RECURRENT DEEP NEURAL NETWORKS FOR ROBUST SPEECH RECOGNITION

Chao Weng[1] *, Dong Yu*[2]*, Shinji Watanabe*[3]*, Biing-Hwang (Fred) Juang*[1]

[1] Georgia Institute of Technology, Atlanta, GA, USA
[2] Microsoft Research, One Microsoft Way, Redmond, WA, USA
[3] Mitsubishi Electric Research Laboratories, Cambridge, MA, USA
[1]{chao.weng,juang}@ece.gatech.edu, [2]dongyu@microsoft.com, [3]watanabe@merl.com

ABSTRACT

In this work, we propose recurrent deep neural networks (DNNs) for robust automatic speech recognition (ASR). Full recurrent connections are added to certain hidden layer of a conventional feedforward DNN and allow the model to capture the temporal dependency in deep representations. A new backpropagation through time (BPTT) algorithm is introduced to make the minibatch stochastic gradient descent (SGD) on the proposed recurrent DNNs more efficient and effective. We evaluate the proposed recurrent DNN architecture under the hybrid setup on both the 2^{nd} CHiME challenge (track 2) and Aurora-4 tasks. Experimental results on the CHiME challenge data show that the proposed system can obtain consistent 7% relative WER improvements over the DNN systems, achieving state-of-the-art performance without front-end preprocessing, speaker adaptive training or multiple decoding passes. For the experiments on Aurora-4, the proposed system achieves 4% relative WER improvement over a strong DNN baseline system.

Index Terms— DNN, RNN, robust ASR, CHiME, Aurora-4

1. INTRODUCTION

Improving environmental robustness of automatic speech recognition (ASR) systems has been studied for decades. To deal with the mismatched acoustical conditions between training and testing, feature space compensation approaches typically involve removing additive noise and channel distortions using speech enhancement techniques [1] such as spectral subtraction, Weiner filtering and MMSE estimators [2, 3, 4]. Other researchers explored use of noise resistant features [5, 6] or feature transformations [7, 8]. Model adaptation methods attempt to achieve compensation by adapting the models to the noisy condition. The most straightforward way is using the multi-style training strategy [9] to train models on the multi-condition data that includes different acoustical conditions of the test data. Other model space adaptation methods include parallel model combination (PMC), data-driven PMC [10] and vector Taylor series (VTS) based compensation [11, 12, 13]. The combination of both feature space and model space compensation techniques usually offer the state-of-the-art environmental robustness for an ASR system.

Recently, deep neural network (DNN) based acoustic models have been introduced for LVCSR [14, 15] tasks and show its great success in both Tandem [16] and hybrid DNN-HMM systems [17]. This opens new possibilities for further improving the noise robustness of ASR systems. In [18] and [19], it is shown that DNN based systems have remarkable robustness to environment distortions and the authors can achieve state-of-the-art performance on Aurora-4 benchmark without multiple decoding passes and model adaptation. Meanwhile, recurrent neural networks (RNNs) have been also explored for robust ASR in [20, 21, 22, 23]. However, the authors only investigated RNNs in the Tandem setup or used it as a front-end denoiser and reported results on a small vocabulary task. Few if any have explored the RNNs combined with deep structure in the hybrid setup and report results on larger tasks where the language model (LM) matters during decoding.

In this work, we investigate the RNNs with deep architecture in hybrid systems for robust ASR. Specifically, we add full recurrent connections to certain hidden layer of a feedforward DNN to allow the model to capture the temporal dependency in deep representations. A new backpropagation through time (BPTT) algorithm for updating the parameters of the recurrent layer is introduced to make the minibatch stochastic gradient descent (SGD) on the proposed recurrent DNN more efficient and effective. We evaluate the proposed recurrent DNN architecture under the hybrid setup on both the 2^{nd} CHiME challenge (track 2) [24] and Aurora-4 tasks. Experimental results on the CHiME challenge data show that we can obtain consistent 7% relative WER improvements over DNN systems, achieving the state-of-the-art performance reported in [25] without front-end preprocessing, speaker adaptive training and multiple decoding passes. For the experiments on Aurora-4, the proposed system achieves 4% relative WER improvement over a strong DNN baseline system.

The remainder of the paper is organized as follows. In Section 2, we review the DNN-HMM hybrid system and describe the architecture of the recurrent DNN. A new backpropagation through time algorithm for the recurrent layer and minibatch SGD on the whole network will be elaborated in Section 3. We report our experimental results in Section 4 and conclude our work in Section 5.

2. RECURRENT DNN ARCHITECTURE

2.1. Hybrid DNN-HMM System

In a conventional GMM-HMM LVCSR system, the state emission log-likelihood of the observation feature vector $\mathbf{o}_t$ for certain tied state or senone s_j of HMMs is generated using,

$$\log p(\mathbf{o}_t|s_j) = \log \sum_{m=1}^{M} \pi_{jm} \mathcal{N}_{jm}(\mathbf{o}_t|s_j), \tag{1}$$

where M is the number of Gaussian mixtures in the GMM for state j and π_{jm} is the mixing weight. As the outputs from DNNs represent the state posteriors $p(s_j|\mathbf{o}_t)$, a DNN-HMM hybrid system [15] uses pseudo log-likelihood as the state emissions,

$$\log p(\mathbf{o}_t|s_j) \propto \log p(s_j|\mathbf{o}_t) - \log p(s_j), \tag{2}$$

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

where the state priors $\log p(s_j)$ can be estimated using the state alignments on the training speech data. The input features vectors $\mathbf{o}_t$ to the first layer of DNNs usually use a context of l frames [15], *e.g.* $l = 9$ or $l = 11$.

2.2. Recurrent Deep Architecture

The architecture of recurrent DNN we use is shown in Fig.1. The fundamental structure is a feedforward DNN but with certain hidden layer having full recurrent connections with itself (In the Fig.1, the third hidden layer from the input layer has recurrent property). The values corresponding to those neurons at the feedforward hidden layers can be expressed as,

$$\mathbf{x}^i = \begin{cases} W_1\mathbf{x}^0 + \mathbf{b}_1, & i = 1 \\ W_i\mathbf{y}^{i-1} + \mathbf{b}_i, & i > 1 \end{cases}, \tag{3}$$

$$\mathbf{y}^i = \begin{cases} \mathrm{sigmoid}(\mathbf{x}^i) & i < n \\ \mathrm{softmax}(\mathbf{x}^i) & i = n \end{cases}, \tag{4}$$

where n is the total number of the feedforward hidden layers and both the $\mathrm{sigmoid}$ and $\mathrm{softmax}$ functions are element-wise operations. The vector $\mathbf{x}^i$ corresponds to pre-nonlinearity activations except that $\mathbf{x}^0$ is the input feature vector and $\mathbf{y}^i$ is the neuron vector at the i^{th} hidden layer. For the recurrent hidden layer, denote by $\mathbf{x}_t^i$ and $\mathbf{y}_t^i$ the pre-nonlinearity activation vector and neuron vector at frame t, the value of neuron vector at the i^{th} hidden layer is given by,

$$\mathbf{x}_t^i = W_{ii}\mathbf{y}_{t-1}^i + \mathbf{b}_{ii} + W_i\mathbf{y}_t^{i-1} + \mathbf{b}_i \tag{5}$$

$$\mathbf{y}_t^i = \mathrm{sigmoid}(\mathbf{x}_t^i), \tag{6}$$

where W_{ii} and $\mathbf{b}_{ii}$ are the recurrent weight matrix and bias vector.

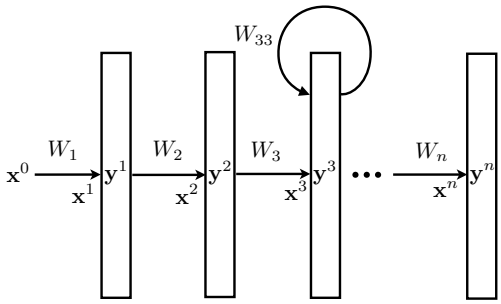

Fig. 1. Recurrent DNNs architecture: the third layer from the input layer is the recurrent hidden layer with the parameters W_{33}, note that the bias terms are omitted for simplicity.

3. BACKPROPAGATION ON THE RECURRENT DNN

3.1. Backpropagation on the Feedforward Layers

For convenience, we will use the notations as shown in Fig.1. Taking partial derivatives of the loss objective function with respect to the pre-nonlinearity activations of output layer ($\mathbf{x}^n$ in the Fig.1) will give us the error vector to be backpropagated to the previous hidden layers. The negative cross-entropy is commonly used loss function. The loss functions based on discriminative training criteria such as sMBR [26], MMI and MPE/MWE [27] have also been used for ASR. When various loss functions are used, the only difference reflected in the backpropagation lies in the error vector we backpropagate to the previous hidden layers. If we use the negative cross-entropy loss and let $\mathcal{X}$ be the whole training set which contains N frames, *i.e.* $\mathbf{x}_{1:N}^0 \in \mathcal{X}$, then the loss associated with $\mathcal{X}$ is given by,

$$\mathcal{L}_{1:N} = -\sum_{t=1}^{N}\sum_{j=1}^{J}\mathbf{d}_t(j)\log\mathbf{y}_t^n(j), \tag{7}$$

and $\mathbf{d}_t(j)$ is the j^{th} element of the label vector at frame t, then the error vector to be backpropagated to the previous layers is given by,

$$\boldsymbol{\epsilon}_t^n = \frac{\partial\mathcal{L}_{1:N}}{\partial\mathbf{x}^n} = \mathbf{y}_t^n - \mathbf{d}_t, \tag{8}$$

the backpropagated error vectors at previous hidden layer are thus,

$$\boldsymbol{\epsilon}_t^i = W_{i+1}^T\boldsymbol{\epsilon}_t^{i+1} * \mathbf{y}^i * \left(\mathbf{1} - \mathbf{y}^i\right), i < n \tag{9}$$

where $*$ denotes element-wise multiplication. With the error vectors at certain hidden layers, the gradient over the whole training set with respect to the weight matrix W_i is given by,

$$\frac{\partial\mathcal{L}_{1:N}}{\partial W_i} = \mathbf{y}_{1:N}^{i-1}(\boldsymbol{\epsilon}_{1:N}^i)^T, \tag{10}$$

note that in above equation, both $\mathbf{y}_{1:N}^{i-1}$ and $\boldsymbol{\epsilon}_{1:N}^i$ are matrices, which is formed by concatenating vectors corresponding to all the training frames from frame 1 to N, *i.e.* $\boldsymbol{\epsilon}_{1:N}^i = [\boldsymbol{\epsilon}_1^i, \ldots, \boldsymbol{\epsilon}_t^i, \ldots, \boldsymbol{\epsilon}_N^i]$. The batch gradient descent updates the parameters with the gradient in (10) only once after each sweep through the whole training set and in this way parallelization can be easily conducted to speedup the learning process. However, SGD usually works better in practice where the true gradient is approximated by the gradient at a single frame t, *i.e.* $\mathbf{y}_t^{i-1}(\boldsymbol{\epsilon}_t^i)^T$, and the parameters are updated right after seeing each frame. The compromise between the two, minibatch SGD, is more widely used, as the reasonable size of minibatches makes all the matrices fit into GPU memory, which leads to a more computationally efficient learning process.

3.2. BPTT on the Recurrent Layer

BPTT updates the recurrent weights by unfolding the networks in time. As shown in Fig.2, the standard error BPTT over a minibatch $\mathbf{x}_{1:M}^0 \in \mathcal{X}$ is given by,

$$\boldsymbol{\epsilon}_t^i = \begin{cases} (W_{i+1}^T\boldsymbol{\epsilon}_t^{i+1}) * \mathbf{y}_t^i * \left(\mathbf{1} - \mathbf{y}_t^i\right), & t = M \\ (W_{i+1}^T\boldsymbol{\epsilon}_t^{i+1} + W_{ii}^T\boldsymbol{\epsilon}_{t+1}^i) * \mathbf{y}_t^i * \left(\mathbf{1} - \mathbf{y}_t^i\right), & t < M \end{cases}, \tag{11}$$

where M is the size of minibatch. Note that the evaluation of the exact gradients with respect to the recurrent weight matrix W_{ii} needs the corresponding error vectors backpropagated through time to the first frame of the current training speech utterance, but in practice the gradients with respect to the recurrent weight matrix W_{ii} over the minibatch are usually approximated by truncating the BPTT process within the corresponding minibatch. In (11), each time step of BPTT needs both the error vector from next frame and the one from next hidden layer which forces us to backpropagate the error vector frame by frame rather than in a minibatch mode. In the standard minibatch BPTT, a key observation is that error signals backpropagated from the next hidden layer (*i.e.*, $\epsilon_{t-1}^{i+1}, \epsilon_t^{i+1}, \epsilon_{t+1}^{i+1}$ as in Fig.2) will be backpropagated in different time steps which indicates each training frame within a minibatch will make non-uniform contribution to the final minibatch gradient. Thus, we introduce the *truncated*

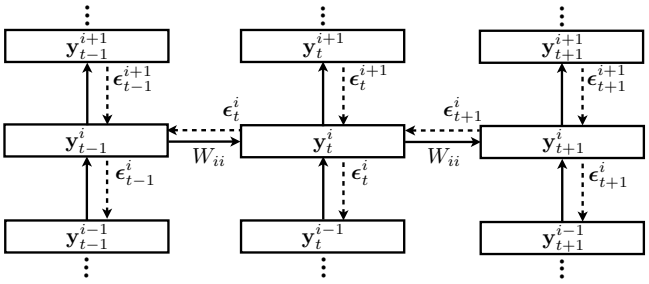

Fig. 2. Backpropagation through time for i^{th} recurrent layer's parameter W_{ii}: the solid lines denote the directions of forward propagation and the dotted lines denote the directions of backpropagation.

minibatch BPTT where for each individual online gradient, we truncated the BPTT process in fixed time steps,

$$\frac{\partial \mathcal{L}_{1:M}}{\partial W_{ii}} = \sum_{t=1}^{M} \frac{\partial \mathcal{L}_t}{\partial W_{ii}} \approx \sum_{t=1}^{M} \sum_{\tau=1}^{\mathrm{T}} \mathbf{y}_{t-\tau}^{i} (\boldsymbol{\epsilon}_{t-\tau+1}^{i})^T, \qquad (12)$$

where $\frac{\partial \mathcal{L}_t}{\partial W_{ii}}$ is the online gradient at frame t, while for each individual online gradient, we backpropagate for multiple time steps as in [28], *e.g.* $\mathrm{T} = 4$ or $\mathrm{T} = 5$,

$$\boldsymbol{\epsilon}_{t-1}^{i} = W_{ii}^{T} \boldsymbol{\epsilon}_{t}^{i} * \mathbf{y}_{t-1}^{i} * \left(\mathbf{1} - \mathbf{y}_{t-1}^{i}\right). \qquad (13)$$

Another benefit of the introduced truncated minibatch BPTT is that after we exchange the summation order (see below), the BPTT on the recurrent weights can be conducted in a minibatch mode,

$$\begin{aligned} \frac{\partial \mathcal{L}_{1:M}}{\partial W_{ii}} &= \sum_{t=1}^{M} \frac{\partial \mathcal{L}_t}{\partial W_{ii}} \approx \sum_{t=1}^{M} \sum_{\tau=1}^{\mathrm{T}} \mathbf{y}_{t-\tau}^{i} (\boldsymbol{\epsilon}_{t-\tau+1}^{i})^T \\ &= \sum_{\tau=1}^{\mathrm{T}} \sum_{t=1}^{M} \mathbf{y}_{t-\tau}^{i} (\boldsymbol{\epsilon}_{t-\tau+1}^{i})^T \\ &= \sum_{\tau=1}^{\mathrm{T}} \underbrace{\mathbf{y}_{1-\tau:M-\tau}^{i} (\boldsymbol{\epsilon}_{1-\tau+1:M-\tau+1}^{i})^T}_{\text{minibatch gradient}}, \end{aligned} \qquad (14)$$

note that in above equations the vectors with negative indices come from the corresponding ones in previous minibatch. Therefore, the gradients for updating the recurrent weights can also be calculated in a minibatch mode using matrix multiplication which can be considerably speeded up using GPU.

4. EXPERIMENTS

4.1. Experiments on CHiME challenge data

Track 2 of the 2^{nd} CHiME challenge [24] is a medium vocabulary (5k) task under reverberated and noisy environment. There are three sets of data, clean, reverberated, isolated (reverberated and noisy). All the clean speech utterances are extracted from WSJ0 database. The reverberated speech utterances are generated by convolving clean speech with time-varying binaural room impulse responses. Noise backgrounds including concurrent speakers, TV, game console, footsteps, and distant noise from outside or from the kitchen are first recorded and the isolated speech utterances are created by selecting appropriate pre-recorded noise background excerpts, mixing them to reverberated speech utterances to obtain the speech signals with the SNR of -6, -3, 0, 3, 6, and 9 dB without rescaling. The multi-condition training set contains 7138 speech utterances (reverberated and noisy version of SI-84) with SNR from -6 to 9 dB. The development set contains 2460 multi-condition speech utterances and evaluation set contains 1980 (reverberated and noisy version of NOV-92, *i.e.* 330×6) utterances with uniform number for each condition.

Systems	Conditions						Avg.
	9dB	6dB	3dB	0dB	-3dB	-6dB	
GMM	29.87	36.26	43.25	54.64	61.63	69.51	49.19
DNN I	19.19	23.41	28.17	36.56	45.99	56.81	35.02
DNN II	17.88	21.58	24.83	33.42	40.91	52.06	31.78
DNN III	**16.85**	**20.25**	23.05	30.81	39.59	50.70	30.21
DNN IV	16.89	20.29	**22.83**	**30.36**	**39.49**	**49.47**	**29.89**

Table 1. WERs (%) of baseline GMM-HMM, and DNN-HMM systems on CHiME challenge data, DNN I∼IV systems correspond to the iteratively retrained DNNs with the new alignments

We first build a GMM-HMM system using Kaldi toolkit [29] for the task: 2008 distinct tied-state GMMs are trained with MFCC features coupled with their linear discriminant analysis (LDA) and maximum likelihood linear transform (MLLT) on the 7138 multi-condition speech utterances and feature-space maximum likelihood linear regression (fMLLR) for speaker adaptation during later iterations. For the DNN-HMM systems, we first do generative pre-training using RBMs, and stacking them together in the end to initialize the DNN with 7 2048-dim hidden layers. With the alignments obtained from the GMM-HMM system, we train DNN I system with 40-dim log Mel filter-bank features. We use 256 minibatch and 0.008 as the initial learning rate. After each epoch of training, we validate the frame accuracy on the development set, shrink the learning rate by 0.5 when the improvements are less than 0.5% and stop training when the improvements are less than 0.1%. With the trained DNN I system, we do the realignments and train a second DNN system DNN II using the new alignments. We repeat this process until the performance gain from the realignments become saturated. The standard 5k tri-gram language models are used for the decoding. The WER results are listed in Table 1. As can be seen, all DNN systems achieve significant gains over GMM-HMM system, the performance gain from the realignments saturated until the fourth system is trained and the best realigned DNN IV system obtains 29.89% WER, which will be the baseline system we use for the comparison with recurrent DNN system.

Then we build the proposed recurrent DNN-HMM systems. For the comparisons, the alignments used to train all the recurrent DNN system are the same as the DNN IV systems. We initialize recurrent DNN parameters as follows: for the feedforward layers, we just copy the weights from the DNN trained after 5 epochs in DNN IV systems (in our experiment, $15 \sim 17$ iterations are needed to reach convergence. This is for speeding up the training, in the end, the total epochs are almost the same); the recurrent parameters are initialized with randomization. We use 256 minibatch for SGD, and 0.004 for the initial learning rate. The learning rate scheduling and stop criteria are the same as DNN training as described earlier. We try 5 different setups in our recurrent DNN experiments: RDNN system corresponds to the recurrent DNN with the recurrent units at the 4^{th} hidden layer from the input layer using standard minibatch BPTT. RDNN I∼IV systems correspond to the recurrent DNN with the recurrent units at the 2^{nd}, 3^{rd}, 4^{th} and 5^{th} hidden layer from the input layer using the introduced truncated minibatch BPTT as described in Section 3.2. As shown in Table 2, with the stan-

Systems	Conditions						Avg.
	9dB	6dB	3dB	0dB	-3dB	-6dB	
DNN IV	16.89	20.29	22.83	30.36	39.49	49.47	29.89
RDNN	17.26	20.29	22.83	30.21	38.67	48.98	29.71
RDNN I	15.92	18.59	21.41	28.28	**35.81**	47.23	27.87
RDNN II	15.95	18.49	**20.87**	**27.89**	36.65	**46.35**	**27.70**
RDNN III	16.22	18.49	21.24	28.04	36.26	46.46	27.79
RDNN IV	**15.84**	**18.16**	21.03	28.21	36.93	47.17	27.89

Table 2. WERs (%) of best DNN-HMM system and five recurrent DNN-HMM systems trained on CHiME challenge multi-condition data: RDNN system corresponds to the recurrent DNN with the recurrent units at 4^{th} hidden layer using standard minibatch BPTT. RDNN I∼IV systems correspond to the recurrent DNN with the recurrent units at 2^{nd}, 3^{rd}, 4^{th} and 5^{th} hidden layer from the input layer using the introduced truncated BPTT as described in Section 3.2.

Systems	Conditions						Avg.
	9dB	6dB	3dB	0dB	-3dB	-6dB	
DNN V	14.27	16.44	19.39	24.68	31.65	42.05	24.75
RDNN V	**13.60**	**14.96**	**17.92**	**22.98**	**29.07**	**38.11**	**22.77**

Table 3. WERs (%) of best DNN-HMM system and recurrent DNN-HMM systems trained on CHiME challenge multi-condition data with available stereo data.

dard minibatch BPTT, system RDNN only shows marginal WER improvements over the baseline DNN system. While with the truncated minibatch BPTT, all recurrent DNN systems significantly outperform the DNN systems in all conditions. This is very likely because the non-recurrent weights are copied from the DNN system while the recurrent weights are randomly initialized. Thus the recurrent weights are less trained if not using the proposed approach. Therefore, the truncated minibatch BPTT will be used through all following recurrent DNN experiments. The best system with recurrent hidden layer at the 3^{rd} layer obtain 27.70% WER, achieving the state-of-the-art performance[1] and 7.3% relative improvement over our best DNN system. Furthermore, we observe no significant performance difference between different setups of recurrent DNN, but the system seems working best with the architecture where the recurrent layer is located at the middle of the DNN. Finally we conduct the experiments on the dataset with the assumption that the stereo data is available. We train the similar GMM-HMM system on the clean speech data, and then using the alignments on clean data as the label to train the DNN and recurrent DNN with similar setup as described earlier. The experimental results are list in Table 3, as can be seen that recurrent DNN also obtain consistent and significant performance gain over the DNN system.

4.2. Experiments on Aurora-4

Aurora-4 is also a medium vocabulary task based on the WSJ0 corpus. The training set contains both 8kHz and 16 kHz multi-condition 7137 utterances from 83 speakers. One half of the utterances were recorded by the primary closed microphone and the other half were recorded using one of secondary open microphones. Among the whole training set, there are six different types of noise backgrounds, street traffic, train station, car, babble, restaurant, airport at $10 \sim 20$ dB SNR. The evaluation set is noisy and reverberated version of WSJ0 5K NOV-92 which consists of 4620 utterances in 14 conditions (330×14) and can be grouped into 4 subsets: clean, noisy, clean with channel distortion, noisy with channel distortion.

[1]The state-of-the-art system reported in [25] achieves 26.86% WER but with discriminatively trained LM and MBR decoding. With the tri-gram LMs, the best system reported by authors is 27.61%

Systems	Conditions				Avg.
	Clean	Noise	Channel	Channel+Noise	
GMM	8.28	13.83	17.84	29.24	20.32
DNN I	3.51	8.15	10.69	22.59	14.19
DNN II	3.34	7.48	10.09	21.76	13.49
DNN III	**3.24**	**7.44**	**10.07**	**21.43**	**13.33**

Table 4. WERs (%) of baseline GMM-HMM, and DNN-HMM systems on Aurora-4 data, DNN I∼III systems correspond to the iteratively retrained DNNs with the new alignments.

Systems	Conditions				Avg.
	Clean	Noise	Channel	Channel+Noise	
DNN III	3.34	7.44	10.07	21.43	13.33
RDNN I	3.27	7.30	9.15	20.67	12.88
RDNN II	**3.06**	**7.26**	**9.10**	**20.44**	**12.74**

Table 5. WERs (%) of best DNN-HMM system and two recurrent DNN-HMM systems trained on Aurora-4 multi-condition data: RDNN I, II systems correspond to the recurrent DNN with the recurrent units at the 3^{rd} and 4^{th} hidden layer from the input layer.

We first build the baseline GMM-HMM system for the tasks, 2026 distinct tied-state GMMs are trained with MFCC features coupled with their linear discriminant analysis (LDA) and maximum likelihood linear transform (MLLT) on the 7137 multi-condition speech utterances. For the DNN-HMM systems, the setup is similar to the one in previous experiment: we first generatively pretrain the DNN using RBMs, and stack them to initialize the DNN with 7 2048-dim hidden layers; With the alignments from GMM-HMM systems, we trained the DNN I system using 40-dim log Mel filterbank features. Then we further do the realignments using trained DNN I system and train the second DNN system with the new alignments. Then this process is repeated until there are no further significant improvements. The experimental results are shown in Table 4. The best DNN III system achieves 13.33% average WER. Following the same setup as described in the experiment on CHiME challenge data, we build two recurrent DNN systems on top of DNN III system, namely RDNN I and II systems which correspond to the recurrent DNN with the recurrent units at the 3^{rd} and 4^{th} hidden layer from the input layer. As shown in Table 5, both recurrent DNN system outperform the DNN III system in all conditions. The RDNN II system achieves 0.59% absolute improvements over our best DNN system.

5. CONCLUSIONS

In this work, we propose recurrent DNNs for robust acoustic modeling. A new BPTT algorithm is introduced to make the minibatch SGD on the proposed recurrent DNNs more efficient and effective. We evaluate the proposed recurrent DNN architecture under the hybrid setup on both 2^{nd} CHiME challenge (track 2) and Aurora-4 tasks. The experimental results on the CHiME challenge data show that we can obtain consistent 7% relative WER improvements over DNN systems, achieving the state-of-the-art performance without front-end preprocessing, speaker adaptive training or multiple decoding passes. On the Aurora-4, the proposed system obtains 4% relative WER improvement over a strong DNN baseline system.

6. REFERENCES

[1] B.-H. Juang, "Speech recognition in adverse environments," *Computer Speech and Language*, vol. 5, 1992.

[2] A. Acero, "Environmental robustness in automatic speech recognition," in *ICASSP*. 1990, IEEE.

[3] L. Deng, A. Acero, J. Li, and J. Droppo, "High-performance robust speech recognition using stereo training data," in *ICASSP*. 2001, IEEE.

[4] Dong Yu, Li Deng, Jasha Droppo, Jian Wu, Yifan Gong, and Alex Acero, "A minimum-mean-square-error noise reduction algorithm on mel-frequency cepstra for robust speech recognition.," in *ICASSP*. 2008, pp. 4041–4044, IEEE.

[5] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *J. Acoust. Soc. Am.*, vol. 57, pp. 1738–52, 1990.

[6] H. Hermansky and N. Morgan, "RASTA processing of speech," *Speech and Audio Processing, IEEE Transactions on*, vol. 2, no. 4, pp. 578–589, 1994.

[7] M. J. Hunt and C. Lefebvre, "A comparison of several acoustic representations for speech recognition with degraded and undegraded speech," in *Proc. ICASSP1989*, 1989.

[8] M.J.F. Gales, "Maximum likelihood linear transformations for hmm-based speech recognition," *Computer Speech and Language*, vol. 12, pp. 75–98, 1998.

[9] R. Lippmann, E. Martin, and D.B. Paul, "Multi-style training for robust isolated-word speech recognition," in *Proc. ICASSP1987*, 1987.

[10] M. Gales and S. J. Young, "Robust continuous speech recognition using parallel model combination," *IEEE Transactions on Speech and Audio Processing*, vol. 4, pp. 352–359, 1996.

[11] Pedro J. Moreno, *Speech Recognition in Noisy Environments*, Ph.D. thesis, ECE Department, CMU, 1996.

[12] Jinyu Li, Li Deng, Dong Yu, Yifan Gong, and Alex Acero, "High-performance hmm adaptation with joint compensation of additive and convolutive distortions via vector taylor series.," in *ASRU*. 2007, pp. 65–70, IEEE.

[13] Yong Zhao and Biing-Hwang Juang, "On noise estimation for robust speech recognition using vector taylor series," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, 2010, pp. 4290–4293.

[14] G.E. Dahl, Dong Yu, Li 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, jan. 2012.

[15] Geoffrey E. Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, 2012.

[16] Hynek Hermansky, Daniel P. W. Ellis, and Sangita Sharma, "Tandem connectionist feature extraction for conventional hmm systems," in *PROC. ICASSP*. 2000, pp. 1635–1638, IEEE.

[17] Herve A. Bourlard and Nelson Morgan, *Connectionist Speech Recognition: A Hybrid Approach*, Kluwer Academic Publishers, Norwell, MA, USA, 1993.

[18] Dong Yu, Michael L. Seltzer, Jinyu Li, Jui-Ting Huang, and Frank Seide, "Feature learning in deep neural networks - a study on speech recognition tasks," *CoRR*, vol. abs/1301.3605, 2013.

[19] D. Yu M. L. Seltzer and Y.-Q. Wang, "An investigation of deep neural networks for noise robust speech recognition," in *Proc. ICASSP2013*, 2013.

[20] Oriol Vinyals, Suman Ravuri, and Daniel Povey, "Revisiting Recurrent Neural Networks for Robust ASR," in *ICASSP*, 2012.

[21] A. Maas, Q. Le, T. O'Neil, O. Vinyals, P. Nguyen, and A. Ng, "Recurrent neural networks for noise reduction in robust asr," in *Proceedings of INTERSPEECH*, 2012.

[22] F. Weninger, J. Geiger, M. Wöllmer, B. Schuller, and G. Rigoll, "The munich 2011 chime challenge contribution: Nmf-blstm speech enhancement and recognition for reverberated multisource environments," in *Proc. Machine Listening in Multisource Environments (CHiME 2011), satellite workshop of Interspeech 2011, ISCA, Florence, Italy*, 2011.

[23] A. Graves, A. Mohamed, and G. Hinton., "Speech recognition with deep recurrent neural networks," in *Acoustics Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2013.

[24] Emmanuel Vincent, Jon Barker, Shinji Watanabe, Jonathan Le Roux, Francesco Nesta, and Marco Matassoni, "The second 'CHiME' Speech Separation and Recognition Challenge: Datasets, tasks and baselines," in *ICASSP*, Vancouver, Canada, 2013.

[25] Yuuki Tachioka, Shinji Watanabe, Jonathan Le Roux, and John R. Hershey, "Discriminative methods for noise robust speech recognition: A CHiME challenge benchmark," in *Proceedings of the CHiME 2013 International Workshop on Machine Listening in Multisource Environments*, 2013.

[26] Brian Kingsbury, "Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling," in *ICASSP*, 2009, pp. 3761–3764.

[27] K. Vesely, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks," in *Proceedings of INTERSPEECH*, 2013.

[28] Tom Mikolov, *Statistical Language Models Based on Neural Networks*, Ph.D. thesis, Brno university of technology, 2012.

[29] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely, "The Kaldi speech recognition toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. Dec. 2011, IEEE Signal Processing Society.