Bottleneck Features from SNR-Adaptive Denoising Deep Classifier for Speaker Identification

Zhili TAN and Man-Wai MAK
Center for Signal Processing, Dept. of Electronic and Information Engineering
The Hong Kong Polytechnic University, Hong Kong SAR
E-mail: enmwmak@polyu.edu.hk

Abstract—**In this paper, we explore the potential of using deep learning for extracting speaker-dependent features for noise robust speaker identification. More specifically, an SNR-adaptive denoising classifier is constructed by stacking two layers of restricted Boltzmann machines (RBMs) on top of a denoising deep autoencoder, where the top-RBM layer is connected to a soft-max output layer that outputs the posterior probabilities of speakers and the top-RBM layer outputs speaker-dependent bottleneck features. Both the deep autoencoder and RBMs are trained by contrastive divergence, followed by backpropagation fine-tuning. The autoencoder aims to reconstruct the clean spectra of a noisy test utterance using the spectra of the noisy test utterance and its SNR as input. With this denoising capability, the output from the bottleneck layer of the classifier can be considered as a low-dimension representation of denoised utterances. These frame-based bottleneck features are than used to train an iVector extractor and a PLDA model for speaker identification. Experimental results based on a noisy YOHO corpus show that the bottleneck features slightly outperform the conventional MFCC under low SNR conditions and that fusion of the two features lead to further performance gain, suggesting that the two features are complementary with each other.**

Index Terms—**Deep learning; Bottleneck features, denoising autoencoder, speaker identification, deep belief networks**

I. Introduction

In recent years, deep learning has achieved a great success in many areas, including speech recognition [1], computer vision [2], speech synthesis [3], [4] and music recognition [5]. In many of these studies, deep neural networks (DNN) and deep belief networks (DBN) [6] are used as classifiers. This is achieved by adding a softmax layer on top of the hidden layers of restricted Boltzmann machines (RBM). Deep learning is powerful in that the resulting deep networks have strong ability to disentangle the variation in the input patterns, and therefore greatly improve the performance in many classification problems. The posterior probabilities generated by the softmax layer can replace the ones generated by other generative models, e.g. Gaussian mixture models (GMM) in speaker recognition [7], hidden Markov models (HMM) in large vocabulary continuous speech recognition (LVCSR) [1], and the posterior of senones in i-vector based speaker verification [8].

This paper explores the use of DNNs for extracting speaker-dependent features for speaker recognition. To this end, we stacked a denoising deep autoencoder [9], [10], two layers of RBMs and a softmax layer to form a DNN classifier that produces posterior probabilities of speaker identities as output. However, instead of using the classifier directly for speaker identification, we used the RBM just below the softmax output layer of the DNN as the bottleneck layer for feature extraction. More precisely, bottleneck features are extracted from the RBM's outputs before sigmoid nonlinearity. The bottleneck features, which provide a low-dimensional representation of the input patterns [11], are used for training an iVector-PLDA speaker identification system. The advantage of using the DNN as feature extractor rather than using it directly as speaker identifier is that the number of test speakers will not be limited by the number of nodes in the softmax layer.

We used noisy speech as the input and clean speech as the target output to train the denoising autoencoder [10], which is pre-trained by using contrastive divergence [12] followed by backpropagation fine-tuning. Then, two layers of RBMs are trained using the outputs of the denoising autoencoder as input. Finally, a softmax layer with number of nodes equal to the number of training speakers is put on top of the last (bottleneck) layer of RBM and backpropagation fine-tuning is further applied to minimize the cross-entropy training error. Therefore, the first several layers in the DNN classifier help to make the whole neural network more noise robust, while the top layers extract the speaker-dependent information from the denoisd spectra. We demonstrated that at 0dB SNR, the bottleneck features are slightly more robust than the standard mel frequency cepstral coefficient (MFCC) [13].

II. SNR-Adaptive Denoising Deep Autoencoder

A. Input Preprocessing

To train the denoising autoencoder, it is necessary to preprocess the input speech. On one hand, cepstral features have shown promise in previous research; on the other hand, it is intuitive to apply raw features as input to realize the potential of autoencoders in modelling speech signals. In particular, the log-spectra, the log mel-scale triangular filterbank output and even MFCC are candidate inputs.

For the log-spectra, we performed 512-point fast Fourier transform on 8 kHz speech data, followed by taking logarithm. Due to the symmetry property of Fourier transform for real numbers, only the first 256 components were used in subsequent steps.

For the log mel-scale triangular filterbank output, 20 triangular filterbanks from 300Hz to 3700Hz were used, and therefore the 256-dimensional spectra were reduced to 20 dimensions. After applying discrete cosine transform (DCT), we obtained the MFCCs.

For each of the input types, we packed it with another input node that represents the SNR to form the input patterns of the SNR-adaptive denosing autoencoder. The structure of hidden layers is identical for all input types. It has been shown [14] that it is beneficial to apply Z-norm to the input vectors. In our experiments, the SNR and the input features are normalized independently, i.e. the mean and standard deviation of the SNR and the bottleneck features were estimated separately.

In addition to the preprocess techniques above, we can also use a contextual window covering several frames as the input to the DNN. For example, a sliding window covering 7 frames of mel filterbank outputs consists of $20 \times 7 = 140$ nodes. Together with the SNR node, there are 141 nodes in the input layer. Fig. 1 shows the architecture of the denoising autoencoder with the SNR node omitted.

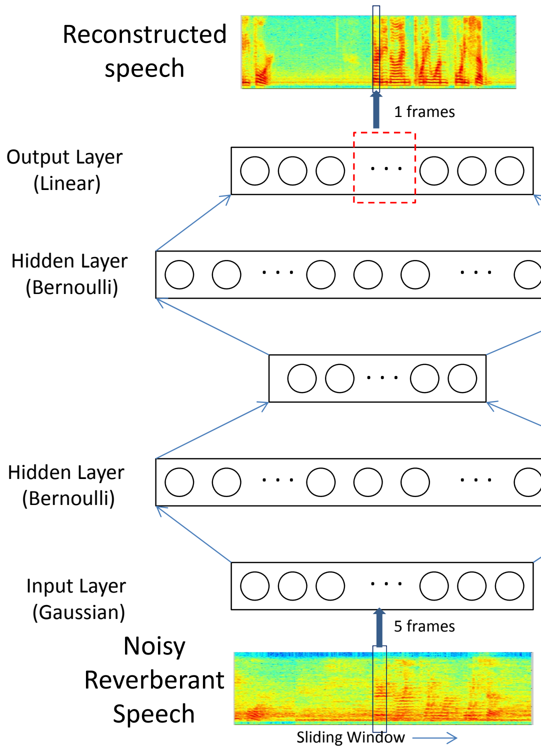

Fig. 1. Denoising deep autoencoder.

B. RBM Pre-training

Backpropagation (BP) [15] is commonly used for training DNNs. However, BP is a gradient descent algorithm, which could be easily trapped in local minima, especially when the neural network has a deep structure (having many hidden layers). This is because the gradients in the bottom layers are too small. In this case, we can consider the DNN as comprising a number of stacked RBMs, which is trained layer-by-layer via the contrastive divergence algorithm. It is commonly believed that this pre-training step can bring the DNN close to the global optimal solution, which helps the backpropagation algorithm to convergence to a better solution.

RBM is an energy-based model in which nodes within the same layer do not have interaction. For denoising deep autoencoders, the structure is symmetric with respect to the middle layer. Thus we only need to train the first half of the network, i.e., from the input to the middle layer, and then copy the parameters to the upper half of the network before the backpropagation fine-tuning (see Fig. 2).

The nodes of RBMs in the middle layers of the autoencoder follows a Bernoulli distribution, which means that both the visible and hidden units in the middle layers are binary. However, the nodes in the input layer should follow a Gaussian distribution, because log-spectra and MFCC follow Gaussian distributions.

In the Bernoulli-Bernoulli RBM, the activation function is the sigmoid function:

$$s(z) = \frac{1}{1+e^{-z}} \tag{1}$$

and the energy function is defined as:

$$E(\mathbf{v},\mathbf{h}) = -\sum_{i\in\text{vis}} a_i v_i - \sum_{j\in\text{hid}} b_j h_j - \sum_{i,j} v_i h_j w_{ij} \tag{2}$$

where v_i and h_j are the binary states of visible unit i and hidden unit j respectively, a_i and b_j are their biases, and w_{ij} is the weight between the i-th visible unit and the j-th hidden unit. By using contrastive divergence [12], we maximize the probability that the network assigns to a visible vector, $\mathbf{v}$:

$$p(\mathbf{v}) = \frac{1}{Z}\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}. \tag{3}$$

For Gaussian-Bernoulli cases, the activation function in the visible layer is linear, and the energy function is defined by:

$$E(\mathbf{v},\mathbf{h}) = \sum_{i\in\text{vis}} \frac{(v_i-a_i)^2}{2\sigma_i^2} - \sum_{j\in\text{hid}} b_j h_j - \sum_{i,j} \frac{v_i}{\sigma_i} h_j w_{ij} \tag{4}$$

where σ_i is the standard deviation of the Gaussian noise for visible unit i.

C. Backpropagation Fine-tuning

After RBM pre-training, we can stack the RBMs, copy the parameters in the lower half of the DBN to the upper half, and then fine-tune them by using backpropagation, as Fig. 2 illustrates.

In backpropagation training, we presented a mini-batch of input patterns and update the network parameters to bring the actual outputs closer to the target values. To equip our autoencoder with denoising ability, we used noisy speech as input and their corresponding clean counterparts as target outputs, while the error function is the squared loss, $L(z,\tilde{z}) = \|z-\tilde{z}\|_2^2$. However, we kept the SNR component in the output the same as the input in the experiments since we only focused on the speech denoising capability of this autoencoder.

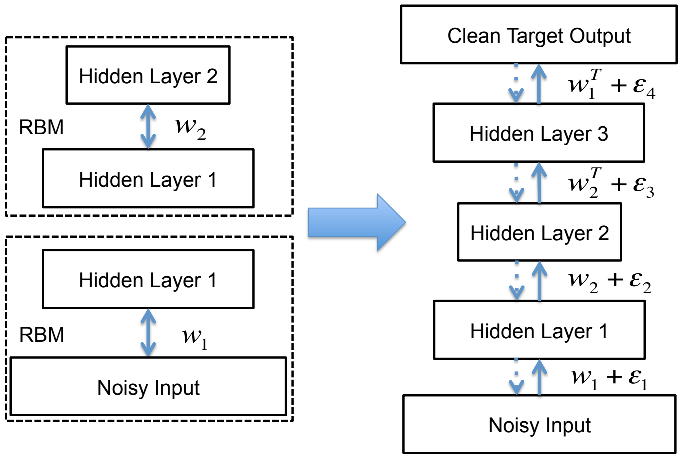

Fig. 2. Construction of a denoising autoencoder by training two RBMs layer-by-layer and then stacking them symmetrically, followed by backpropagation fine-tuning.

Backpropagation [15] uses the chain rule to iteratively compute the error gradient of each layer, collects the gradients from top to bottom layers, and updates the weights layer by layer. In our experiments, the first three hidden layers are all Bernoulli layers, and therefore they have a sigmoid activation function. However, the output layer of the autoencoder, which aims to reconstruct the input, uses a linear activation function. The autoencoder is a key component of the DNN classifier, as Fig.3 shows.

III. SNR-Adaptive Denoising Deep Classifier

Because of the denoising autoencoder, the DNN learns how to extract clean information from the noisy input patterns. However, our goal is to enable the DNN to extract speaker-dependent features. To this end, we construct a speaker classifier by putting two more layers of RBMs on top of the autoencoder as shown in Fig. 3. Finally, a softmax layer with the number of nodes equals to the number of training speakers is added to the network. Backpropagation is then applied to fine-tune the DNN by minimising the cross entropy error. Specifically, we assume that we have N training speakers whose spectral feature vectors and speaker labels are given by

$$\begin{aligned}\mathcal{X} &= \{\mathbf{x}_{i,j} \in \Re^D; i = 1, \ldots, N; j = 1, \ldots, M_i\} \\ \mathcal{C} &= \{\mathbf{c}_{i,j} \in \Re^N; i = 1, \ldots, N; j = 1, \ldots, M_i\}\end{aligned} \tag{5}$$

where $\mathbf{c}_{i,j}$'s are one-of-N vectors indicating to which speaker the spectral vector $\mathbf{x}_{i,j}$ belongs and M_i is the number of vectors from speaker i. Then, we minimize the cross-entropy error:

$$\begin{aligned}E(\mathcal{X}, \mathcal{C}) &= -\sum_{i=1}^{N}\sum_{j=1}^{M_i}\sum_{k=1}^{N} c_{i,j,k} \log(f(\mathbf{x}_{i,j})_k) \\ &= -\sum_{i=1}^{N}\sum_{j=1}^{M_i}\sum_{k=1}^{N} c_{i,j,k} \log(y_{i,j,k})\end{aligned} \tag{6}$$

where k indexes to the output nodes of the DNN, $f(\cdot)$ represents the mapping function of the DNN, and $y_{i,j,k}$ represents the output of the k-th output node subject to the input vector $\mathbf{x}_{i,j}$.

After backpropagation fine-tuning, bottleneck features can be extracted from the output (before sigmoid nonlinearity) of the BN layer in a frame-by-frame basis as shown in Fig. 3.

The first RBM (comprising *Hidden Layer 4* and *Hidden Layer 5* in Fig. 3) is Gaussian-Bernoulli due to the characteristic of the autoencoder's reconstruction layer. The top-most RBM (comprising *Hidden Layer 5* and *BN Layer* in Fig. 3) is Bernoulli-Bernoulli. Because what we require is a classifier, the last layer (*Speaker ID* in Fig. 3) is a softmax layer, and therefore no RBM pre-training for this layer was applied. Its size is equal to the number of classes, which in our case is the number of training speakers here. We used the 1-of-N coding scheme for the output nodes. Specifically, for each input vector, the desired output of the nodes in the softmax layer are all zeros, excepting the one that indicates the speaker to which the input vector belongs. After BP training, given a noisy input spectral vector $\mathbf{x}$, the k-th node of the softmax layer is computed as

$$y_k = f(\mathbf{x}) = \frac{e^{h_k}}{\sum_{k'=1}^{N} e^{h'_k}} \tag{7}$$

where $f(\mathbf{x})$ represents the whole DNN mapping function and h_k is the output of the BN layer, which implicitly depends on $\mathbf{x}$ through the other hidden layers and the autoencoder.

Note that there are two layers in the DNN that do not use the sigmoid function as the activation function. The first one is the reconstruction layer in the autoencoder, which uses the linear function instead; the second one is for classification, which uses the softmax function.

Fig. 4 shows the spectrogram and histograms of log-spectra of the clean, 0dB noisy, and denoised speech, respectively. The figure shows that the noise seriously distorted the spectral patterns of the clean speech. However, the denoising deep autoencoder performs very well in restoring the patterns, as demonstrated in the third panel of the figure. The histogram on the right of Fig. 4 shows that after denoising, the distribution of log-spectra is non-Gaussian. However, after applying BP to fine tune the whole DNN, the output of the autoencoder follows a Gaussian-like distribution.

A. *iVector-PLDA Speaker Identification*

The iVector [16] and probabilistic LDA (PLDA) [17] are commonly used for speaker verification. The idea is to represent the speaker and channel characteristics of an utterance by a low-dimensional vector called the iVector, which is essentially the posterior mean of the latent factors of a factor analysis model. Given an iVector, the channel variability is removed by marginalizing out the channel factors in a PLDA model (which is a supervised factor analysis model) during verification. This iVector-PLDA framework was originally designed for speaker verification, which is a binary classification problem.

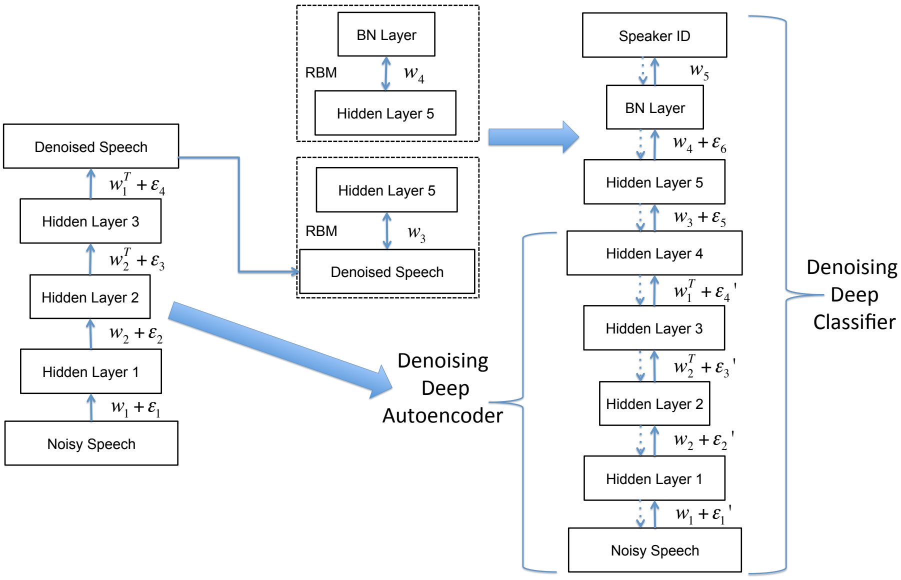

Fig. 3. Constructing the DNN classifier and bottleneck (BN) feature extractor by stacking two RBM layers (*Hidden Layer 5* and *BN Layer*) and a softmax layer (*Speaker ID*) on top of the autoencoder (from *Noisy Speech* to *Hidden Layer 4*), followed by backprogagation fine-tuning. Note that after fine-tuning, only the features extracted from the BN layer will be used for iVector-PLDA speaker identification.

In this work, we applied this framework to speaker identification, which is a multi-class problem. First, we used the utterances of all training speakers to train an iVector extractor. Then, the iVectors of each registered speaker in the speaker identification system were computed, one for each enrollment utterance. Also, the iVectors of all training speakers (who can be different from the registered speakers) were extracted. These training iVectors together with their speaker labels were used for training a PLDA model [18], [19].

During identification, given a test utterance, an iVector is computed. Then, the test iVector is scored against the iVectors of each of the registered speakers using the PLDA scoring function [20], [19] and the scores were averaged. For a system comprising R registered speakers, these steps give R averaged scores for each test utterance, and the speaker identity corresponds to the maximum averaged score. More precisely, denote $\mathbf{w}_{i,j}$ as the iVector of the j-th session of the i-th registered speaker in the system. Then, given a test utterance with iVector $\mathbf{w}_t$, the speaker ID of the test utterance is

$$\text{ID}(\mathbf{w}_t) = \arg\max_{i=1}^{R} \frac{1}{N_i} \sum_{j=1}^{N_i} S_{\text{PLDA}}(\mathbf{w}_{i,j}, \mathbf{w}_t), \qquad (8)$$

where N_i is the number of enrollment iVectors of speaker i.

IV. Experiments

A. Experimental Setup

We performed speaker identification experiments based on 138 speakers in the YOHO corpus [21].[1] We used the enrollment sessions, which consist of 96 utterances per speaker, as training data. We used the verify sessions of the corpus as testing data, in which there are 40 utterances per speaker.[2] Totally, there are 13,248 utterances for training and 5516 utterances for testing. Each utterance is about 3 to 4 seconds long, sampled at 8kHz, and comprises three two-digit numbers in English, e.g. 26-81-57. We used the FaNT tool to add babble noise to the original YOHO utterances at 15dB, 6dB, and 0dB. Thus, there are totally $13,248 \times 4 = 52,992$ utterances for training the DNN classifier. We used an energy-based voice activity detector [22], [23] to extract the speech regions of each utterance.

To train the autoencoder part of the classifier, each clean utterance is paired with itself and its noisy counterparts, which amounts to 13,248 clean–clean training pairs and $13,248 \times 3 = 39,744$ clean–noisy training pairs. The autoencoder comprises three hidden layers, whose weights were trained by using both noisy and clean speech data as input and only clean speech data as the target output. Each of the hidden layers contains 256 nodes. Our aim is to enable it to perform speech denoising in the spectral domain. To achieve this, we split the training data into mini-batches comprising 100 consecutive spectra and applied 30 epochs of RBM pre-training, and then used mini-batches comprising 1,000 consecutive spectra and applied 100 epochs of backpropagation fine-tuning using gradient descent with a learning rate of 0.1 and momentum of 0.6.

By using the denoised spectra from the autoencoder, we trained another two RBMs with 256 and 60 hidden nodes, respectively. Because the number of training speakers is 138, the DNN has 138 output nodes, i.e., $N = 138$ in Eq. 5. However,

[1] To minimize computation time in this pilot study, we did not use the NIST evaluation corpora. This will be our future work.

[2] Except for Speaker 277 whose only have 36 valid utterances in the verify sessions.

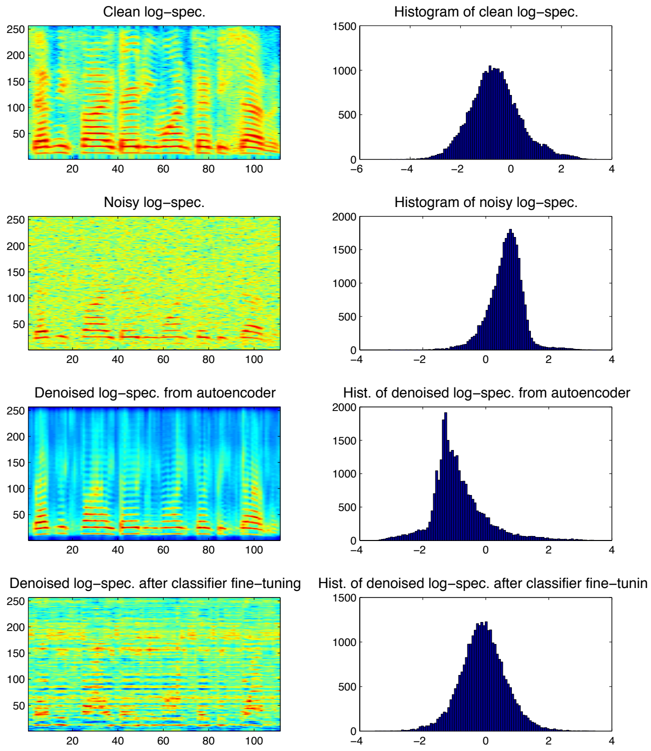

Fig. 4. The spectrograms and histograms of log-spectra of clean, 0dB noisy, and denoised speech.

because the cross-entropy error has a larger fluctuation when fine-tuning the whole DNN, we reduced the learning rate to 0.0003.

Finally, we obtain a denoising deep classifier with D-256-256-256-D-256-60-138 nodes in the respective layer starting from input to output, where D represents the dimension of the spectral vectors.

In the experiments, the 60-dimensional bottleneck features extracted from the second-top layer in the classifier before sigmoid non-linearity were compared with 60-dimensional MFCC baseline features. For the former, the frame-based BN features were whitened using PCA whitening. For the latter, we computed 19 MFCCs and the log-energy for each frame. Then we packed the MFCCs and log-energy together with their first and second order derivatives to form a 60-dimensional acoustic vector for each frame.

In the back-end system, we used the state-of-the-art iVector [16] and probabilistic LDA (PLDA) [18]. To train an iVector extractor, we used all of the 52992 training utterances from the clean and the three SNR conditions to train a universal background model (UBM) with 256 Gaussians and a total variability matrix with 400 factors. For each utterance, an iVector was extracted from the iVector extractor [16] so that the speaker characteristics of the entire utterance is represented by this 400-dimensional vector. Given 52992 training utterances, we have 52992 iVectors. Then they were used for training an SNR-independent PLDA model with 138 latent factors by grouping the iVectors of the same speaker together. Because there are 138 training speakers, we have 138 groups of speaker-dependent iVectors and each group comprises $96 \times 4 = 384$ iVectors.

B. Results of Denoising BN Features

We used three types of input for the DNN: 1 frame of 256-dimensional log-spectra (*Log-spec*), a contextual window covering 7 frames of 20-dimensional log mel-scale triangular filterbank output (*Log-mel*), and 5 frames of 60-dimensional MFCC (*MFC*) to generate the BN features. These 3 types of inputs result in Log-spec BN, Log-mel BN and MFC BN features, respectively.

Table I shows the accuracy of speaker identification, which is a bit disappointing in that only the Log-mel BN features with the Log-mel input are comparable with the standard MFCC under high SNR conditions and outperform it under low SNR conditions. Besides, it is surprising that the MFC BN features using 5 frames of MFCC as input always perform worse than the Log-mel ones and the standard MFCC. This may be due to the smearing effect of the denoising autoencoder. Further investigations are warranted to investigate the reasons behind the poor performance.

The Log-spec BN feature performs slightly poorer than the MFC BN feature under both clean and noisy conditions. We suspect that the poor performance is due to the lack of contextual frames (in *Log-spec*, the size of contextual window is 1) in the input. However, we found that increasing the contextual window size to 5 leads to even poorer performance. This could be caused by the high-dimensionality of the log-spectral vectors, which forbids us to use multiple frames in the contextual window.

TABLE I
COMPARISON BETWEEN MFCC AND BN FEATURES

Feature	SNR of Test Utterances			
	Clean	15dB	6dB	0dB
MFCC	**98.31%**	95.61%	90.08%	65.65%
Log-spec BN	95.56%	93.04%	83.39%	62.98%
Log-mel BN	98.21%	**96.77%**	**91.48%**	**75.29%**
MFC BN	97.44%	94.24%	86.84%	63.61%

C. PLDA Score Combination

Because the BN features, especially the Log-mel one, perform well under low SNR conditions, we can fuse the MFCC and the BN features to improve the performance of speaker identification at the PLDA score level:

$$S_{\text{fused}} = \alpha \times S_{\text{mfcc}} + (1 - \alpha) S_{\text{bn}}, \tag{9}$$

where α is the fusion weight and S denotes the PLDA scores. When $\alpha = 1$, no fusion is performed and only MFCC features were used. On the other hand, when $\alpha = 0$ only BN features were considered. We varied the value of α with a step size of 0.01. As Table II illustrates, the Log-mel BN features with score fusion increase the accuracy significantly.

TABLE II
FUSION OF THE PLDA SCORES BASED ON MFCC AND BN FEATURES

Feature	Fusion Weight α (Eq. 9)	SNR of Test Utterances			
		Clean	15dB	6dB	0dB
MFCC	1.00	98.31%	95.61%	90.08%	65.65%
Log-spec BN	0.00	95.56%	93.04%	83.39%	62.98%
	0.57	99.15%	98.11%	94.13%	79.89%
Log-mel BN	0.00	98.21%	96.77%	91.48%	75.29%
	0.51	**99.56%**	**98.55%**	**95.87%**	**84.34%**
MFC BN	0.00	97.44%	94.24%	86.84%	63.61%
	0.56	98.89%	96.72%	93.31%	76.53%

V. CONCLUSIONS AND FUTURE WORK

This paper shows that Log-mel BN features from denoising deep classifier are noise robust under low SNR environments. The noise robustness is mainly attributed to the autoencoder's denoising ability, which facilitates the upper hidden layers of the DNN to extract more noise robust bottleneck features. The BN features are comparable with the standard MFCC, and they are complementary to each other, leading to significant performance gain after fusing the MFCC- and BN-based PLDA scores.

In the experiment, we did not use a contextual window for the 256-dimensional Log-spec input, since the input dimension is very high when compared with other types of input. Even without contextual window, the performance of the Log-spec BN features is still comparable with that of the Log-mel ones, but the performance under low SNR conditions drops significantly.

The poor performance of the MFC BN features is surprising. The reason is possibly that the distribution of the MFCC is more complex than a single Gaussian (otherwise we do not need GMM for speaker recognition), and therefore the Gaussian-Bernoulli RBM cannot model the input patterns properly. Some other preprocessing techniques deserve a try for the MFCC input, e.g. the feature warping [24].

Our preliminary experiment is done on the YOHO corpus, whose speech contents do not have much phonetic variation. In future work, experiments on NIST datasets are necessary.

ACKNOWLEDGMENT

This work was in part supported by The RGC of Hong Kong SAR, Grant No. PolyU 152117/14E.

REFERENCES

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

[2] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[3] H. Ze, A. Senior, and M. Schuster, "Statistical parametric speech synthesis using deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 7962–7966.

[4] Z.H. Ling, S.Y. Kang, H. Zen, A. Senior, M. Schuster, X.J. Qian, H.M. Meng, and L. Deng, "Deep learning for acoustic modeling in parametric speech generation: A systematic review of existing techniques and future trends," *Signal Processing Magazine, IEEE*, vol. 32, no. 3, pp. 35–52, May 2015.

[5] P. Hamel and D. Eck, "Learning features from music audio with deep belief networks.," in *ISMIR*. Utrecht, The Netherlands, 2010, pp. 339–344.

[6] G. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[7] W.M. Campbell, "Using deep belief networks for vector-based speaker recognition," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[8] Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1695–1699.

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

[10] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *The Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[11] S. Yaman, J. Pelecanos, and R. Sarikaya, "Bottleneck features for speaker recognition," in *Odyssey*, 2012, vol. 12, pp. 105–108.

[12] G. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[13] M. Sahidullah and G. Saha, "Design, analysis and experimental evaluation of block based transformation in mfcc computation for speaker recognition," *Speech Communication*, vol. 54, no. 4, pp. 543–565, 2012.

[14] E. Turajlic and O. Bozanovic, "Neural network based speaker verification for security systems," in *Telecommunications Forum (TELFOR), 2012 20th*, 2012, pp. 740–743.

[15] D.E. Rumelhart, G. Hinton, and R. Williams, "Learning representations by back-propagating errors," *Cognitive Modeling*, vol. 5, 1988.

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

[17] P. Kenny, "Bayesian speaker verification with heavy-tailed priors," in *Proc. of Odyssey: Speaker and Language Recognition Workshop*, Brno, Czech Republic, June 2010.

[18] S. Prince and J.H. Elder, "Probabilistic linear discriminant analysis for inferences about identity," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, 2007, pp. 1–8.

[19] N. Li and M.W. Mak, "SNR-invariant PLDA modeling in nonparametric subspace for robust speaker verification," *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, vol. 23, no. 10, pp. 1648–1659, Oct 2015.

[20] D. Garcia-Romero and C.Y. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems," in *Interspeech'2011*, 2011, pp. 249–252.

[21] J. Campbell, "Testing with the yoho cd-rom voice verification corpus," in *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*. IEEE, 1995, vol. 1, pp. 341–344.

[22] M. W. Mak and H. B. Yu, "A study of voice activity detection techniques for NIST speaker recognition evaluations," *Computer, Speech and Language*, vol. 28, no. 1, pp. 295–313, Jan 2013.

[23] H.B. Yu and M.W. Mak, "Comparison of voice activity detectors for interview speech in NIST speaker recognition evaluation," in *Interspeech*, 2011, pp. 2353–2356.

[24] J. Pelecanos and S. Sridharan, "Feature warping for robust speaker verification," in *A Speaker Odyssey - The Speaker Recognition Workshop*, 2001, pp. 213–218.