Neural Networks Based Text-Independent Speaker Identification with Breath and Ey

Wenbo Zhao
School of Electrical and
Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
Email: wzhao1@andrew.cmu.edu

***Abstract*—In text-independent scenarios and with short-time utterances, identifying speakers is difficult due to rich acoustical variations, lack of contextual information and data. To represent speech features, prior speaker models often use Gaussian Mixture Models. While effective, these models rely on prior data distribution assumptions. Moreover, training these models usually requires large amount of data and iterative procedures. To overcome these problems, this study proposes a neural networks based speaker identification framework. The framework extracts constant-Q spectrogram features from speech signals and models them using convolutional network and Long Short-Term Memory network. The framework is tested by identifying speakers using small amount of breath or 'Ey' sound recordings. Results show that the framework is accurate and converges fast.**

I. Introduction

Speaker recognition is useful in many applications, such as in voice authentication, fraud caller detection, multi-speaker tracking, etc. It usually refers to two different tasks involving discriminating people from their voices, *speaker verification* and *speaker identification*. Speaker verification answers question like "Did this person speak this utterance or not?" by comparing the claimed speaker with a potentially large set of speakers, while speaker identification answers question like "Who spoke this utterance?" for a close-set of known speakers. Speaker recognition can be *text-dependent* or *text-independent*, where the former means the spoken words are predefined, like a phrase, and, on the other hand, the latter means no prior knowledge of the spoken words. In this study, we focus on text-independent speaker identification, and develop a neural network based speaker identification framework using breath and 'Ey' data.

Generally speaking, the speaker identification process involves in three phases: feature extraction, speaker modeling, and decision making. Next, we briefly review these three phases.

A. Review of Speaker Identification Process

Features: The features used for speaker identification can be roughly divided into two classes, the high-level features and the low-level features. The high-level features relate to nonacoustic aspects of speech, e.g., semantics, word choice, accents, etc. Contrarily, the low-level features refer to acoustic aspects of speech, e.g., temporal or spectral characteristics, etc. Although low-level features do not capture the contextual information as high-level features do, they reflect the mechanical aspects of articulatory system. These aspects are unique among individuals. For instance, different vocal tract configurations for two different people directly result in distinct formants in their speech spectrum. Hence, the spectral features can be used to distinguish speakers. In fact, the Mel spectrogram is one of the most commonly used features.

Derived from Mel spectrogram, the Mel Frequency Cepstral Coefficients (MFCC) is a good local representation of a short frame of speech signals for its several advantages. Firstly, it uses Mel (logarithmic) frequency scale. The Mel scale mimics the frequency sensitivity of human ear that has higher resolution at low frequencies and lower resolution at high frequencies. Secondly, MFCC uses triangular filters which are overlapped and evenly spaced along the Mel frequency. This helps to capture the energy at each band and gives a rough approximation of the spectrum envelop. Furthermore, MFCC uses Discrete Cosine Transform (DCT) to decorrelate the log spectrum, and remove high frequency variations. Additionally, MFCC can combine its delta and second delta to capture spectrogram's temporal information. Lastly, subtracting the mean of each MFCC feature can compensates channel variations.

Speaker models: In text-independent case, Gaussian Mixture Models (GMM) is one of the most commonly used speaker models. GMM statistically models the speech features as a finite mixture of Gaussian distributions. It is governed by three factors, the mixture responsibility, the mean, and the variance. In order to infer the model factors, the literature uses Expectation Maximization (EM) method to iteratively maximize the expected log likelihood of the observed speech data over latent variables. GMM is a reasonable model because firstly, it represents each class of acoustic events with an individual Gaussian, and secondly, a linear combination of Gaussian basis functions can smoothly approximate a large family of densities. In practice, the literature first trains a GMM model on a large amount of background speakers, which is referred to as the universal background model (UBM). Then the UBM is adapted to model the enrolled speakers using maximum a posteriori (MAP) estimation.

Decisions: Using the GMM-UBM speaker model, one then can decide speakers for given new utterances. Specifically,

the speaker model yields speaker likelihoods for a given test utterance, and classify this utterance to the speaker that has the largest likelihood.

Discussion: Despite of many of its advantages, our study shows that Mel spectrogram is not a good representation for very short utterance in text-independent speaker identification tasks. The reason is that a good feature should give similar representations for utterances from a same speaker, and dissimilar representations for utterances from different speakers, but Mel spectrogram fails to do so in this problem settings.

On the other hand, the computation of GMM needs to compute MFCC first, which cannot leverage the pitch variations in speech. GMM also relies on prior data distribution assumptions, which may fail when these assumptions do not hold. Additionally, training these models usually requires large amount of data, but in many cases we only have small amount of short phone call recordings. Moreover, GMM has iterative procedures, resulting in inefficiency.

B. Contribution of This Paper

In this study, we propose an alternative approach to address the drawbacks associated with Mel spectrogram feature and GMM model. We consider the task of identifying speakers with breath and 'Ey' sound (as in 'Mayday' as an international radio distress signal) recordings. These recordings are intercepted from phone calls. The contributions are two-folded. (1) We propose to use constant-Q features instead of Mel spectrograms. We observe that constant-Q features are good representations for short utterances in our problem settings because firstly, they promote distinction between speakers, and secondly, they help leverage pitch variations within the same speaker. (2) We propose a neural networks based speaker model. It models constant-Q spectrograms using convolutional network (CNN) and Long Short-Term Memory (LSTM) network. CNN automatically learns speaker's shift-invariant features from constant-Q spectrograms. LSTM captures the temporal information from length-varying input sequences. The CNN-LSTM model directly outputs speaker likelihoods for decision making. An advantage of this speaker model is that it combines feature extraction and speaker modeling into a single network, enabling joint optimization of the speaker-dependent feature extractor and the speaker model. Furthermore, the speaker model is accurate, distribution assumption free, and fast.

C. Organization of This Paper

The remainder of the paper is organized as follows. Section 2 presents the speaker identification framework. Section 3 concludes the paper.

II. The Speaker Identification Framework with Breath and Ey

In this section, we first formulate the speaker identification problem, and then describe the speaker identification framework. At the end, we present and discuss experimental results.

A. Problem Formulation

Consider a collection of N constant-Q features $\mathbb{X} = \{\mathbf{X}_1, \ldots, \mathbf{X}_N\}$, $\mathbf{X}_i \in \mathbb{R}^{Q \times T}$, $i = 1, \ldots, N$, where Q is the number of frequency bins, and T is the number of frames. Denote the collection of corresponding labels as $\mathbb{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$, $\mathbf{y}_i \in \mathbb{R}^C$, $i = 1, \ldots, N$, where C is the number of classes / speakers. If $\mathbf{y}_i$ labels class c, then its the c^{th} entry is 1 and the rest entries are 0. We design a classifier $h : \mathbb{R}^{Q \times T} \to \mathbb{R}^C$ which predicts the probability mass over the C classes given feature $\mathbf{X}$, $\widehat{\mathbf{y}} = h(\mathbf{X}) = \mathbb{P}(\mathbf{y} \mid \mathbf{X})$. Define the loss function

$$\begin{aligned} L(\widehat{\mathbf{y}}, \mathbf{y}) &= D_{\mathrm{KL}}(\mathbf{y} \parallel \widehat{\mathbf{y}}) \\ &= \sum_{j=1}^{C} \mathbf{y}_j \log \frac{\mathbf{y}_j}{\widehat{\mathbf{y}}_j}, \end{aligned} \tag{1}$$

where $D_{\mathrm{KL}}(\mathbf{y} \parallel \widehat{\mathbf{y}})$ is the KL distance between $\mathbf{y}$ and $\widehat{\mathbf{y}}$. It is often used to measure the distance between densities. Now we define the classification risk

$$\begin{aligned} R(h) &= \mathbb{E}_{(\mathbb{X},\mathbb{Y})\sim\mathcal{D}} \left[L(\widehat{\mathbf{y}}, \mathbf{y}) \right] \\ &= \mathbb{E}_{(\mathbb{X},\mathbb{Y})\sim\mathcal{D}} \left[D_{\mathrm{KL}}(\mathbf{y} \parallel \widehat{\mathbf{y}}) \right], \end{aligned} \tag{2}$$

which is the expectation of the loss over the data distribution $\mathcal{D}$. Since the true data distribution is unknown, we consider he empirical risk instead

$$\begin{aligned} \widehat{R}(h) &= \tfrac{1}{N} \sum_{i=1}^{N} D_{\mathrm{KL}}(\mathbf{y} \parallel \widehat{\mathbf{y}}) \\ &= \tfrac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} y_{ij} \log \frac{y_{ij}}{\widehat{y}_{ij}}, \end{aligned} \tag{3}$$

such follows the optimization objective

$$h^* = \operatorname*{argmin}_{h \in \mathcal{H}} \widehat{R}(h), \tag{4}$$

where $\mathcal{H}$ is a class of classifiers. Later in this section we define the form of $\mathcal{H}$ as a multi-layer neural network.

B. Feature Extraction

A speech signal is usually divided into $20 \sim 30$ ms frames with overlap of $1/3 \sim 1/2$ of the frame size. Hence within each frame the signal can be seen as stationary. Consider one frame signal $s[n]$. The constant-Q transform

$$x^{\mathrm{cq}}[k] = \tfrac{1}{N_k} \sum_{n < N_k} s[n] w_{N_k}[n] e^{-j2\pi n Q / N_k}, \ k = 1, \ldots, K \tag{5}$$

where $w[n]$ is a window function, and its length N_k varies with frequency index k, $N_k = Q \frac{f_s}{f_k}$. The symbol f_s is the sampling frequency, and f_k is the k^{th} center frequency in the k^{th} bin. The symbol Q is the ratio of f_k to the bandwidth of the k^{th} filter δ_k^{cq}. The total number of frequency bins K is determined by $K = b \log_2 \frac{f_{\max}}{f_0}$, where $f_{\max}$ is the maximal frequency, f_0 is the minimal frequency, and b is the number of bins

per octave. Let $\mathbf{x}^{\mathrm{cq}} = [x^{\mathrm{cq}}[1], \ldots, x^{\mathrm{cq}}[K]]$. Concatenating $\mathbf{x}^{\mathrm{cq}}$ for different frames into a matrix $\mathbf{X}$ gives us the constant-Q spectrogram for input signal. This is our feature for classifier.

Data augmentation with elastic transform

...

C. Speaker Identification Network

Next, we describe the speaker model, which is composed of a convolutional layer, a LSTM layer, and a fully connected layer. The network takes in feature and predicts its speaker label. See Fig. ? for an overview of the network.

1) Convolutional Network: The convolutional layer performs convolution operations on input image to learn its shift-variant features, which can be used to identify speakers. Specifically, consider an input spectrogram $\mathbf{X} \in \mathbb{R}^{Q \times T}$, the convolutional layer convolutes it with F filters $\mathbf{W}^i \in \mathbb{R}^{R \times C}$, $i = 1, \ldots, F$, where M and N are the filter height and width, respectively. The resulting feature maps $\mathbf{Z}^{\mathrm{conv},i} \in \mathbb{R}^{M' \times N'}$, where $(M', N') = ((Q - M)/S + 1, (T - N)/S + 1)$ and S is the stride size

$$Z^{\mathrm{conv},i}_{j',k'} = \sum_{j=1}^{M} \sum_{k=1}^{N} W^i_{j,k} X_{j'S+j,k'S+k} + b^i, \tag{6}$$

$$j' = 1, \ldots, M',\ k' = 1, \ldots, N', \tag{7}$$

where b^i is the bias term. The feature map is then activated

$$\mathbf{X}^{\mathrm{conv},i} = r(\mathbf{Z}^{\mathrm{conv},i}), \tag{8}$$

where $r(\cdot)$ is the rectifier function $r(x) = \max(0, x)$.

2) LSTM Network: The feature maps from the convolutional layer is fed into a LSTM network to learn temporal information between frames. Specifically, consider a collection of input feature maps $\mathbb{X}^{\mathrm{conv}} = \{\mathbf{X}^{\mathrm{conv},1}, \ldots, \mathbf{X}^{\mathrm{conv},F}\}$, where $\mathbf{X}^{\mathrm{conv},i} \in \mathbb{R}^{M' \times N'}$, $i = 1, \ldots, F$. We rearrange $\mathbb{X}^{\mathrm{conv}}$. For each of the N' time steps, the corresponding feature map slices are stacked together, resulting in a single matrix $\mathbf{X}^{\mathrm{r}} \in \mathbb{R}^{(M'F) \times N'}$. We call it embedding. The the number of time steps N' varies with embeddings, which is well suited for LSTM since it can process varying-length sequences. LSTM consists of a sequence of processing units, see Fig. ?. For a single unit, it has three gates controlling the computation flow and one memory cell selectively storing previous sequential information

$$\begin{aligned}
\text{forget gate: } f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\
\text{input gate: } i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\
\text{candidate input: } a_t &= tanh(W_c x_t + U_c h_{t-1} + b_c) \\
\text{cell state update: } c_t &= f_t c_{t-1} + i_t a_t \\
\text{output gate: } o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\
\text{hidden output: } h_t &= o_t tanh(c_t),
\end{aligned} \tag{9}$$

where x_t is the t^{th} step input, h_t is the t^{th} step output, W, U, b are weights for current input, last step output and bias, respectively, and $\sigma(\cdot)$ is the logistic function $\sigma(x) = \frac{1}{1+e^{-x}}$. The output for the $(t-1)^{\mathrm{th}}$ time step h_{t-1} is back fed to the input of the t^{th} time step.

The output from the LSTM is further fed into a fully connected layer and normalized suing softmax function

$$\widehat{y}_i = \frac{e^{\mathbf{w}_i^T \mathbf{h}}}{\sum_{j=1}^{C} e^{\mathbf{w}_j^T \mathbf{h}}},\ i = 1, \ldots, C. \tag{10}$$

This final output is the multi-class likelihood for each speaker. We then can minimize the risk (3). Substitution (10) into (3) yields

$$\begin{aligned}
\mathbf{w}^* &= \operatorname*{argmin}_{\mathbf{w}} J(\mathbf{w}) \\
&= \operatorname*{argmin}_{\mathbf{w}} \tfrac{1}{N} \sum_{i=1}^{N} (\log \sum_{j=1}^{C} e^{\mathbf{w}_j^T \mathbf{h}} - \mathbf{w}_i^T \mathbf{h}).
\end{aligned} \tag{11}$$

3) Learning and Inference: To train the network, we use backpropagation. The training errors are backpropagated and the network parameters are updated with Adadelta. Let

$$\begin{aligned}
g_t &= \nabla_{\mathbf{w}} J(\mathbf{w}_t) \\
\mathbb{E}[g^2]_t &= \gamma \mathbb{E}[g^2]_{t-1} + (1 - \gamma) g_t^2 \\
RMS[g]_t &= \sqrt{\mathbb{E}[g^2]_t + \epsilon},
\end{aligned} \tag{12}$$

the parameters are updated by

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{RMS[\Delta \mathbf{w}]_{t-1}}{RMS[g]_t} \nabla_{\mathbf{w}} J(\mathbf{w}_t), \tag{13}$$

where γ is tuning paramter, and ϵ is a small rounding number. The training converges when the risk (11) maintains small for both training and testing data.

After training, we can predict the class label $\widehat{c}$ given a new utterance, which is the index of the largest value in $\widehat{\mathbf{y}}$.

D. Experiments, Results and Discussion

Now we describe the experimental settings and discuss the results.

1) Data and Experiments: The breath and 'Ey' data are both intercepted from continuous phone call recordings. The 'Ey' data contains 12942 instances corresponding to 53 speakers. The breath data contains 9376 instances corresponding to 44 speakers. To select the best model from a potentially large set of models represented by different weight sets in the network, we use cross-validation. For each speaker, we select 70% of its utterances as training set, 20% as validation set, and 10% as test set.

In the experiment, we first extract constant-Q features from all data using (5). Next, we train the network shown in Fig. ? by feeding training and validation data into it. We evaluate the performance every 5000 iterations. The performance is evaluated using prediction accuracy in the test data

$$\text{accuracy} = \frac{N_{\text{correct}}}{N_{\text{test}}}. \tag{14}$$

2) Results and Discussion: Table. I shows the speaker identification accuracy for breath and 'Ey' data.

Comparison of Mel spectrogram and constant-Q spectrogram

...

Compared with Mel spectrograms, the constant-Q features are more discriminative.

TABLE I
SPEAKER IDENTIFICATION ACCURACY

data	Breath	Ey
Accuracy	86%	85%

III. CONCLUSIONS

In this study, we propose a neural network based framework to identify speakers in text-independent scenarios with short breath and 'Ey' data. This approach addresses the drawbacks associated with Mel spectrogram feature and GMM model. We use constant-Q features to promote distinction between speakers and leverage pitch variations within the same speaker. We propose a CNN-LSTM network to learn input's shift-invariant features and temporal information. It also combines feature extraction and speaker modeling into a single network, enabling joint optimization of the speaker-dependent feature extractor and the speaker model. Compared with GMM speaker model, our proposed CNN-LSTM model is accurate, distribution assumption free, small data effective and fast.