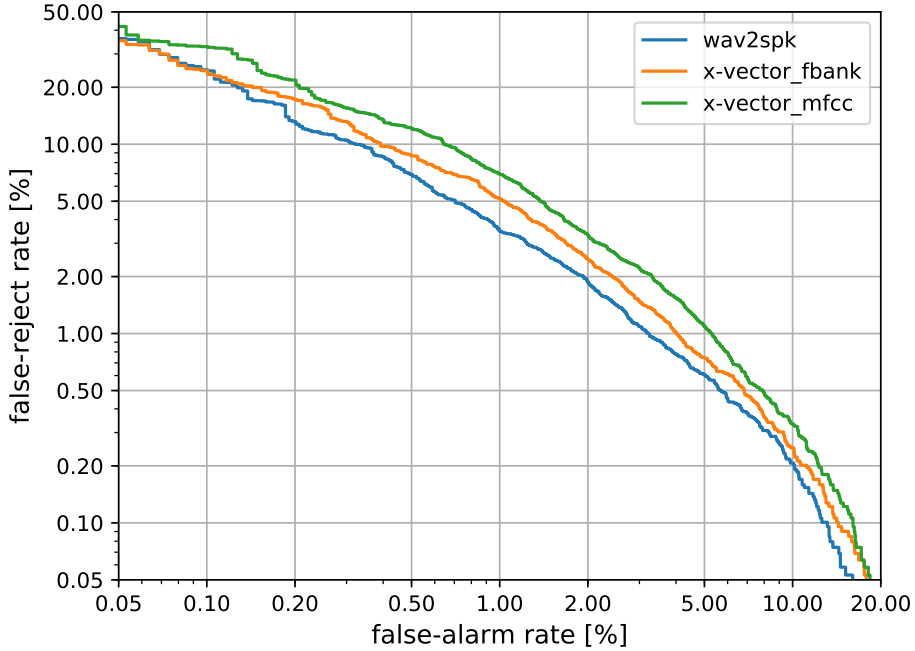

Figure 1: *Detection error tradeoff plots of the proposed wav2spk, the x-vector network trained on MFCC features and the x-vector network trained on filter-bank features.*

Table 4: *Performance of fusion systems. "Our x-vector fbank" refers to the x-vector network trained using AM-softmax on filter-bank features. For the fusion of the same model type, we conducted two independent trainings. The scores of all fusion systems are produced by simple score averaging.*

Model1	Model2	EER(%)	minDCF
Our x-vector fbank	Our x-vector fbank	2.14	0.225
wav2spk	wav2spk	1.81	0.192
Our x-vector fbank	wav2spk	1.55	0.174

5. Conclusions

In this paper, we investigate the possibility of learning deep speaker embeddings directly from waveforms. The results show that with a properly designed architecture, the proposed wav2spk network can directly process waveforms and performs better than the convention x-vector network. There are several crucial components in the wav2spk network that attribute to this superior performance. These components include (1) a feature encoder that converts raw waveforms into frame-level features, (2) an instance norm that performs mean and variance normalization across the time, and (3) a gating mechanism that selects important frames from the feature encoder. In the future, we will investigate using different network architecture to see if the performance of wav2spk can be improved further.

6. Acknowledgements

This work was in part supported by the National Natural Science Foundation of China (NSFC), Grant No. 61971371 and RGC of HKSAR, Grant No. PolyU 152137/17E.

7. References

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[2] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted Gaussian mixture models," *Digital Signal Processing*, vol. 10, no. 1-3, pp. 19–41, 2000.

[3] 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.

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

[5] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," in *Interspeech*, 2018.

[6] 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.

[7] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*. MIT press Cambridge, 2016.

[8] N. Zeghidour, N. Usunier, I. Kokkinos, T. Schaiz, G. Synnaeve, and E. Dupoux, "Learning filterbanks from raw speech for phone recognition," in *Proc. ICASSP*, 2018, pp. 5509–5513.

[9] M. Ravanelli and Y. Bengio, "Speaker recognition from raw waveform with SincNet," in *Proc. 2IEEE Spoken Language Technology Workshop (SLT)*, 2018, pp. 1021–1028.

[10] H. Muckenhirn, M. M. Doss, and S. Marcell, "Towards directly modeling raw speech signal for speaker verification using CNNs," in *Proc. ICASSP*, 2018, pp. 4884–4888.

[11] J.-W. Jung, H.-S. Heo, I.-H. Yang, H.-J. Shim, and H.-J. Yu, "A complete end-to-end speaker verification system using deep neural networks: From raw signals to verification result," in *Proc. ICASSP*, 2018, pp. 5349–5353.

[12] J.-W. Jung, H.-S. Heo, J.-h. Kim, H.-j. Shim, and H.-J. Yu, "RawNet: Advanced end-to-end deep neural network using raw waveforms for text-independent speaker verification," *arXiv preprint arXiv:1904.08104*, 2019.

[13] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," *arXiv preprint arXiv:1904.05862*, 2019.

[14] 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.

[15] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 7132–7141.

[16] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022*, 2016.

[17] Y. Wu and K. He, "Group normalization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 3–19.

[18] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: a large-scale speaker identification dataset," in *Interspeech*, 2017.

[19] 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.

[20] A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman, "Voxceleb: Large-scale speaker verification in the wild," *Computer Speech & Language*, vol. 60, p. 101027, 2020.