VARIABLE-ACTIVATION AND VARIABLE-INPUT DEEP NEURAL NETWORK FOR ROBUST SPEECH RECOGNITION

Rui Zhao[1], *Jinyu Li*[2], and *Yifan Gong*[2]
[1]Microsoft Search Technology Center Asia, Beijing, China
[2]Microsoft Corporation, One Microsoft Way, Redmond, WA 98052

ABSTRACT

In a previous study, we proposed a variable-component deep neural network (VCDNN) to improve the robustness of context-dependent deep neural network hidden Markov model (CD-DNN-HMM). We model the components of DNN with a set of polynomial functions of environmental variables, more specifically signal-to-noise ratio (SNR) in that study. We refined VCDNN on two types of DNN components: (1) weighting matrix and bias (2) the output of each layer. These two methods are called variable-parameter DNN (VPDNN) and variable-output DNN (VODNN). Although both methods got good gain over the standard DNN, they doubled the number of parameters even with only the first-order environment variable. In this study, we propose two new types of VCDNN, namely variable activation DNN (VADNN) and variable input DNN (VIDNN). The environment variable is applied to the hidden layer activation function in VADNN, and is applied directly to the input in VIDNN. Both VCDNNs only increase a negligible number of parameters compared to the standard DNN. Experimental results on the Aurora4 task show that both methods have similar performance as VPDNN, obtaining around relative 3.71% word error reduction from the standard DNN with negligible increase in number of parameters.

Index Terms— deep neural network, variable component, variable input, variable activation, robust speech recognition

1. INTRODUCTION

Recently, a new acoustic model, referred to as the context-dependent deep neural network hidden Markov model (CD-DNN-HMM), has been developed. It has been shown, by many groups [1][2][3][4][5][6], to outperform the conventional GMM-HMMs in many automatic speech recognition (ASR) tasks. However, there are only very few works to investigate the effectiveness of CD-DNN-HMM on noise-robust ASR tasks [7][8][9][10], although robustness is still very challenging to real-world applications [12][13].

In a previous study, we proposed a model-based noise-robust method called variable-component DNN (VCDNN) [14], which is inspired by the idea from the variable-parameter HMM (VPHMM) method [15]. In VCDNN, we want to have any component in the DNN to be modeled as a set of polynomial functions of an environment variable. In [14], we investigated two types of variation: variable-parameter DNN (VPDNN) in which the weight matrix and bias are environment-variable dependent, and variable-output DNN (VODNN) in which the output of each hidden layer is environment-variable dependent. As in VPHMM, the variable-dependent components are computed online for the environment condition detected in the testing data using their associated polynomial functions during recognition.

Although better performance is achieved, even with the first-order environment variable, VPDNN and VODNN doubled the number of parameters from the standard DNN. The impact of an environment variable to the DNN should be in a low dimension space. Therefore, we should be able to use only a limited number of parameters to handle it. In this paper, we propose two new types of VCDNN, namely variable activation DNN (VADNN) and variable input DNN (VIDNN). An environment variable is applied to the hidden layer activation function in VADNN, and is applied directly to the input in VIDNN. Both DNNs only slightly increase the number of parameters. Experimental results on the Aurora4 task [16] show that both methods are very effective, obtaining relative around 3.71% word error reduction from the standard DNN, with negligible increase in number of parameters.

This paper is organized as follows. In Section 2, we review the standard DNN and previously proposed VPDNN and VODNN. Then, in Section 3, the proposed VADNN and VIDNN will be described in detail. In Section 4, the experimental results on Aurora4 will be presented. Finally, the conclusions and future works will be given in Section 5.

2. STANDARD DNN, VPDNN AND VODNN

In this section, we first describe the standard DNN formulation and training methods. Then, we introduce our previously proposed variable-component DNN (VCDNN) methods in the form of variable-parameter DNN (VPDNN) and variable-output DNN (VODNN) [14].

2.1 Standard DNN

The standard DNN can be considered as a multi-layer perceptron (MLP) consisting of one input layer, one output layer and many hidden layers. Each node in the output layer represents one senone.

Usually, a sigmoid function is chosen as the activation function for hidden layers of DNN and the output of the l-th hidden layer o^l is given by:

$$o^l = f_{sigm}(u^l) \quad (1)$$

$$u^l = (W^l)^T o^{l-1} + b^l \quad (2)$$

where o^{l-1} is the input of the l-th layer, W^l and $b^l$are the weight matrix and bias of the l-th layer, respectively. $f_{sigm}(x) = 1/(1+e^x)$.

The activation function of the output layer (layer L) is a softmax function

$$o_k^L = \frac{exp(u_k^L)}{\sum_{i=1}^{S} exp(u_i^L)}. \quad (3)$$

Hence, the senone posterior probability $P(s_k|x)$ is:

$$P(s_k|x) = \frac{exp(u_k^L)}{\sum_{i=1}^{S} exp(u_i^L)}, \quad (4)$$

where x is the input feature vector of DNN, s_k is the senone responding to unit k of the top layer, and S is the total number of senones. The first layer's input $o^0 = x$. The senone emission likelihood of HMM $p(x|s)$ is then calculated according to

$$p(x|s) = P(s|x) \cdot p(x)/P(s). \quad (5)$$

$P(s)$ is the prior probability of senone s. $p(x)$ is independent of s and can be ignored during HMM decoding.

In DNN training, the commonly used optimization criterion is the cross-entropy between the posterior distribution represented by the reference labels $\hat{P}(s|x)$ and the predicted distribution $P(s|x)$. The objective function is:

$$F_{CE} = -\sum_{i=1}^{S} \hat{P}(s_i|x)\log\big(P(s_i|x)\big). \quad (6)$$

The reference label is typically decided based on the forced-alignment:

$$\hat{P}(s_i|x) = \begin{cases} 1 & \text{if x is aligned to senone } s_i \\ 0 & \text{else} \end{cases}. \quad (7)$$

Then equation (6) is simplified as:

$$F_{CE} = -\log\big(P(s'|x)\big), \quad (8)$$

where s' is the reference senone for the speech input x.

With the above objective function, a DNN can be trained with the method introduced in [1], which consists of unsupervised pre-training and supervised fine-tuning. The algorithm used in the fine-tuning stage is error back propagation, where the weight matrix W and bias b of layer l are updated with:

$$\hat{W}^l = W^l + \alpha o^{l-1}(e^l)^{\mathrm{T}} \quad (9)$$

$$\hat{b}^l = b^l + \alpha e^l \quad (10)$$

where α is the learning rate. o^{l-1} and e^l are the input and error vector of layer l, respectively. e^l is calculated by propagating the error from its upper layer:

$$e_i^l = \Big[\sum_{k=1}^{N_{l+1}} w_{ik}^{l+1} e_k^{l+1}\Big] f'_{sigm}\big(u_i^l\big), \quad (11)$$

where w_{ik}^{l+1} is the element of weight matrix W^{l+1} in i-th row and k-th column for layer $l+1$, and e_k^{l+1} is the k-th element of error vector e^{l+1} for layer $l+1$. N_{l+1} is the node number in layer $l+1$. $f'_{sigm}\big(u_i^l\big)$ is the derivative of sigmoid function. The error of the top layer (i.e. output layer) is the derivative of the objective function defined in equation (8).

$$e_s^L = -\frac{\partial F_{CE}}{\partial u_s^L} = (\delta_{ss'} - o_s^L). \quad (12)$$

$\delta_{ss'}$ is the Kronecker delta function.

2.2 VPDNN and VODNN

The basic idea in VCDNN is to refine the DNN components by modeling their variation against environment changes, which is not explicitly taken into consideration in a standard DNN. In [14], we have specifically worked on two types of components of DNN: (a) VPDNN: on weight matrix and bias, and b) VODNN: on the output of each layer.

In VPDNN, the weight matrix W and bias b of layer l is modeled as a function of the environment variable v:

$$W^l = f_w^l(v)\,,\ b^l = f_b^l(v) \quad 0 < l \le L\,. \quad (13)$$

Here, we use a polynomial function for both f_w^l and f_b^l based on its advantages and effectiveness shown in VPHMM [15]. SNR is selected as the environment variable. So we have:

$$W^l = \sum_{j=0}^{J} H_j^l v^j \quad 0 < l \le L \quad (14)$$

$$b^l = \sum_{j=0}^{J} p_j^l v^j \quad 0 < l \le L \quad (15)$$

In VODNN, we assume the output of each hidden layer could be described by a polynomial function of the environment variable v.

$$o^l = \sum_{j=0}^{J} f_{sigm}\big(u_j^l\big) v^j \quad 0 < l < L\,, \quad (16)$$

where

$$u_j^l = \big(H_j^l\big)^T o^{l-1} + p_j^l \quad 0 < l < L\,. \quad (17)$$

In equation (14) to (16), J is the polynomial function order. H_j^l is a matrix with the same dimensions as W^l and p_j^l is a vector with the same dimension as b^l. In VPDNN and VODNN, H_j^l and p_j^l ($0 \le j < J$) need to be estimated instead of W^l and b^l, so the parameters number is $J+1$ times of that in the standard DNN.

In [14] SNR is normalized with a sigmoid function $\tilde{v} = f_{sigm}(\beta v)$ for both VPDNN and VODNN because the numerical value range of raw SNR is too big compared with the DNN components. $-1 < \beta < 0$ is a constant to make the normalized SNR for the clean data to be close to zero.

3. VADNN AND VIDNN

Although better recognition performance is achieved, VPDNN and VODNN doubled the number of parameters from the standard DNN even with the first-order SNR variable. We believe that the impact of SNR to DNN could be represented in a low dimension space. Therefore, we should be able to use only a limited number of parameters to handle it. In this section, we propose two new types of VCDNN, namely variable activation DNN (VADNN) and variable input DNN (VIDNN). SNR variable is applied to hidden layer activation function in VADNN, and is applied directly to the input in VIDNN. The increased number of parameters of both DNNs is almost negligible compared to that of the standard DNN.

3.1 VADNN

In VADNN, the activation function of hidden layer has environment-variable-dependent parameters.

$$o^l = f_{sigm}\big(a^l \circ u^l + m^l\big), \quad (18)$$

where $a^l \circ u^l$ means the element-wise product of vector a^l and u^l. a^l and m^l are defined as the polynomial functions of SNR:

$$a^l = \sum_{j=0}^{J} h_j^l v^j \quad (19)$$

$$m^l = \sum_{j=0}^{J} p_j^l v^j \quad (20)$$

We can that see for VADNN, the additional variable-dependent parameters $h_j^l = [h_{j1}^l, h_{j2}^l, \dots, h_{jN_l}^l]$ and $p_j^l = [p_{j1}^l, p_{j2}^l, \dots, p_{jN_l}^l]$ for each hidden layer are vectors with dimension N_l, which is the

number of nodes of layer l. So its number of parameters is much smaller than that in VPDNN or VODNN.

In the training of VADNN, we need to estimate h_j^l and p_j^l as well as the standard DNN parameters W^l and b^l. The initial value of a^l and m^l are set to:

$$h_{jk}^l = 1 \text{ if } j = 0$$
$$h_{jk}^l = 0 \text{ if } j > 0 \quad (21)$$
$$p_{jk}^l = 0$$

The updating formula can be derived with error back propagation algorithms as below:

$$\hat{w}^l = w^l + \alpha o^{l-1}\left(e^l \circ a^l\right)^{\mathrm{T}} \quad (22)$$
$$\hat{b}^l = b^l + \alpha\left(e^l \circ a^l\right) \quad (23)$$
$$\hat{h}_j^l = h_j^l + \alpha\left(e^l \circ u^l\right)v^j \quad (24)$$
$$\hat{p}_j^l = p_j^l + \alpha e^l v^j \quad (25)$$

In the recognition stage, the activation output for each hidden layer is calculated according to (18) with the estimated SNR of the testing data. Then the senone posterior is computed with equation (4) as in standard DNN.

3.2 VIDNN

In VIDNN, the SNR value is concatenated to the input feature directly to make the input variable dependent, as show in Figure 1.

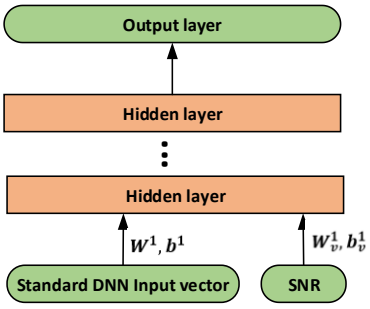


Figure 1. *Flowchart of VIDNN*

The SNR related parameters W_v^1, b_v^1 can be estimated in the same way as the standard DNN parameters W^1, b^1 which is described in section 2.1. During the recognition, the SNR is estimated for the testing data and concatenated with the standard DNN feature. Then the concatenated feature is fed into the DNN show in Figure 1 to get the senone posterior.

Now we can apply an environment variable to any component of DNN: to input (via VIDNN), to parameter (via VPDNN), to activation function (via VADNN), and to layer output (via VODNN). This completes the whole framework of VCDNN.

4. EXPERIMENTS

The proposed methods are evaluated with Aurora 4 [16], a noise-robust medium-vocabulary task based on the Wall Street Journal corpus (WSJ0). Aurora 4 has two training sets: clean and multi-condition. Each of them consists of 7138 utterances (about 14 hours of speech data). For the multi-condition training set, half of the data was recorded with a Sennheiser microphone and the other was with a secondary microphone. Besides, 6 types of noises (car, babble, restaurant, street, airport, and train) were added with SNRs from 10 to 20dB. The subset recorded with the Sennheiser microphone was called as channel wv1 data and the other part as channel wv2 data.

The test set contains 14 subsets. 2 of them are clean and the other 12 are noisy. The noisy test sets were recorded with the same types of microphone as in multi-condition training set. Also, the same 6 types of noise as in multi-condition training set were added with SNRs between 5 and 15 dB.

The acoustic feature of baseline CD-DNN-HMM system are 24-dimensional log Mel filter-bank static features without utterance-level mean normalization plus their first- and second-order derivative features, totally 72 dimensions. The dimension of the DNN input layer is 792, formed from a context window of 11 frames. Its output layer contains 1209 units, which means there are 1209 senones in the HMM system. The DNN has 5 hidden layers with 2048 units in each layer.

4.1. Experiment Results

In the experiments, we compare the standard DNN with variations of VCDNN: VPDNN, VADNN, and VIDNN. VODNN is not included here because it has similar performance as VPDNN [14]. The standard DNN and VCDNNs are all trained with the noisy wv1 data from multi-condition training set. The test data are clean and six noisy wv1 sub sets. The reason we only choose wv1 data for the experiments is that wv2 data contains channel distortion which is another factor affecting the performance of ASR. Currently we want to only consider noise as a factor and use SNR as the environment variable, and in the future we will introduce channel distortion into our modeling. The results are given in Table 1. The results of standard DNN and VPDNN are slightly different from those in [14] because the DNN models in [14] were initiated first with clean data only and updated with multi-condition data. That setup may not be available in realistic condition. Therefore, in this paper we only use the available multi-condition data for all the experiments.

Table 1. *Comparison of the standard DNN with the first-order VCDNNs using SNR as the environment variable*

Model	WER
standard DNN	11.31
VPDNN	10.74
VADNN	10.89
VIDNN	10.92

In Table 1, for VPDNN and VADNN, the normalized SNR is used as the variable because the numerical value range of raw SNR is too big compared with the DNN components, which may bring some optimization problem to VCDNN. However, it is popular to remove the sigmoid or softmax nonlinearity of a variable if it is used as the input of a network (e.g., TANDEM [17] or bottle-neck [18] feature). Hence, the raw SNR is used in VIDNN in Table 1. As shown in [14], the first-order polynomial is good enough to model the variation caused by SNR changes within the DNN

framework. Therefore, the first-order polynomial is used in all the experiments. The results show that the three variations of VCDNN give very similar word error rate (WER), obtaining 3.45%-5.04% relative WER reduction from the standard DNN.

Table 2 gives the parameter number for standard DNN and VCDNNs. VPDNN doubles the number of parameters from the standard DNN, VADNN and VIDNN only increase a negligible number of parameters compared to that of the standard DNN. This confirms our belief that the impact of SNR to DNN may be in a low dimension space, without the need to use a large number of parameters to model it.

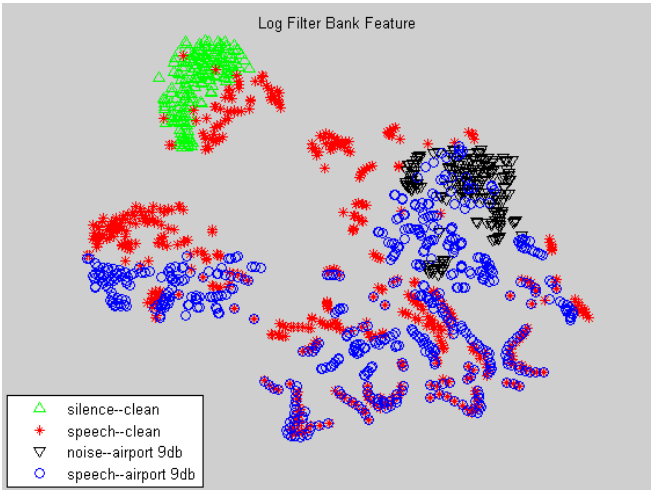
Table 2. *Comparison of the parameter number of the standard DNN with the first-order VCDNNs*

Model	Parameter number
standard DNN	20,886,713
VPDNN	39,296,185
VADNN	20,927,673
VIDNN	20,890,809

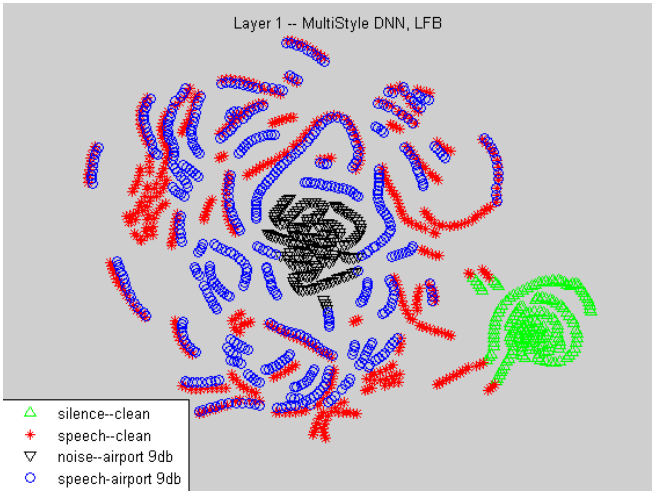
4.2. Visualized Analysis

At first glance, we are surprised at the improvement of VIDNN by only adding a single SNR value to the input vector. The original log Mel filter-bank feature (static together with its dynamic feature) has 72 dimensions. How can the additional single dimension input from SNR makes such an impact to the whole DNN? Next, we try to use t-SNE [19] to visualize the impact of adding SNR as an additional input variable. t-SNE is a tool to visualize high dimensional data in a low dimensional space. It preserves neighbor relation so that the nearby points in the two-dimensional t-SNE plot are also very close in the original high dimensional space. A parallel pair of utterances is used for the illustration. The first one is a clean utterance (447c020s in the clean test set), and the second one is synthesized from the clean one by adding airport noise with SNR 9db (447c020s in the airport test set).

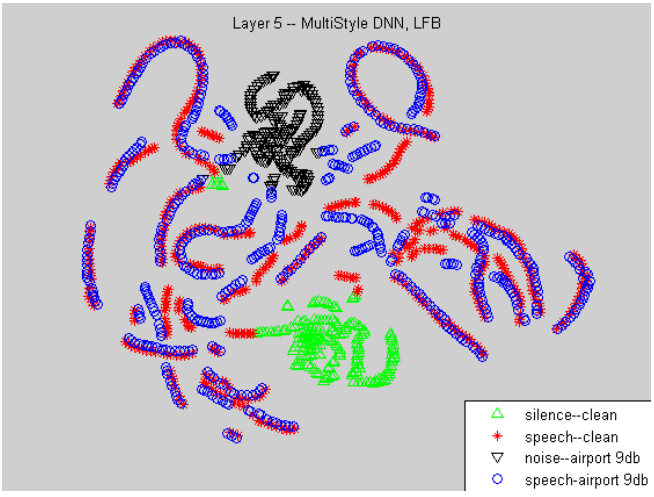
We first look at the t-SNE plots of the log filter bank feature and the 1st, 5th layer activation vectors of the corresponding DNN for this pair of stereo utterances in Figure 2. The green upper triangles correspond to the frames of pure clean silence, and the black lower triangles are the frames of pure noise. The red stars refer to clean speech frames, and the blue circle are noisy speech frames. From Figure 2.a), we can clearly see that most clean speech and noisy speech samples are scattered from each other. Also, the pure noise samples are mixed with some noisy speech samples. The DNN training provides a layer-by-layer feature extraction strategy that automatically derives powerful noise-resistant features from primitive raw data for senone classification. This invariance property can be observed in Figure 2.b) and 2.c), in which the noisy and clean speech samples become better aligned together when going through the DNN from the lower layer to the higher layer. There are still some not-well-aligned samples, which the DNN cannot perform good invariance on. If the noisy and clean samples are perfectly aligned in the t-SNE plot, that means these samples are very close to each other in the original high-dimension space. Therefore, when being presented to a classifier,



a) log filter bank feature

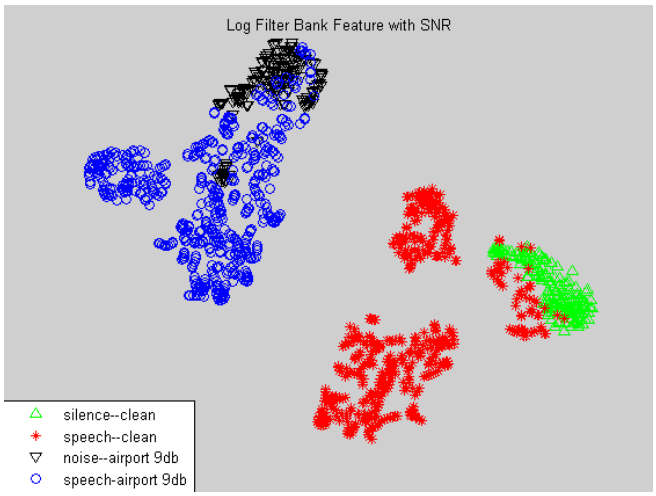


b) 1st layer activation vector

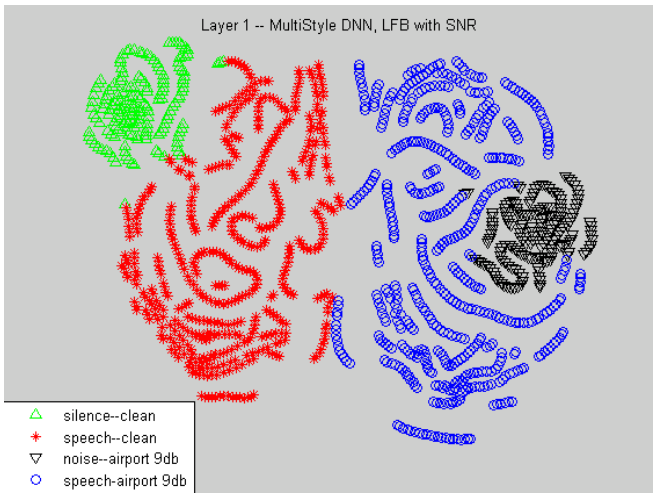


c) 5th layer activation vector

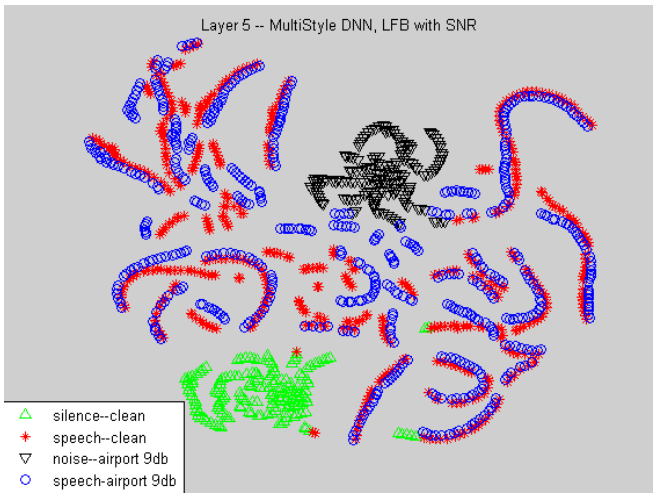
Figure 2. t-SNE plot for the log filter bank feature, and the 1st, 5th layer activation vectors of the corresponding DNN for a pair of stereo utterances



a) log filter bank + SNR feature



b) 1st layer activation vector



c) 5th layer activation vector

Figure 3. t-SNE plot for the log filter bank + SNR feature, and the 1st, 5th layer activation vectors of the corresponding VIDNN for a pair of stereo utterances

similar results will generated from these well-aligned samples. By comparing Figure 2.a) and Figure 2.c), it is clear that the activation vector in the 5th layer of the DNN is much better aligned for clean and noisy speech samples than the log filter bank feature. Note that the pure silence samples and the pure noise samples are not well aligned because the difference between the log filter bank values of pure silence and noise samples is much larger than that of the clean and noisy speech samples. As discussed in [20], the DNN's invariance property can only handle the input samples with moderate difference. Otherwise, even after going through multiple layers of a DNN, the invariance cannot be obtained.

In Figure 3, we examine the concatenated log filter bank and SNR feature, and the 1st, 5th layer activation vectors of the corresponding VIDNN for the same pair of stereo utterances. Comparing Figure 3.a) with 2.a), we can see that by concatenating SNR with the log filter bank feature, the data distribution significant changes. In Figure 3.a), the samples from the clean utterance (i.e., the pure silence and clean speech frames) are totally separated from the noisy utterance (i.e., the pure noise and noisy speech frames). This means that the impact of adding only one dimensional SNR to the original 72 dimension log filter bank feature is large. It is also interesting to compare Figure 3.b) with 2.b), in which the 1st layer activation vectors are plotted. Rather than beginning to align the samples from clean and noisy utterances together as in Figure 2.b), the function of the first layer of VIDNN is to separate the non-speech samples from speech samples. Then with multiple higher layers, the clean and noisy speech samples begin to align very well as in Figure 3.c). As a conclusion, although being of only one dimension compared to the 72 dimensions of log filter bank feature, the SNR variable really makes difference to the sample distribution of clean and noisy utterances, and results in different recognition performance.

5. CONCLUSIONS AND FUTURE WORKS

In this paper, we complete the whole framework of variable-component DNN by proposing variable activation DNN (VADNN) and variable input DNN (VIDNN). Together with previously proposed variable parameter DNN (VPDNN) and variable output DNN (VODNN), now we can apply an environment variable to any component of DNN: to input (via VIDNN), to parameter (via VPDNN), to activation function (via VADNN), and to layer output (via VODNN). Both VADNN and VIDNN only increase a negligible number of parameters compared to standard DNN. This is much better than the previously proposed VPDNN and VODNN, which double the number of parameters even with the first-order environment variable.

Experimental results on the Aurora4 task show that all the variations of VCDNN give similar WERs They obtained 3.45%-5.04% relative WER reduction from the standard DNN. The advantage of VADNN and VIDNN proposed in this paper is that they can achieve satisfactory WER reduction with negligible increase number of parameters. Furthermore, there is almost no cost to implement VIDNN which only needs to estimate the SNR value for the current utterance and then concatenate it with the original feature. We also use a t-SNE plot to show that although the SNR variable only has one dimension, it makes a clear impact to the data distribution when concatenated with the 72-dimensional log filter bank feature, resulting in better recognition performance.

In this study, we only use SNR as the environment variable. Encouraged by the results, we will continue to add more environment variables in the future.

REFERENCES

[1] D. Yu, L. Deng, and G. Dahl, "Roles of pretraining and fine-tuning in context-dependent DBN-HMMs for real-world speech recognition," in *Proc. NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

[2] T. N. Sainath, B. Kingsbury, B. Ramabhadran, P. Fousek, P. Novak, and A. Mohamed, "Making deep belief networks effective for large vocabulary continuous speech recognition," in *Proc. Workshop on Automatic Speech Recognition and Understanding*, pp. 30–35, 2011.

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

[4] N. Jaitly, P. Nguyen, and V. Vanhoucke, "Application of pretrained deep neural networks to large vocabulary speech recognition", in *Proc. Interspeech*, 2012.

[5] 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: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[6] L. Deng, J. Li, J. -T. Huang et al. "Recent advances in deep learning for speech research at Microsoft," in *Proc. ICASSP*, 2013.

[7] M. L. Seltzer, D. Yu, and Y. Wang, "An investigation of deep neural networks for noise robust speech recognition," in *Proc. ICASSP*, pp. 7398–7402, 2013.

[8] B. Li and K. C. Sim, "Noise adaptive front-end normalization based on vector Taylor series for deep neural networks in robust speech recognition," in *Proc. ICASSP*, pp. 7408–7412, 2013.

[9] B. Li, Y. Tsao, and K. C. Sim, "An investigation of spectral restoration algorithms for deep neural networks based noise robust speech recognition," in *Proc. Interspeech*, pp. 3002–3006, 2013.

[10] M. Delcroix, Y. Kubo, T. Nakatani, and A. Nakamura, "Is speech enhancement pre-processing still relevant when using deep neural networks for acoustic modeling," in *Proc. Interspeech*, pp. 2992–2996, 2013.

[11] A. Narayanan and D. Wang, "Investigation of speech separation as a front-end for noise robust speech recognition," *IEEE/ACM Transactions on Audio, Speech & Language Processing*, vol. *22*, no. 4, pp. 826-835, 2014.

[12] J. Li, L. Deng, Y. Gong, and R. Haeb-Umbach, "An overview of noise-robust automatic speech recognition, in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2014.

[13] M. Seltzer, "Robustness is dead! Long live robustness!" in *Reverb Challenge Workshop*, 2014.

[14] R. Zhao, J. Li, and Y. Gong, "Variable-component deep neural network for robust speech recognition," In *Proc. Interspeech*, 2014.

[15] X. Cui and Y. Gong, "A study of variable-parameter Gaussian mixture hidden Markov modeling for noisy speech recognition," *IEEE Trans. on Audio, Speech and Language Processing*, vol. 15, no. 4, pp. 1366–1376, 2007.

[16] N. Parihar and J. Picone, "Aurora working group: DSR front end LVCSR evaluation AU/384/02," *Tech. Rep.*, Institute for Signal and Information Processing, Mississippi State Univ., 2002.

[17] H. Hermansky, D. P. W. Ellis, and S. Sharma, "Tandem connectionist feature extraction for conventional HMM systems," in *Proc. ICASSP*, vol. 3, pp. 1635–1638, 2000.

[18] F. Grezl, M. Karafiat, S. Kontar, and J. Cernocky, "Probabilistic and bottle-neck features for LVCSR of meetings," in *Proc. ICASSP*, vol. IV, pp. 757–760, 2007.

[19] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. 11, 2008.

[20] D. Yu, M. Seltzer, J. Li, J. T. Huang, and F. Seide, "Feature learning in deep neural networks–studies on speech recognition tasks," in *Proc. International Conference on Learning Representations*, 2013.