Improving Bottleneck Features for Automatic Speech Recognition using Gammatone-based Cochleagram and Sparsity Regularization

Chao Ma[1,2,3]*, Jun Qi*[4]*, Dongmei Li*[1,2,3]*, Runsheng Liu*[1,2,3]

1. Department of Electronic Engineering, Tsinghua University, Beijing, China, 100084
2. School of Information Science and Technology, Tsinghua University, Beijing, China, 100084
3. Research Institute of Tsinghua University in Shenzhen, 518055
4. Electrical Engineering, University of Washington, Seattle, USA, 98195

c-ma13@mails.tsinghua.edu.cn, qij13@u.washington.edu

{lidmei, lrs-dee}@tsinghua.edu.cn

Abstract

Bottleneck (BN) features, particularly based on deep structures of a neural network, have been successfully applied to Automatic Speech Recognition (ASR) tasks. This paper goes on the study of improving the BN features for ASR tasks by employing two different methods: (1) a Cochleagram generated by Gammatone filters as the input feature for a deep neural network; (2) imposing the sparsity regularization on the bottleneck layer to control the sparsity level of BN features by constraining the activations of the hidden units to be averagely inactive most of the time. Our experiments on the Wall Street Journal (WSJ) database demonstrate that the two approaches can deliver certain performance gains to BN features for ASR tasks. In addition, further experiments on the WSJ database from different noise levels show that the Cochleagram as input has better noise-robust performance than the commonly used Mel-scaled filterbank.

Index Terms: bottleneck feature, Cochleagram, sparsity regularization, deep neural network

1. Introduction

Bottleneck (BN) features are generated from a neural network, typically multiple layer perceptron (MLP) or deep neural network (DNN). The neural network is designed such that the input corresponds to certain primary features and the output corresponds to the sequential labels (e.g., a sequence of context-dependent triphone states). The BN feature associated with a latent pattern of an input feature is obtained from a shallow layer of a neural network and can be regarded as a high-level feature for automatic speech recognition (ASR) [1, 2, 3]. A multitude of research has demonstrated that the BN feature can deliver significant improvement for ASR tasks when compared to primary features, such as Mel-frequency cepstral coefficient (MFCC), perceptual linear predictive (PLP) and Gammatone frequency cepstral coefficient (GFCC) [4, 5]. Furthermore, recent research has shown that the BN feature, when generated based on a deep neural network, is even more promising [6, 7, 8].

A commonly used neural network input is the filterbank generated by Mel-scaled filters on the spectrogram, which has shown a better ASR performance than other primary features [6]. In this paper, we propose a novel cochleagram-based input for a neural network. The production of a Cochleagram is based on a set of Gammatone filters which simulate the frequency response of human ears. As has been shown in our previous work [4], the GFCC is generated from applying a Discrete Cosine Transformation (DCT) on each column of a Cochleagram because the dimensions of the Cochleagram are highly correlated. Since the deepest layer of the DNN has the capability of learning discriminative representations with respect to the inputs, the Cochleagram is assumed to be directly applied. One exploration of this paper is to study whether the Cochleagram as input for the DNN is able to bring in further improvement compared to the Mel-scaled filterbank both in clear and noisy conditions.

In addition, a problem associated with the BN feature is that the feature vectors tend to be highly sparse, i.e., most of the probability mass concentrates on one or a few dimensions, which leads to high correlations among feature dimensions. Considering the sparsity of the BN feature, our previous work proposed a subspace Gaussian Mixture Model (SGMM) for the BN feature [9]. On the top of the SGMM, this paper proposes a sparsity regularization on the BN layer by adding an extra sparsity penalty term to the cost function of the neural network. Sparsity regularization tries to control the sparsity level of the BN feature by constraining the BN units to be averagely inactive most of the time.

The rest of the paper is organized as follows: Section 2 presents the implementation of the BN feature, Section 3 introduces the sparsity regularization approach and Section 4 presents an implementation of Gammatone filters for the generation of the Cochleagram. Experiments are reported in Section 5 and the paper is concluded in Section 6.

2. Bottleneck feature

The BN feature is derived from the bottleneck layer of an MLP/DNN structure. A typical bottleneck structure is illustrated in Figure 1. There are 7 layers in total and 5 of them are hidden. The units at the input layer (at the bottom) correspond to a long-context feature vector that is generated by concatenating 15 consecutive frames of the primary feature followed by a DCT. The dimension of the primary feature is 24, and hence the context window involves 360 primary feature elements. After the DCT, the dimension is reduced to 128, which corresponds to the number of units at the input layer of the MLP.

The output layer (at the top) consists of 3560 units, corresponding to the clustered triphone states. The five hidden layers are constructed following a 1024-1024-39-1024-1024 configu-

ration, where the 39-unit layer (in the middle) is the 'bottleneck layer', and the activations of the units yield the BN feature. The configuration is selected to optimize the ASR performance on a development set.

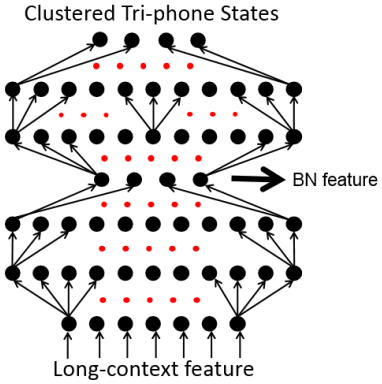

Figure 1: *Bottleneck feature extraction.*

The bottleneck structure shown in Figure 1 consists of a number of hidden layers and hence is regarded as 'deep'. A layer-by-layer growing approach can be employed to train such a network. In this work, we employ a popular pre-training approach based on the restricted Boltzmann machine (RBM) [8].

Once the MLP structure is obtained from the RBM pre-training, conventional back-propagation [10] is employed to learn the parameters in a supervised and discriminative way. The cost function of the MLP is given as (1):

$$J(\mathbf{W},\mathbf{b}) = \frac{1}{N}\sum_{i=1}^{N}(||h_{\mathbf{W},\mathbf{b}}(\mathbf{x}^{(i)})-\mathbf{y}^{(i)}||_2^2)+\lambda \sum_{l=1,l\neq l_{BN}}^{L-1} ||\mathbf{W}^{(l)}||_F^2, \quad (1)$$

where parameters W, b represent the MLP weights and bias respectively, symbols $||\cdot||_2^2$ and $||\cdot||_F^2$ correspond to vector $l2$-norm and matrix Frobenius-norm respectively, h is a sigmoid function, N is the size of the mini-batch data, and λ represents a regularization constant for weight decay. In addition, $x^{(i)}$ represents the i-th data of a mini-batch, $y^{(i)}$ is its corresponding label, L is the number of layers, and L_{BN} represents the BN layer index.

A feed-forward pass is performed before the back-propagation process. The vectors $\mathbf{a}^{(i)}$ and $\mathbf{z}^{(i)}$, which correspond to activations of layer i and its sigmoid function respectively, should be computed iteratively as (2) and (3):

$$\mathbf{a}^{(i)} = h(\mathbf{z}^{(i-1)}) = \frac{1}{1+exp(\mathbf{z}^{(i-1)})} \quad (2)$$

$$\mathbf{z}^{(i)} = \mathbf{W}^{(i)}\mathbf{a}^{(i)} + \mathbf{b}^{(i)}, \quad (3)$$

where $\mathbf{b}^{(i)}$ corresponds to the bias vector of layer i. The back-propagation algorithm is presented in detail in Table 1. When performing the parameter update in back-propagation, the learning rate α is set in an auto-regularized manner: It is set to 4.0 at the first several iterations, and then is decreased by a factor of 2 at each of the remaining iterations. The maximum number of iterations is set to 20, and the convergence is tested on a development set.

Back-propagation algorithm
($\boldsymbol{\delta}$, $\triangle_{\mathbf{W}^{(l)}}$, $\triangle_{\mathbf{b}^{(l)}}$ are temporary variables, l_{max} is the top layer index)
(1). As to the top layer index, set
$\boldsymbol{\delta}^{(l_{max})} = (\mathbf{a}^{(l_{max})} - \mathbf{y})$
(2). For the hidden layer index $l = l_{max}-1, ..., 2$, set
$\boldsymbol{\delta}^{(l)} = (\mathbf{W}^{(l)})^T\boldsymbol{\delta}^{(l+1)} \circ h'(\mathbf{z}^{(l)})$
For $l = 1, 2, ..., l_{max}-1$,
(3). Accumulate the statistics for gradient descent
$\triangle_{\mathbf{W}^{(l)}} = \triangle_{\mathbf{W}^{(l)}} + \boldsymbol{\delta}^{(l+1)}(\mathbf{a}^{(l)})^T$
$\triangle_{\mathbf{b}^{(l)}} = \triangle_{\mathbf{b}^{(l)}} + \boldsymbol{\delta}^{(l+1)}$
(4). Obtain the regularized gradient for updating an MLP
$\frac{\partial}{\partial \mathbf{W}^{(l)}}J(\mathbf{W},\mathbf{b}) = \frac{1}{N}\triangle_{\mathbf{W}^{(l)}} + \lambda\mathbf{W}^{(l)}$
$\frac{\partial}{\partial \mathbf{b}^{(l)}}J(\mathbf{W},\mathbf{b}) = \frac{1}{N}\triangle_{\mathbf{b}^{(l)}}$
(5). Update of parameters
$\mathbf{W}^{(l)} = \mathbf{W}^{(l)} - \alpha\frac{\partial}{\partial \mathbf{W}^{(l)}}J(\mathbf{W},\mathbf{b})$
$\mathbf{b}^{(l)} = \mathbf{b}^{(l)} - \alpha\frac{\partial}{\partial \mathbf{b}^{(l)}}J(\mathbf{W},\mathbf{b})$

Table 1: Back-propagation algorithm.

3. Sparsity regularization

A particular property of the bottleneck MLP is that the bottleneck layer can learn some prominent patterns of the input speech signals in the training phase. In prediction, the BN feature represents the coefficients of the input primary feature based on the learned patterns and hence can be taken as a high-level feature for ASR. Since the patterns are discriminative and representative, the BN feature tends to be highly sparse, i.e., most of the mass of the feature concentrates on a few dimensions.

To control the sparsity level of the BN feature, a sparsity regularization is imposed on the BN layer of the MLP by adding an extra sparse penalty term to the original cost function (1). The sparse penalty term is defined based on Kullback-Leibler (KL) divergence as (4) [11]:

$$\sum_{j=1}^{S_{BN}} KL(f||\hat{f}_j) = \sum_{j=1}^{S_{BN}} flog\frac{f}{\hat{f}_j} + (1-f)log\frac{1-f}{1-\hat{f}_j}, \quad (4)$$

where S_{BN} corresponds to the number of units of the BN layer, f represents a constant of sparsity target, typically a small value close to zero, and $\hat{f}_j$ represents the average activation of the BN unit j as shown in (5) and needs to be computed in the feed-forward pass.

$$\hat{f}_j = \frac{1}{N}\sum_{i=1}^{N}\mathbf{a}_j^{(l_{BN})}(x^{(i)}) \quad (5)$$

Therefore, the cost function (1) is modified as (6):

$$\hat{J}_{sparse}(\mathbf{W},\mathbf{b}) = J(\mathbf{W},\mathbf{b}) + \eta\sum_{j=1}^{S_{BN}} KL(f||\hat{f}_j), \quad (6)$$

where η is the sparsity regularization. Besides, for the BN layer, the step (2) shown in Table 1 is modified as (7):

$$\boldsymbol{\delta}^{(l_{BN})} = (\mathbf{W}^{(l_{BN})})^T\boldsymbol{\delta}^{(l_{BN}+1)} \circ h'(\mathbf{z}^{(l_{BN})}) + \eta(-\frac{f}{\hat{f}_j} + \frac{1-f}{1-\hat{f}_j}). \quad (7)$$

For optimal ASR performance based on sparsity regularization on development set, f is set to 0.1, and the regularization constants λ and η are set to 0.002 and 0.01 respectively.

4. Cochleagram

The Cochleagram is generated based on a set of Gammatone filters. A Gammatone filter (GF) is formally represented in the form of impulse response in the time domain as follows:

$$g(t) = at^{n-1}e^{-2\pi bt}cos(2\pi f_c t + \theta), \quad (8)$$

where f_c is the central frequency of the filter and θ is the phase which is usually set to 0. The constant a controls the gain and n is the order of the filter which is normally set to be equal or less than 4. Finally, b is the decay factor which is related to f_c.

A set of GFs with different f_c forms a Gammatone filterbank, which can be applied to obtain the signal characteristics at various frequencies, resulting in a temporal-frequency representation similar to the Fast Fourier Transform (FFT)-based short-time spectral analysis. In order to simulate the human auditory behavior, the central frequencies of the filterbank are equally distributed on the Bark scale.

The time-efficient implementation of a Gammatone filter can be realized in the time-domain. As is illustrated in Figure 2, for the channel whose central frequency is f_c, the input signal $x(t)$ is first compensated by a frequency-dependent component $e^{-j2\pi f_c t}$, and then passes a frequency-independent base filter $\hat{G}(z)$. The output of the channel, denoted by $y(t; f_c)$, is finally obtained from the output of $\hat{G}(z)$ followed by a reverse compensation $e^{j2\pi f_c t}$.

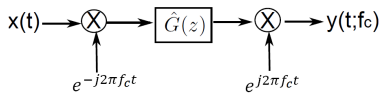

Figure 2: *Time domain Gammatone filtering.*

Considering the special case of $n = 4$, the base filter $\hat{G}(z)$ takes the following form:

$$\hat{G}(z) = \frac{3a}{1 - 4mz^{-1} + 6m^2z^{-2} - 4m^3z^{-3} + m^4z^{-4}}, \quad (9)$$

where $m = e^{-2\pi b/f_s}$ and f_s is the sampling frequency.

A pre-emphasis is implemented to reduce the dynamic range of the spectrum and intensify the low frequency components which normally involve most of the information of the speech signals. The pre-emphasis is designed as a 2-order low-pass filter given by (10).

$$H(z) = 1 + 4e^{-2\pi b/f_s}z^{-1} + e^{-2\pi b/f_s}z^{-2} \quad (10)$$

The generation of the Cochleagram extracts frames from the GF outputs by down-sampling $y(t; f_m)$ to 100 Hz where f_m is the central frequency of the m-th GF. An averaging approach is applied by using a window covering K points and shifting every L points to frame $y(t; f_m)$. For the n-th frame, the average value of $y(t; f_m)$ within the window $t \in [nL, nL + K]$ is computed as the m-th component:

$$\bar{y}(n; m) = \frac{1}{K}\sum_{i=0}^{K-1}\gamma(f_m)|y(nL + i; f_m)|, \quad (11)$$

where $|.|$ represents the magnitude of complex numbers, $\gamma(f_m)$ is a center frequency-dependent factor, and m is the index of the channel whose central frequency is f_m.

The resulting matrix $\bar{y}(n; m)$ provides a frequency-time representation of the original signal and is often referred to as a Cochleagram. A typical Cochleagram is shown in Figure 3 (right), which significantly contrasts to the Mel-scaled filterbank of the same utterance as shown in Figure 3 (left). It is clear that the most amount of energy of the Cochleagram is concentrated on a certain range of frequencies and the discriminative feature pattern of the Cochleagram is assumed to improve the high correlation among dimensions of BN feature. On the other hand, the energy distribution of the Mel-scaled filterbank is more disperse.

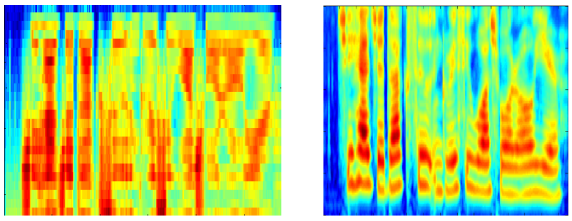

Figure 3: *Spectrogram(left) vs. Cochleagram(right).*

5. Experiments

5.1. Data profile

Our experiments are conducted on the Wall Street Journal (WSJ) database. The dataset for training/development is from the SI-84 subset (7240 utterances). The development set randomly selects 700 utterances from these, and the rest are used for training. For evaluation, data from Dev-93 (504 utterances) and Eval-92 (330 utterances) subsets are used as test data. All the data are reading speech recorded in a noise-free environment. Performance is evaluated on a recognition task involving 5000 words.

To simulate the acoustic mismatch, a multitude of noise signals from the NOISEX-92 database are used to corrupt the test data. These noise signals involve three types: white noise, babble noise and f16 noise. The mixing is conducted at various SNR levels, including 30dB, 20dB and 15dB.

We study two features in the experiments: Mel-scaled filterbank and Cochleagram. Both are used to build the baseline systems. For a fair comparison, all the primary features are derived from the same frequency range (80-5000 Hz) with the same frame rate (100 frames per second). The dimension of the primary features is set to the same. In addition, the root compression is used for the primary features for better noise-robust performance [12]. The generation of long-context features is based on the way that is introduced in Section 2.

The Kaldi toolkit [13] is utilized to train the acoustic models, to build the decoding graph, and to conduct the decoding. The MLP training and Cochleagram extraction are both based on the the authors' released codes.[1]

5.2. System architecture

The entire system architecture is shown in Figure 4, where a conventional Gaussian Mixture Model-Hidden Markov Model

[1]https://github.com/uwjunqi

(GMM-HMM) system is first constructed, and then a universal background model (UBM) and a SGMM are created.

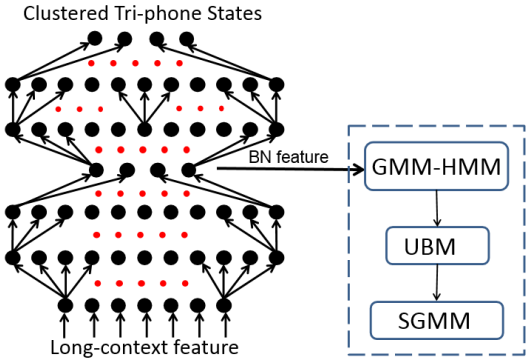

Figure 4: *ASR baseline system.*

We first build the baseline systems which are based on the conventional diagonal GMM. Each context-dependent phone (tri-phone) is modeled as an HMM which consists of three left-to-right non-skipping states. The probability distribution of each state is modeled by a GMM that comprises 8 Gaussian components. The states belonging to the same phone are tied by a decision tree and there are 3560 tied states in total. A tri-gram model with 5000 words in the lexicon is used as the language model in decoding.

In order to construct the SGMM, a UBM was first created by clustering all the Gaussian components of the GMM system. The UBM is composed of 400 Gaussian components in our experiments. The full covariance matrices of the Gaussian components are then trained via an E-M procedure. Afterwards, the SGMM is constructed by copying the UBM followed by an E-M full training [14]. The dimension of each state-specific vector of the SGMMs is set to 40. The final number of sub-states of the SGMMs is 8900 in our experiments.

5.3. Results and analysis

The experiments are firstly conducted with the BN features based on Mel-scaled filterbank and Cochleagram. The performance of BN features with and without sparsity regularization is tested. The results on Dev-93 and Eval-92 are presented in Table 2 and Table 3 respectively, where 'Sparsity Reg.' stands for Sparsity regularization and 'fbank' represents filterbank.

	WER%	
	Mel fbank	Cochleagram
BN+GMM	6.72	6.50
BN+Sparsity Reg.+ GMM	6.61	6.38
BN+SGMM	6.37	6.09
BN+Sparsity Reg.+SGMM	6.25	**5.95**

Table 2: Results of the experiments on Dev-93.

It is clearly seen that Cochleagram brings in an improvement on the ASR task by lowering the word error rate (WER) 4.68% relative, while the experiments with sparsity regularization for BN features demonstrate that an extra 2.74% relative gain can be obtained as well.

The test results on noisy ASR tasks are shown in Figure 5, where ASR results on white, babble and f16 noises are presented. Table 4 lists all WERs in Figure 5. For achieving the best performance, the SGMM and sparsity regularization are applied for all the tests. It is clear that the Cochleagram achieves consistent performance improvements over the counterpart in terms of WERs. Especially in high-level noisy conditions, the Cochleagram is more effective in reducing the WER in ASR tasks.

	WER%	
	Mel fbank	Cochleagram
BN+GMM	3.85	3.73
BN+Sparsity Reg.+ GMM	3.76	3.66
BN+SGMM	3.61	3.49
BN+Sparsity Reg.+SGMM	3.56	**3.42**

Table 3: Results of the experiments on Eval-92.

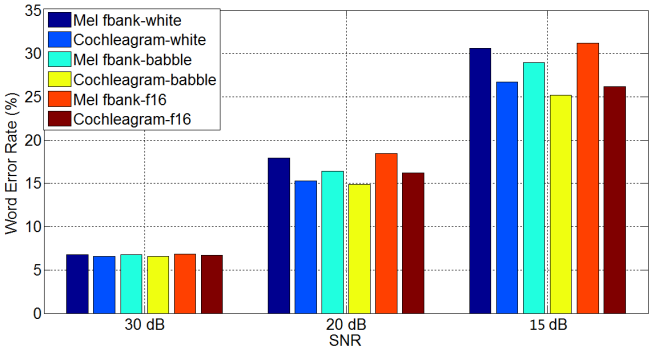

Figure 5: *WERs with three kinds of noises on Dev-93.*

	WER%		
	30dB	20dB	15dB
Mel fbank + white noise	6.79	17.94	30.58
Cochleagram + white noise	**6.60**	**15.29**	**26.74**
Mel fbank + babble noise	6.75	16.38	28.98
Cochleagram + babble noise	**6.54**	**14.86**	**25.21**
Mel fbank + f16 noise	6.82	18.46	31.22
Cochleagram + f16 noise	**6.68**	**16.22**	**26.63**

Table 4: Results of the noise-robust experiments on Dev-93.

6. Conclusions

This paper studies the application of Cochleagram and sparsity regularization on the BN feature. Our experiments on the WSJ database, on the one hand, demonstrate that Cochleagram improves the BN features on ASR tasks compared to the commonly used Mel-scaled filterbank. On the other hand, sparsity regularization also brings in further improvement for BN features on ASR tasks, although the gains are comparatively marginal. Further experiments on noisy data show that the Cochleagram is able to improve the BN feature against noisy effects of all levels.

7. Acknowledgment

This work is funded by Research Institute of Tsinghua University, Shenzhen with the No. JCYJ20140419122040609.

8. References

[1] Q. Zhu, Y. Chen, and N. Morgan, "On using MLP features in LVCSR," in *Proc. Interspeech*, 2004, pp. 921–924.

[2] F. Grezl, M. Karafiat, S. Kontar, and J. Cernocky, "Probabilistic and bottleneck features for LVCSR of meetings," in *Proc. ICASSP*, 2007, pp. 757–760.

[3] J. Frankel, D. Wang, and S. King, "Growing bottleneck features for tandem ASR," in *Proc. Interspeech*, 2008, p. 1549.

[4] J. Qi, D. Wang, Y. Jiang, and R. Liu, "Auditory feature based on gammatone filters for robust speech recognition," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, 2013, pp. 2237–2241.

[5] J. Qi, D. Wang, J. Xu, and J. Tejedor, "Bottleneck features based on gammatone frequency cepstral coefficients," in *Proc. Interspeech*, 2013, pp. 1751–1755.

[6] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, vol. 28, no. 6, pp. 82–97, 2012.

[7] D. Yu and M. L. Seltzer, "Improved bottleneck features using pretrained deep neural networks," in *Proc. Interspeech*, 2011, pp. 237–240.

[8] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[9] J. Qi, D. Wang, and J. Tejedor, "Subspace models for bottleneck features," in *Proc. Interspeech*, 2013, pp. 1746–1750.

[10] M. Christopher, *Pattern Recognition and Machine Learning*. New York, Inc. Secaucus, NJ, USA: Springer, 1998.

[11] A. Ng, "Sparse autoencoder," *Stanford CS294A Lecture notes*.

[12] R. Sarikaya and J. Hansen, "Analysis of the root cepstrum for acoustic modeling and fast decoding in speech recognition," in *Proc. Eurospeech*, 2001.

[13] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The KALDI speech recognition toolkit," in *Proc. of ASRU*, 2011.

[14] D. Povey, "A tutorial introduction to subspace Gaussian mixture models for speech recognition," MSR-TR-2009-11, Microsoft Research, Tech. Rep., 2009.