END-TO-END ATTENTION BASED TEXT-DEPENDENT SPEAKER VERIFICATION

Shi-Xiong Zhang, Zhuo Chen[†], Yong Zhao, Jinyu Li and Yifan Gong

Microsoft Corporation, One Microsoft Way, Redmond, WA 98052
[†]Columbia University, New York, NY, USA
{zhashi, yonzhao, jinyli, ygong}@microsoft.com, [†]zc2204@columbia.edu

ABSTRACT

A new type of End-to-End system for text-dependent speaker verification is presented in this paper. Previously, using the phonetic discriminate/speaker discriminate DNN as a feature extractor for speaker verification has shown promising results. The extracted frame-level (bottleneck, posterior or d-vector) features are equally weighted and aggregated to compute an utterance-level speaker representation (d-vector or i-vector). In this work we use a speaker discriminate CNN to extract the noise-robust frame-level features. These features are smartly combined to form an utterance-level speaker vector through an attention mechanism. The proposed attention model takes the speaker discriminate information and the phonetic information to learn the weights. The whole system, including the CNN and attention model, is joint optimized using an end-to-end criterion. The training algorithm imitates exactly the evaluation process — directly mapping a test utterance and a few target speaker utterances into a single verification score. The algorithm can smartly select the most similar impostor for each target speaker to train the network. We demonstrated the effectiveness of the proposed end-to-end system on Windows 10 "Hey Cortana" speaker verification task.

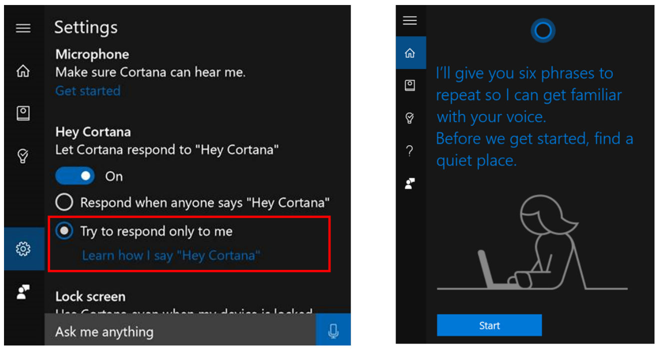
Index Terms— speaker verification, end-to-end training, attention model, deep learning, CNN

1. INTRODUCTION

Speaker verification (SV) is a binary classification problem in which a person's identity is verified based on his/her voice. Speaker verification can be categorized into *text-dependent* and *text-independent* [1]. In text-dependent systems, the same set of phrases are used for enrollment and recognition. In text-independent systems, on the other hand, different phrases are used. The text-dependent SVs usually outperforms the text-independent SVs, because of the constraint of the phonetic variability [2, 3, 4]. Especially with the increasing popularity of mobile/wearable devices, it is competitively beneficial to enable the full voice interaction beginning from a fixed-phrase (keyword) voice-authenticated wake-up [5]. At Microsoft, we are interested in the *text-dependent* speaker verification with the global keyword "Hey Cortana" (see Fig. 1). Mathematically, the relationship between text-dependent/independent SV and Keyword Spotting (KWS) wake-up system can be described in the following equation

$$\overbrace{P(\text{spk}|\mathbf{O}_{1:T})}^{\text{text-independent SV}} = \sum_{\mathbf{w}} \overbrace{P(\text{spk}, \mathbf{w}_{1:L}|\mathbf{O}_{1:T})}^{\text{joint speaker \& KWS wakeup}}$$
$$= \sum_{\mathbf{w}} \underbrace{P(\text{spk}|\mathbf{w}_{1:L}, \mathbf{O}_{1:T})}_{\text{text-dependent SV}} \; \underbrace{P(\mathbf{w}_{1:L}|\mathbf{O}_{1:T})}_{\text{KWS speech recognition}}$$

where $\mathbf{O}_{1:T}$ is the observation sequence and $\mathbf{w}_{1:L}$ is stand for the word/phone/senone sequence. One major advantage of *text-dependent* speaker verification systems is it has knowledge of the utterances phonetic content and can achieve robust verification results with very short enrollment utterances [6].


Settings
Microphone
Make sure Cortana can hear me.
Get started
Hey Cortana
Let Cortana respond to "Hey Cortana"
[✓ On]
Respond when anyone says "Hey Cortana"
[✓ Try to respond only to me]
Learn how I say "Hey Cortana"
Lock screen
Ask me anything

I'll give you six phrases to repeat so I can get familiar with your voice.
Before we get started, find a quiet place.
Start

Fig. 1. *The keyword spotting and speaker verification (with the fixed phase "Hey Cortana") in Windows 10.*

Previously, the traditional techniques [7, 8] used for *text-independent* speaker verification have been found ineffective for the *text-dependent* tasks [4]. Better performance can be achieved by slightly modifying the older techniques such as GMM-UBM[4], GMM-SVM[9, 10] and i-vector/PLDA[11]. More recently, the deep neural networks (DNNs) [12] and recurrent neural networks (RNNs) [13] have been successfully applied to *text-dependent* speaker verification. Two types of DNNs, speaker discriminant DNNs [6] and phonetic discriminant DNNs [14], have been investigated to extract the frame-level features, such as d-vectors [6], bottleneck features [14, 15] and phonetic posterior features (alignments) [12, 15]. These features are treated equally important and then aggregated together to compute an utterance-level speaker representation, such as i-vectors [12, 15], d-vectors [5, 13]. These utterance-level features from the test speaker and enrolled speaker are then scored using a similarity measure, e.g., cosine distance/PLDA [16, 17]. The DNNs/RNNs used to extract phonetic/speaker features are fixed after the training stage. Google recently proposed an End-to-End method to train the DNNs/RNNs for text-dependent speaker verification [5]. This is in contrast to the established approach of training DNNs to discriminate between speakers at the frame-level. In [5] the cosine distance score of two utterance-level representations are passed to a logistic regression layer to produce the final loss $-\log P(\mathsf{accept}/\mathsf{reject})$. The parameters of whole networks are learned by minimizing this end-to-end loss.

In this paper we use speaker discriminate CNNs to extract noise robust frame-level features. This is inspired by recent works in [18, 19] which illustrated a deep convolution neural network (CNN) architecture outperformed LSTMs in many speech recognition tasks. Another contribution of this work is the CNN extracted frame-level features are smartly combined through an attention mechanism to generate an utterance-level speaker representation, in stead of just equally weighting and averaging all the frames [5, 6]. The proposed attention model takes the speaker discriminative information and phonetic information to learn the attention weights. The third contribution is the whole system, including the CNN and attention model, is joint optimized using a novel end-to-end training algorithm. Unlike the Google's end-to-end training [5] which randomly samples the test speaker and the target speaker, our algorithm uses the most competing impostor for each target speaker (in the case of rejection). Finally, the end-to-end system is evaluated on Windows 10 "Hey Cortana" speaker verification task.

2. END-TO-END SPEAKER VERIFICATION

This section describes the overview architecture of our end-to-end speaker verification system. The detail of its important components will be discussed in Section 3 and 4. The experimental results and analysis can be found in Section 5.

2.1. End-to-End architecture

A typical speaker verification protocol includes three phases: training, enrollment, and evaluation. In the training phase, our network learns to extract an internal speaker representation from the utterance. The learning network includes two parts, a CNN and an attention network, as shown in Figure 2. This network learning stage is like the UBM training stage. The details of our CNN model and attention network will be discussed in Section 3 and 4. Note that the networks are not

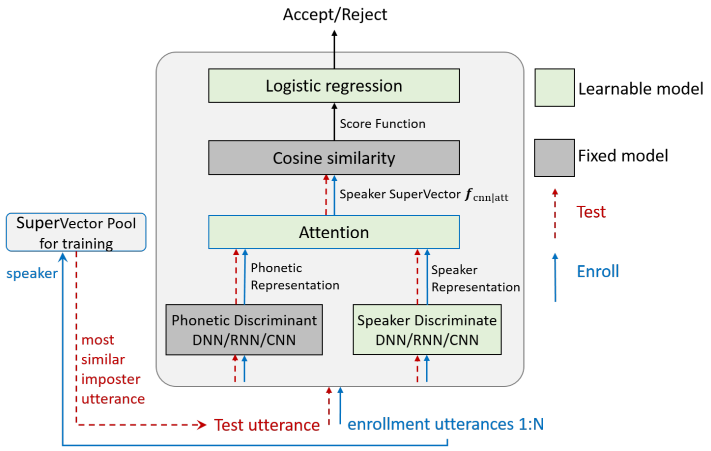

Accept/Reject
Logistic regression
Score Function
Cosine similarity
Speaker SuperVector $f_{\mathrm{cnn|att}}$
Attention
Phonetic Representation
Speaker Representation
Phonetic Discriminant DNN/RNN/CNN
Speaker Discriminate DNN/RNN/CNN
SuperVector Pool for training
speaker
most similar imposter utterance
Test utterance
enrollment utterances 1:N
Learnable model
Fixed model
Test
Enroll

Fig. 2. The architecture of our end-to-end attention based system for speaker verification. The green blocks indicate the models can be learned in the training phase. All the green and grey blocks are fixed in the enrollment and verification phases. During enrollment, the extracted speaker supervectors are stored as a speaker model. The red dash and blue solid arrows indicate the representations of the test utterance and enrolled utterances, respectively. The speaker vector pool contains all the speaker representations in the training set and is used to select the most similar impostor during training.

trained to discriminate between speakers at the frame-level. In stead, all the parameters in the whole system will be jointly trained using an end-to-end criterion described in Section 2.2.

After training, the CNN and attention networks are used to extract the utterance-level speaker vectors (see Fig. 2) for the enrolment and test speakers. This is done by freezing the parameters of the learning network. In the enrollment phase, each speaker provides a few utterances and each utterance is converted to a supervector through the trained network. The speaker model is obtained by averaging over a small mount of enrollment suptervectors.

In the evaluation phase, a scoring function $S(\mathbf{O}_{\mathsf{tst}}, \mathbf{O}_{\mathsf{spk}}^{1:\mathsf{N}})$ is used to compute the similarity between a test utterance $\mathbf{O}_{\mathsf{tst}}$ and a claimed speaker spk. The final score will be compared against a pre-defined (may be speaker dependent) threshold. The system will accept if the score exceeds the threshold, i.e., the utterance $\mathbf{O}_{\mathsf{tst}}$ belongs to spk, and reject otherwise. Typically, a simple scoring function is the cosine similarity [16] between the test speaker representation $\boldsymbol{f}_{\mathsf{cnn|att}}(\mathbf{O}_{\mathsf{tst}})$ and the speaker model $\boldsymbol{f}_{\mathsf{cnn|att}}(\mathbf{O}_{\mathsf{spk}}^{1:\mathsf{N}})$,

$$S(\mathbf{O}_{\mathsf{tst}}, \mathbf{O}_{\mathsf{spk}}^{1:\mathsf{N}}) = \frac{\boldsymbol{f}_{\mathsf{cnn|att}}(\mathbf{O}_{\mathsf{tst}})^{\mathsf{T}} \boldsymbol{f}_{\mathsf{cnn|att}}(\mathbf{O}_{\mathsf{spk}}^{1:\mathsf{N}})}{\|\boldsymbol{f}_{\mathsf{cnn|att}}(\mathbf{O}_{\mathsf{tst}})\| \, \|\boldsymbol{f}_{\mathsf{cnn|att}}(\mathbf{O}_{\mathsf{spk}}^{1:\mathsf{N}})\|} \quad (1)$$

where the speaker model $\boldsymbol{f}_{\mathsf{cnn|att}}(\mathbf{O}_{\mathsf{spk}}^{1:\mathsf{N}})$ is precomputed in the enrollment stage and $\boldsymbol{f}_{\mathsf{cnn|att}}(\cdot)$ indicates that the feature mapping depends on the CNN and attention networks. In [5] the similarity score S is then passed to a logistic regression $\sigma(\cdot)$ which including a linear layer with a bias. Thus,

$$P(\mathsf{accept}|\mathbf{O}_{\mathsf{tst}}, \mathbf{O}^{\mathsf{1:N}}_{\mathsf{spk}}) = \sigma(S) = \frac{1}{1 + e^{-wS(\mathbf{O}_{\mathsf{tst}}, \mathbf{O}^{\mathsf{1:N}}_{\mathsf{spk}}) - b}} \tag{2}$$

where scalar w is a score normalizer trained using the criterion described in the next section, $-b/w$ can be viewed as a verification threshold. The reject probablity can be computed by $P(\mathsf{reject}|\mathbf{O}_{\mathsf{tst}}, \mathbf{O}^{\mathsf{1:N}}_{\mathsf{spk}}) = 1 - \sigma(S)$.

2.2. End-to-End training

Given the current CNN and the attention model's parameters, $\mathbf{W}_{\mathsf{cnn|att}}$, the similarity score $S(\mathbf{O}_{\mathsf{tst}}, \mathbf{O}^{\mathsf{1:N}}_{\mathsf{spk}})$ between the test utterance and the target speaker utterances can be computed and fed to the logistic regression $\sigma(\cdot)$. All the parameters in the whole network (see architecture in Fig. 2) can be jointly optimized using the following end-to-end criterion [5]

$$\mathcal{F}(\mathbf{W}_{\mathsf{cnn|att}}) = -\frac{1}{I}\sum_{i=1}^{I} y_i \log \sigma(x_i) + (1-y_i)\log(1-\sigma(x_i)) \tag{3}$$

where $\mathbf{W}_{\mathsf{cnn|att}}$ represents the parameters of CNN, attention model and logistic regression $\sigma(\cdot)$ and $\mathcal{F}(\cdot)$ is a function of $\mathbf{W}_{\mathsf{cnn|att}}$. The input $x_i = S_i(\mathbf{O}_{\mathsf{tst}}, \mathbf{O}_{\mathsf{spk}})$ is a similarity score of the i-th pair of $(\mathsf{tst}, \mathsf{spk})$, which depends on the $\mathbf{W}_{\mathsf{cnn|att}}$. The label $y_i = 1$ if the testing utterance $\mathbf{O}_{\mathsf{tst}}$ belongs to spk, otherwise $y_i = 0$. I is the total number of $(\mathsf{tst}, \mathsf{spk})$ pairs in the training set (which is huge). To effectively using the data, an algorithm that can smartly pick the most "confusing" pairs is proposed. The detail of the approach is described in Alg. 1. Note the training process with the criterion in Eq. 3 is actually imitating the end-to-end evaluation metric. It directly maps a test utterance and a few target utterances to a single score for verification. The whole network is trained by optimizing the end-to-end loss.

To make sure the parameters are updated using sufficient information from diversified speakers, we group the speakers into mini-batches. Each mini-batch contains 64 speakers as targets. For each target speaker, we sample N utterances as enrollment and T_1 test utterances as "acceptance" data. For each target speaker, we also select k most similar speakers as impostors to make sure the network can learn to discriminate between the most challenging samples. For these k impostors we randomly sample T_2 utterances as "reject" data. The exact number of these hyperparameters, N, T_1, T_2 and k, are discussed in Section 5.2. In order to efficiently search the k most similar impostors, a query table is build to the store the k nearest neighbor for each speaker. The table is built using the all-nearest-neighbors algorithm [20] with the cosine-distance metric in $\mathcal{O}(k \cdot n \log n)$ time. The speaker vector pool used to compute the table is initialized by i-vectors [21]. The pool can be updated periodically (each full sweep) using the speaker supervectors, $\boldsymbol{f}_{\mathsf{cnn|att}}(\mathbf{O}^{\mathsf{1:N}}_{\mathsf{spk}})$.

Algorithm 1: End-to-End Training for SV

```
Data: 10k speakers, each speaker has 10-50 utterances.
Result: Speaker discriminant CNN, attention network
        and logistic regression model.
initial $\mathbf{W}_{\mathsf{cnn|att}}$
initial speaker vector pool $\mathcal{V} \leftarrow$ i-Vectors
for each full sweep do
    for each minibatch in a full sweep do
        for each spk in a minibatch do
            1) sample $N + T_1$ utterances of spk
               $N$ enrollment: $\mathbf{O}^{\mathsf{1:N}}_{\mathsf{spk}}$
               $T_1$ test: $x_i = S_i(\mathbf{O}^i_{\mathsf{tst}}, \mathbf{O}^{\mathsf{1:N}}_{\mathsf{spk}}), y_i = 1|_{i=1}^{T_1}$
            2) search $k$ most similar impostors in $\mathcal{V}$
            3) sample $T_2$ utterances belongs to these imp
               $T_2$ test: $x_i = S_i(\mathbf{O}^i_{\mathsf{imp}}, \mathbf{O}^{\mathsf{1:N}}_{\mathsf{spk}}), y_i = 0|_{i=1}^{T_2}$
            4) gather $(x_i, y_i)$, accumulate $\nabla\mathcal{F}(\mathbf{W}_{\mathsf{cnn|att}})$
        update $\mathbf{W}_{\mathsf{cnn|att}} \leftarrow \mathbf{W}_{\mathsf{cnn|att}} - \eta\nabla\mathcal{F}(\mathbf{W}_{\mathsf{cnn|att}})$
    update $\mathcal{V} \leftarrow \boldsymbol{f}_{\mathsf{cnn|att}}(\mathbf{O}^{\mathsf{1:N}}_{\mathsf{spk}})\ \forall$ spk
```

3. NEURAL NETS FOR SPEAKER VERIFICATION

This section presents a survey of DNN/RNN based methods for speaker verification. A new speaker-discriminate CNN model is then proposed and applied to the end-to-end system described in previous section.

DNNs/RNNs have been successfully integrated into the speaker verification paradigms. The existing neural networks for speaker verification can be categorized into two types — *phonetic discriminant* DNNs and *speaker discriminant* DNNs [13]. These two types of DNNs will discussed in Section 3.1 and 3.2.

3.1. Learning Phonetic Representation

Previous research has demonstrated that using the phonetic information from the DNNs yields significant gains for SV [22, 15]. These *phonetic discriminant* DNNs refer to the neural networks that can classify each frame of speech as a specific phoneme/senone. Basically, they are DNNs trained for speech recognition [23] or keyword spotting [24]. The gain of this approach is mainly due to the phonetic representation extracted from these DNNs can help to align (or normalize) different speakers into the same phonetic space to compare.

3.1.1. Phonetic Bottleneck Features $\boldsymbol{b}_t$

The most successful and simple approach of using *phonetic discriminant* DNNs for speaker verification is the so-called bottleneck feature approach [25]. These DNNs are learned in the supervised manner to classify the phoneme/senone labels. Once the network is trained, the bottleneck features $\boldsymbol{b}_t$ can be extracted for every frame, from the outputs of the last hidden

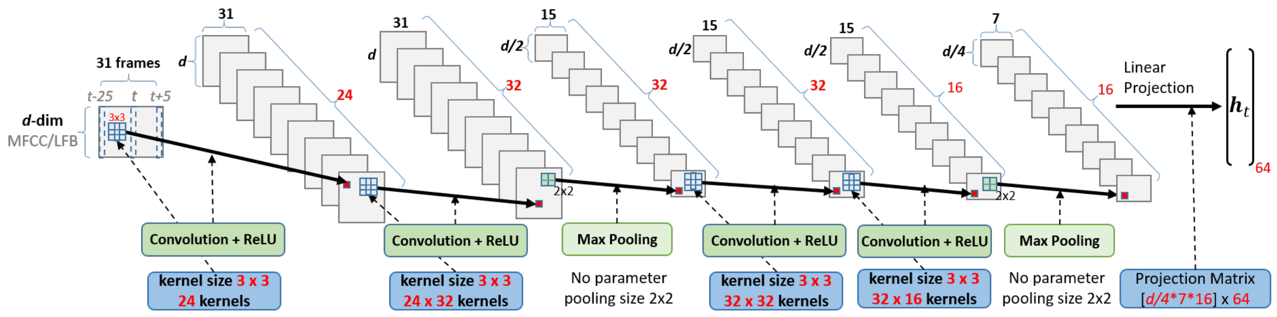


Fig. 3. The architecture of the CNN model for exacting the discriminate speaker representations. The green blocks indicate the operations in the network, such as time-frequency convolutions, ReLU activation function, max pooling and full-connected linear projection. The blue blocks show the number of parameters in each operation. The batch normalization is applied.

layer of the DNN before the sigmoid nonlinearity [26]. The bottleneck features can also be incorporated together with the input MFCC features as Tandem features [27]. These features are then used to train a back-end classifier like a GMM-UBM [22] or i-vector/PLDA [15].

3.1.2. *Phonetic Posterior Features* γ_t

Another approach that makes use of a *phonetic discriminant* DNN for speaker verification is the phonetic posterior feature approach. The basic idea is to replace the frame alignment posteriors γ_t^{UBM} generated by the UBM with the senone posteriors γ_t^{DNN} produced by the DNN [12]. These posteriors γ_t can be directly used to compute the first-order and second-order statistics for i-vector extraction [12]. In this work the phonetic posteriors features γ_t^{DNN} is also used to provide context information in the attention model described in Section 4.

3.2. Learning Speaker Representation

Alternative to the *phonetic discriminant* DNNs, this section discusses the *speaker discriminant* DNNs, which represents a more natural configuration for speaker verification. A *speaker discriminant* DNN is a neural network trained to discriminate between speakers. This type of neural networks has been successfully trained to extract speaker information, such as speaker articulatory features [28] and d-vectors [6].

3.2.1. *dnn-vectors*

In this approach DNNs are trained to discriminate between speakers at the frame-level. Basically, the model tries to classify each frame as belonging to 1-of-N speakers, where N is the number of background speakers [6]. After training each frame of an utterance is forward propagated through the network, and the output of the last hidden layer is used to produce an frame-level speaker representation. All the frame-level features are then averaged to form an utterance-level speaker supervector called a d-vector. The main drawback of d-vectors is that the speaker discrimination is achieved based on a time-scale at which phonetic variability is dominant.

3.2.2. *rnn-vectors*

Previously, a RNN based approach for text-dependent speaker verification that makes use of utterance-level features has been proposed [5, 13]. This approach addresses the main criticism of the d-vector approach, i.e., the frame-level speaker classification. In speaker discriminate RNNs, there is a single label associated with each utterance, instead of one for every frame. The hidden vector (after the activation function) of the RNNs in the last frame, $\boldsymbol{h}_T$, is an effective summary of the entire utterance. One draw back of this utterance-level $\boldsymbol{h}_T$ is it can only be used for text-dependent speaker verification.

3.2.3. *cnn-vectors*

Recently, deep CNNs with small kernels have been shown to achieve a better performance than LSTMs in many speech recognition tasks [18, 19]. Inspired by these works, a deep CNN architecture is proposed here for speaker verification to extract the speaker discriminant information. The rnn-vector approach described in Section 3.2.2 can only be applied to text-dependent speaker verification, while the proposed CNN is suitable for both text-dependent and text-independent tasks.

The CNNs are neural networks with special structures. Fig. 3 illustrates the proposed deep CNN with the VGG-style architecture [29]. In this CNN the first layer is called a convolution layer, which consists of a number of feature maps. Each neuron in the convolution layer receives input from a local field (e.g. a 3×3 window) representing features of a specific frequency range. These local windows shift across time and frequency. The neurons receive different shifted local field as inputs. Neurons in the convolution layer that belong to the same feature map share the same weights, known as kernels or filters. Thus, the convolution layer yield a convolution of the kernels with the inputs. One specialty of our CNN is an asymmetric context window (30 frames in the history and 5 frames in future) is applied to the inputs to control the latency as shown in Fig. 3. Note the LSTMs actually use the information from $\mathbf{O}_{1:t}$ for each frame t, which can also be viewed as an asymmetric context window. The zero-padding is used during the convolution. The batch normalization [30] is applied to improve the training convergence.

4. ATTENTION MECHANISM

As discussed in Section 2, rather than simply averaging the frame-level speaker representation $\boldsymbol{h}_t$ from the CNN to produce an utterance-level feature, we propose to learn the best combination of $\boldsymbol{h}_t$. The basic idea is to use an attention network to learn the best way to combine the frame-level speaker features by utilizing the phonetic context information.

The approach is motivated by the idea of visual attention in the image captioning problem [31]. In the image captioning, when the model is trying to generate the next word of the caption, that word is usually describing *only a part of the image*. Thus the attention network serves as a selection and combination model. In the context of our problem, when the CNN model is trying to generate a speaker feature vector, that vector is only describing the speaker characteristics in a specific context or a specific phonetic space. Thus the attention network servers as an alignment and combination model.

Generally, an attention model is an approach that takes T arguments $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_T$, and a context $\boldsymbol{c}$. It return a vector $\boldsymbol{f}$ which is supposed to be the summary of the $\boldsymbol{h}_{1:T}$, focusing on information corresponding to the context $\boldsymbol{c}$. More formally, it returns a weighted mean of the $\boldsymbol{h}_t$, and the weights are chosen according the relevance of each $\boldsymbol{h}_t$ given the context c, as shown in Fig. 4 (a).

4.1. attention network with posterior weights

The first proposed attention network simply uses the DNN posterior features as the context information shown in Fig. 4 (b). The posterior probabilities from the KWS-DNN naturally provide the alignment information and combination weights. Thus the speaker supervector $\boldsymbol{f}_{\mathsf{cnn|att}}$ can be computed by

$$\boldsymbol{f}_{\mathsf{cnn|att}} = \sum_{t=1}^{T} \boldsymbol{h}_t \otimes \gamma_t^{\mathsf{DNN}} = \sum_{t=1}^{T} \begin{bmatrix} P(\mathsf{hh}|\boldsymbol{o}_t) \cdot \boldsymbol{h}_t \\ \vdots \\ P(\mathsf{ey}|\boldsymbol{o}_t) \cdot \boldsymbol{h}_t \end{bmatrix}_{640} \tag{4}$$

where $\otimes$ is the Kronecker product [32], $\boldsymbol{h}_t$ is a 64-dim feature extracted from the CNN, and the posterior feature γ_t^{DNN} has 10 dimensions, each represents a probability of phoneme in "Hey Cortana"[1],

$$\boldsymbol{h}_t = \begin{bmatrix} \mathsf{cnn}(\boldsymbol{o}_t) \end{bmatrix}_{64}, \quad \gamma_t^{\mathsf{DNN}} = \begin{bmatrix} P(\mathsf{hh}|\boldsymbol{o}_t) \\ \vdots \\ P(\mathsf{ey}|\boldsymbol{o}_t) \end{bmatrix}_{10}.$$

Note that the γ_t is a sparse vector for each frame. The Eq.4 is basically stacking the speaker vector $\boldsymbol{h}_t$ to the corresponding phonetic space, with a probability weight. For example, if $P(\mathsf{hh}|\boldsymbol{o}_1) = 1$, then the supervector $\boldsymbol{f}_{\mathsf{cnn|att}}$ will only have the top 64-dim non-zero features. By this way, different speaker vectors will be compared only in the same phonetic space. The frame features are weighted by DNN posteriors, while combination weights can also be learned in attention network.

[1]The Hey-Cortana DNN has nine phonemes $\{\mathsf{hh}, \mathsf{ey}, \mathsf{k}, \mathsf{ao}, \mathsf{r}, \mathsf{t}, \mathsf{aa}, \mathsf{n}, \mathsf{er}\}$ and one garbage state used to absorb the silence, noise and other phonemes.

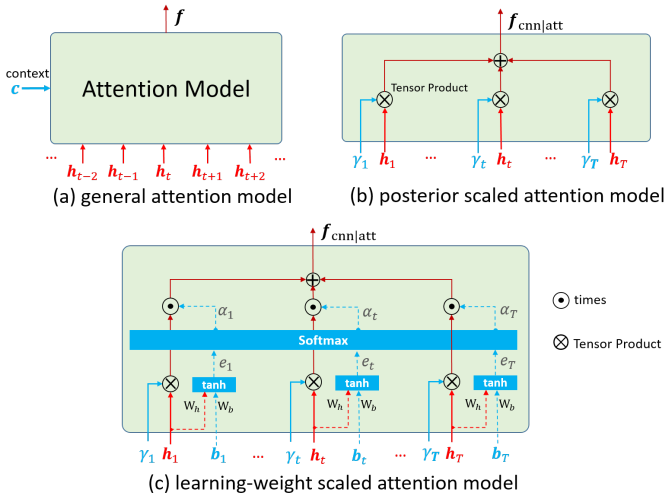

f
context
c
Attention Model
h_{t-2} h_{t-1} h_t h_{t+1} h_{t+2}
(a) general attention model
f_{cnn|att}
Tensor Product
γ_1 h_1 γ_t h_t γ_T h_T
(b) posterior scaled attention model
f_{cnn|att}
α_1 α_t α_T
Softmax
e_1 e_t e_T
tanh
W_h W_b
γ_1 h_1 b_1 γ_t h_t b_t γ_T h_T b_T
(c) learning-weight scaled attention model
times
Tensor Product

Fig. 4. The architecture of the general attention model and two proposed attention networks. The network can learn to align and combine the frame-level speaker representation $\boldsymbol{h}_t$ to form a speaker supervector $\boldsymbol{f}_{\mathsf{cnn|att}}$.

4.2. attention network with learned weights

In this section, an advanced attention network is proposed. It not only utilizes the alignment information but can also learn the combination weights. The architecture of this attention network is shown in Fig. 4 (c). In this model the context information (blue arrows) contains two sources, the posterior features γ_t and the bottleneck feature $\boldsymbol{b}_t$. The sparse vector γ_t is used to mapping the speaker characteristics into the corresponding phonetic space. The bottleneck feature $\boldsymbol{b}_t$ are combined with $\boldsymbol{h}_t$ to learn the combination weights for better speaker discrimination.

$$e_t = \tanh(\boldsymbol{W}_h \boldsymbol{h}_t + \boldsymbol{W}_b \boldsymbol{b}_t) \tag{5}$$

$$\alpha_t = \frac{\exp e_t}{\sum_{j=1}^{T} \exp e_j} \tag{6}$$

$$\boldsymbol{f}_{\mathsf{cnn|att}} = \sum_{t=1}^{T} \left(\alpha_t \cdot \boldsymbol{h}_t \otimes \gamma_t^{\mathsf{DNN}}\right) \tag{7}$$

where the $\otimes\gamma_t^{\mathsf{DNN}}$ serves as an alignment model as discribed in previous section and the e_t serves as a combination model. α_t is the softmax distribution over the input sequence. The combination weights represents the density of speaker information in each frame according to the context. The tensor product and the softmax functions are differentiable, therefore the entire network, including the CNN, attention model, logistic regression can be jointly trained using the stochastic gradient descent.

In this work we applied the attention model to the text-dependent speaker verification. However, as the attention model can learn to map the speaker features into the corresponding phonetic space, the proposed approach would work even better for text-independent task.

5. EXPERIMENTS AND RESULTS

In this section we present the details of network architectures, network training and experimental results. All the models investigated in this work were implemented and trained using Theano [33] and the Keras package [34].

5.1. Data Sets

The proposed approach is evaluated on the Microsoft internal text dependent speaker verification task. A set of utterances beginning with the voice activation phrase "Hey Cortana" is collected from the Windows 10 desktop Cortana service logs. The Hey Cortana segments are taken out using a keyword detector. The length of these segments is around 65 frames to 110 frames. The evaluation comprises about 60k utterances from 3k target speakers and 3k impostors. The enrollment set comprises 6 utterances of Hey Cortana. We then selected about 200k utterances from 10k speakers, each with 10 to 30 utterances for UBM training. There is no overlap of speakers between the training and testing (enrollment/evaluation) data.

5.2. Experiment Setups

The spectral features used in KWS-DNN consist of 38 dimensional mel-frequency cepstral coefficients (MFCC) with 13 filterbanks. The MFCCs contain 12 static (without energy $C0$), 13 delta and 13 delta-delta features. The baseline GMM-UBM and i-Vector systems are using the same MFCCs as KWS-DNNs. A rolling-window based cepstral mean normalization (CMN) and feature warping [35] are applied in these systems. The size of the rolling windows is 41. The CNN model has three channel inputs. The first channel contains 12-dim static features ($d = 12$ in Fig. 3). The second and third channels use the delta and delta-delta features [36], each has 12 dimensions. To control the run-time latency, an asymmetric context window, $\boldsymbol{o}_{t-25}, \ldots, \boldsymbol{o}_{t+5}$, is used (see Fig. 3).

The KWS-DNN consists of two hidden layers. Each layer has 128 and 64 nodes respectively. The output layer has 10 classes. The GMM-UBM system has 128 Gaussian mixtures. The i-vector/PLDA system uses 300 dimensional i-vectors and these are then projected down to 64 dimensions using the LDA [21]. The bottleneck features $\boldsymbol{b}_t$ from the KWS-DNN have 64 dimensions. These bottleneck features are decorrelated using PCA for GMM-UBM systems. The PCA also reduces the dimension of $\boldsymbol{b}_t$ to 32. The CNN model in the end-to-end system uses zero padding and 1-step striding during the convolution operation [18]. A 2×2 pooling window and 2-step striding are used during the max pooling operation. $N = 6$ utterances are selected during the end-to-end network training (see Alg. 1). This is stimulating the real enrollment scenario as Windows 10 will ask users to enroll 6 utterances in order to allow Cortana try to responds only to the user. To avoid mismatch between the negative/positive samples ratio in training and evaluation, $T_1 = 1$ and $T_2 = 5$ are used in the experiments.

5.3. Results

First, we compare the results between different systems. The equal error rates (EERs) are shown in Table 1. The KWS-DNN bottleneck feature is effective for all the systems. The end-to-end system outperforms the GMM and i-vector system by 25% and 9%, respectively. Second, we compare different attention mechanisms and speaker discriminate neural networks (see Table 2). The proposed attention network with learned weights (Section 4.2) works the best. The proposed CNN model also outperforms the DNN (with the same mount of parameters) by 6%. Future work will involve comparing LSTMs with proposed CNN models for text-dependent speaker verification. Third, we show the importance of using speaker vector pool to select the most similar impostor during the end-to-end training (in Table 3).

Features	GMM-UBM	i-vector/PLDA	End-to-End
MFCC $\boldsymbol{o}_t$	5.7%	4.7%	4.3%
MFCC $\boldsymbol{o}_t$+BN $\boldsymbol{b}_t$	4.6%	4.6%	4.3%

Table 1. The results of the GMM-UBM, i-vector/PLDA and End-to-End systems in EER%. The CNN model shown in Fig. 3 and the learning-weight scaled attention network shown in Fig. 4(c) are used in End-to-End system.

	Speaker Modeling		
Attention Mechanism	CNN	DNN	LSTM
posterior scaled attention	4.8%	5.1%	–
learning-weight scaled attention	4.3%	–	–

Table 2. EERs of different attention networks and speaker models (in End-to-End system with MFCC features).

Training Algorithm	best End-to-End system
random sample test speakers	4.7%
using speaker vector pool	4.3%

Table 3. The effectiveness of speaker vector pool in Alg. 1.

6. CONCLUSION

A novel end-to-end system for text-dependent SV is described in this paper. A CNN is used to extract frame-level speaker features. Another contribution of this work is the CNN extracted features are smartly combined through an attention mechanism to generate an utterance-level speaker representation. The proposed attention model takes the speaker discriminative information and phonetic information to learn the attention weights. The third contribution is the whole system, including the CNN and attention model, is joint learned using an end-to-end training algorithm. The algorithm can select the most similar impostors for each target speaker during the training. We show the effectiveness of the proposed system on Windows 10 "Hey Cortana" speaker verification task.

7. REFERENCES

[1] J. P. Campbell Jr., "Speaker recognition: A tutorial," *Proc. IEEE*, vol. 85, no. 9, pp. 1437–1462, 1997.

[2] R. Auckenthaler, E. Parris, and M. Carey, "Improving a GMM speaker verification system by phonetic weighting," in *Proc. ICASSP*, 1999, pp. 1440–1444.

[3] T. Matsui and S. Furui, "A text-independent speaker recognition method robust against utterance variations," in *Proc. ICASSP*, 1991, pp. 377–380.

[4] Bin Ma Haizhou Li Anthony Larcher, Kong-Aik Lee, "Text-dependent speaker verification: Classifiers, databases and rsr2015.," *Speech Communication*, vol. 60, pp. 56–77, 2014.

[5] Georg Heigold, Ignacio Moreno, Samy Bengio, and Noam M. Shazeer, "End-to-end text-dependent speaker verification," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

[6] Ehsan Variani, Xin Lei, Erik McDermott, Ignacio Lopez Moreno, and Javier Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *ICASSP*. IEEE, 2014, pp. 4052–4056.

[7] D. A. Reynolds, "Speaker identification and verification using Gaussian mixture speaker models," *Speech Communications*, vol. 17, pp. 91–108, 1995.

[8] P. Kenny, P. Boulianne, G. amd Ouellet, and P. Dumouchel, "Joint factor analysis versus eigenchannels in speaker recognition," *IEEE Transactions on Audio,Speech and Language Processing*, vol. 15, no. 4, pp. 1435–1447, May 2007.

[9] Sergey Novoselov, Timur Pekhovsky, Andrey Shulipa, and Alexey Sholokhov, "Text-dependent gmm-jfa system for password based speaker verification," in *ICASSP*. IEEE, 2014, pp. 729–737.

[10] Shi-Xiong Zhang and Man-Wai Mak, "Optimized discriminative kernel for SVM scoring and its application to speaker verification," *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 173–185, 2011.

[11] T Stafylakis, Patrick Kenny, P Ouellet, J Perez, M Kockmann, and Pierre Dumouchel, "Text-dependent speaker recognition using plda with uncertainty propagation," *matrix*, vol. 500, pp. 1, 2013.

[12] Yun Lei, Nicolas Scheffer, Luciana Ferrer, and Mitchell McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," in *ICASSP*. IEEE, 2014, pp. 1695–1699.

[13] Gautam Bhattacharya, Patrick Kenny, Jahangir Alam, and Themos Stafylakis, "Deep neural network based text-dependent speaker verification : Preliminary results," in *Odyssey 2016: The Speaker and Language Recognition Workshop*, Bilbao, Spain, June 21-24 2016, pp. 9–15.

[14] Fred Richardson, Douglas Reynolds, and Najim Dehak, "Deep neural network approaches to speaker and language recognition," *IEEE Signal Processing Letters*, vol. 22, no. 10, pp. 1671–1675, 2015.

[15] Hossein Zeinali, Lukas Burget, Hossein Sameti, Ondrej Glembek, and Oldrich Plchot, "Deep neural networks and hidden markov models in i-vector-based text-dependent speaker verification," in *Odyssey 2016: The Speaker and Language Recognition Workshop*, Bilbao, Spain, June 21-24 2016, pp. 24–30.

[16] Najim Dehak, Reda Dehak, James R Glass, Douglas A Reynolds, and Patrick Kenny, "Cosine similarity scoring without score normalization techniques.," in *Odyssey*, 2010, p. 15.

[17] Patrick Kenny, Themos Stafylakis, Pierre Ouellet, Md Jahangir Alam, and Pierre Dumouchel, "Plda for speaker verification with utterances of arbitrary duration," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 7649–7653.

[18] Tom Sercu and Vaibhava Goel, "Advances in very deep convolutional neural networks for LVCSR," *arXiv preprint arXiv:1604.01792*, 2016.

[19] George Saon, Hong-Kwang J Kuo, Steven Rennie, and Michael Picheny, "The IBM 2015 english conversational telephone speech recognition system," *arXiv preprint arXiv:1505.05899*, 2015.

[20] Pravin M Vaidya, "An O(nlogn) algorithm for the all-nearest-neighbors problem," *Discrete & Computational Geometry*, vol. 4, no. 2, pp. 101–115, 1989.

[21] Najim Dehak, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[22] Fred Richardson, Douglas Reynolds, and Najim Dehak, "A unified deep neural network for speaker and language recognition," *arXiv preprint arXiv:1504.00923*, 2015.

[23] George E Dahl, Dong Yu, Li Deng, and Alex Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.

[24] Guoguo Chen, Carolina Parada, and Georg Heigold, "Small-footprint keyword spotting using deep neural networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 4087–4091.

[25] Tianfan Fu, Yanmin Qian, Yuan Liu, and Kai Yu, "Tandem deep features for text-dependent speaker verification.," in *INTERSPEECH*, 2014, pp. 1327–1331.

[26] Frantisek Grézl, Martin Karafiát, Stanislav Kontár, and Jan Cernocky, "Probabilistic and bottle-neck features for lvcsr of meetings," in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*. IEEE, 2007, vol. 4, pp. IV–757.

[27] Qifeng Zhu, Barry Chen, Nelson Morgan, and Andreas Stolcke, "Tandem connectionist feature extraction for conversational speech recognition," in *International Workshop on Machine Learning for Multimodal Interaction*. Springer, 2004, pp. 223–231.

[28] S. X. Zhang, M. W. Mak, and H. M. Meng, "Speaker verification via high-level feature based phonetic-class pronunciation modeling," *IEEE Trans. on Computers*, vol. 56, no. 9, pp. 1189–1198, 2007.

[29] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[30] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[31] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3156–3164.

[32] Alexander Graham, "Kronecker products and matrix calculus: With applications.," *JOHN WILEY & SONS, INC., 605 THIRD AVE., NEW YORK, NY 10158, 1982, 130*, 1982.

[33] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016.

[34] François Chollet, "Keras," `https://github.com/fchollet/keras`, 2015.

[35] Jason Pelecanos and Sridha Sridharan, "Feature warping for robust speaker verification," in *Interspeech*, 2001.

[36] Ossama Abdel-Hamid, Abdel-Rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on audio, speech, and language processing*, vol. 22, no. 10, pp. 1533–1545, 2014.