*INTERSPEECH 2017*
August 20–24, 2017, Stockholm, Sweden

# Unfolded deep recurrent convolutional neural network with jump ahead connections for acoustic modeling

*Dung T. Tran*[1], *Marc Delcroix*[1], *Shigeki Karita*[1], *Michael Hentschel*[1,2], *Atsunori Ogawa*[1], *Tomohiro Nakatani*[1]

[1]NTT Communication Science Laboratories, NTT corporation,
2-4, Hikaridai, Seika-cho (Keihana Science City), Soraku-gun, Kyoto 619-0237 Japan
[2]Nara Institute of Science and Technology, 8916-5 Takayama, Ikoma, Nara, Japan
{dung.tran,marc.delcroix,ogawa.atsunori,nakatani.tomohiro}@lab.ntt.co.jp

## Abstract

Recurrent neural networks (RNNs) with jump ahead connections have been used in the computer vision tasks. Still, they have not been investigated well for automatic speech recognition (ASR) tasks. In other words, unfolded RNN has been shown to be an effective model for acoustic modeling tasks. This paper investigates how to elaborate a sophisticated unfolded deep RNN architecture in which recurrent connections use a convolutional neural network (CNN) to model a short-term dependence between hidden states. In this study, our unfolded RNN architecture is a CNN that process a sequence of input features sequentially. Each time step, the CNN inputs a small block of the input features and the output of the hidden layer from the preceding block in order to compute the output of its hidden layer. In addition, by exploiting either one or multiple jump ahead connections between time steps, our network can learn long-term dependencies more effectively. We carried experiments on the CHiME 3 task showing the effectiveness of our proposed approach.

**Index Terms**: Acoustic modeling, RNN, jump ahead connection automatic speech recognition

## 1. Introduction

Deep neural networks (DNNs) have become an essential part of automatic speech recognition (ASR) systems [1]. Although DNNs dominate the Gaussian Mixture Model/Hidden Markov Model (GMM/HMM), they have limited capacities to model sequences of speech features. Recurrent neural networks can model speech dynamics more explicitly and have recently exhibited superior performance to their DNN counterparts [2, 3, 4, 5, 6, 7].

Unlike DNNs, RNNs use a recurrent connection which has the ability to model the interdependence between the features frames over time. Essentially, unfolded RNN is an RNN model where the network is unfolded over time [2, 3]. Unlike a conventional RNN, an unfolded RNN only sees a limited context within a context window (eg. 11 frames) and can thus be trained using frame randomization. The recurrent connections can be used at the input layer [4, 2, 7] or at deeper layers [5, 6, 8]. In [5, 6, 8], a simple DNN or convolutional neural network (CNN) is used at the input layer to create a better representation of the input features. Therefore, the DNNs and the CNNs act in the same way as feature extraction. Then, one or two RNNs are built on top of the DNNs or the CNNs to capture the dependencies of the sequences of the high-level features. Such approaches have been shown to be more effective than those using recurrent connections at the input layer. RNN networks are usually truncated to a few time steps (i.e. 4,5 time steps) due to vanishing gradient problems. One way to mitigate this vanishing gradient is to use a more sophisticated network such as Long-Short Term Memory (LSTM) network [9, 10, 8, 11]. Beside of using LTSM networks, there have also been studies designed to alleviate the vanishing gradient problems [12, 13, 14, 15, 16, 17] for RNNs. For example, [12, 13] introduced so-called jump ahead connections or skip connections, which are realized by concatenating the input features with the past output of the network. By employing the past outputs far from the current frame, the network can model the long-term dependencies. Such approaches might have limitations as regards learning multiple levels of dependencies. This is because increasing the number of jump ahead connections from time steps in history to the same time step might significantly increase the network complexity. In addition to [12, 13], there have also been some extensions of the LSTMs network to mitigate the vanishing gradient problems by using LSTM network in combination with highway connections [14, 15].

In this paper, we propose a different approach that helps unfolded RNN networks to learn long-term dependencies more effectively and apply it to an acoustic modeling task. In our RNN architecture, a CNN processes a sequence of acoustic input features sequentially within the context window. First, the sequence of acoustic input features is arranged into a limited number of overlapping frame blocks. Second, a current frame block and the output of the hidden layer for the preceding block are fed into the input layer of the CNN to generate a new output for its hidden layer. At the last time step within the context window, the output of the CNN which processes the last frame block predicts the state posterior probabilities. In our unfolded RNN network, instead of using a shallow DNN, the CNN is designed inside the recurrent connection with the hope that it can model the short-term dependency behaviors more effectively. In addition, by exploiting either one or multiple jump ahead connections between time steps, the network can help the unfolded RNN to alleviate the vanishing gradient problem so that it can learn long-term dependencies more effectively. To handle the output from the hidden layers for the preceding blocks, we used an adding operation rather than a concatenating operation [12, 13]. The use of the adding operations potentially keeps the network architecture simpler when learning multiple levels of dependencies. This is the first attempt to use an unfolded RNN in combination with jump ahead connections for acoustic modeling tasks.

In the remainder of the paper, we revisit the unfolded RNN and jump ahead connection concept in Section 2. In Section 3, we present our unfolded recurrent convolutional neural network with jump ahead connections. Section 5 describes our experiments and analyzes the results. Section 6 provides our conclusion and presents some potential future research directions.

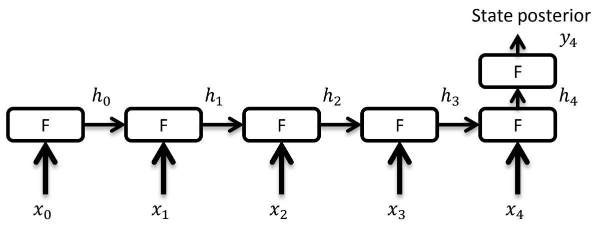

Figure 1: *Unfolded vanilla recurrent neural network*

## 2. Unfolded recurrent neural network and jump ahead connections

### 2.1. Unfolded recurrent neural network

The RNN architecture has the ability to remember its history by introducing a connection between two consecutive hidden states. The RNN's formulation is given by:

$$\mathbf{h}_t = sigmoid(\mathbf{W}_{hx}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1}) \quad (1)$$
$$\mathbf{y}_t = softmax(\mathbf{W}_{yh}\mathbf{h}_t) \quad (2)$$

where $\mathbf{x}_t$ and $\mathbf{h}_t$ are an input feature and a hidden state of the RNN at time step $t$, respectively. Note that, $\mathbf{W}_{hx}$, $\mathbf{W}_{hh}$ and $\mathbf{W}_{yh}$ are an input-hidden matrix, an hidden-hidden matrix and an hidden-output matrix, respectively. Note that the recurrent connection of this simple RNN uses only one hidden-to-hidden matrix. In its unfolded form, an RNN is expended to process only the speech frames within the input context window as shown in Fig. 1. Note that with unfolded RNN, $\mathbf{x}_t$ is a block of consecutive frames. The parameters of an RNN are trained so that they minimize the loss function $\mathbf{J}$ between the output of the network immediately after the soft-max layer and the desired output. This is done by using a stochastic gradient descent (SGD) approach, which requires us to compute the derivation of the loss function with respect to the parameters. The derivative of the loss function $\mathbf{J}$ with respect to $\mathbf{W}_{hx}$ is given by:

$$\frac{\partial \mathbf{J}}{\partial \mathbf{W}_{hx}} = \sum_{\tau=0}^{T} \frac{\partial \mathbf{J}}{\partial \mathbf{y}_t}\frac{\partial \mathbf{y}_t}{\partial \mathbf{h}_t}\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} \cdots \frac{\partial \mathbf{h}_{t-\tau}}{\partial \mathbf{W}_{hx}}. \quad (3)$$

where $T$ represents the truncated time steps. When $T$ is too small or too large, the RNN has trouble learning the long-term dependencies [16]. For example, if $T$ is large, it makes gradient summation in Eq. 3 go to zero exponentially, which prohibits the learning of long-term dependencies.

### 2.2. Recurrent neural network with jump ahead connections

With a conventional RNN, jump ahead or skip connections create a shortcut connection from a time step far back in the history to the current time step [17, 16]. The operations of a conventional RNN can thus be written as

$$\mathbf{h}_t = sigmoid(\mathbf{W}_{hx}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{sc}\mathbf{h}_{t-\tau}) \quad (4)$$
$$\mathbf{y}_t = softmax(\mathbf{W}_{yh}\mathbf{h}_t) \quad (5)$$

where $\tau$ is a certain time step in the past. Compared with conventional RNNs, RNNs with a jump ahead connection introduce only one more jump ahead matrix $\mathbf{W}_{sc}$. The recurrent connection of this RNN architecture uses one hidden-to-hidden matrix and one jump ahead matrix. The derivative of the loss function $\mathbf{J}$ with respect to $\mathbf{W}_{hx}$ is given by:

$$\frac{\partial \mathbf{J}}{\partial \mathbf{W}_{hx}} = \left(\sum_{\tau=0}^{T} \frac{\partial \mathbf{J}}{\partial \mathbf{y}_t}\frac{\partial \mathbf{y}_t}{\partial \mathbf{h}_t}\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} \cdots \frac{\partial \mathbf{h}_{t-\tau}}{\partial \mathbf{W}_{hx}}\right) + \frac{\partial \mathbf{J}}{\partial \mathbf{y}_t}\frac{\partial \mathbf{y}_t}{\partial \mathbf{h}_t}\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-\tau}}\frac{\partial \mathbf{h}_{t-\tau}}{\partial \mathbf{W}_{hx}} \quad (6)$$

The total gradient term $\frac{\partial \mathbf{J}}{\partial \mathbf{W}_{hx}}$ is unlikely to suffer from the vanishing gradient problem because even for a large time step $T$ the second gradient term is unlikely to vanish. To the best of our knowledge, jump ahead connections have not been used with an unfolded RNN for acoustic modeling tasks.

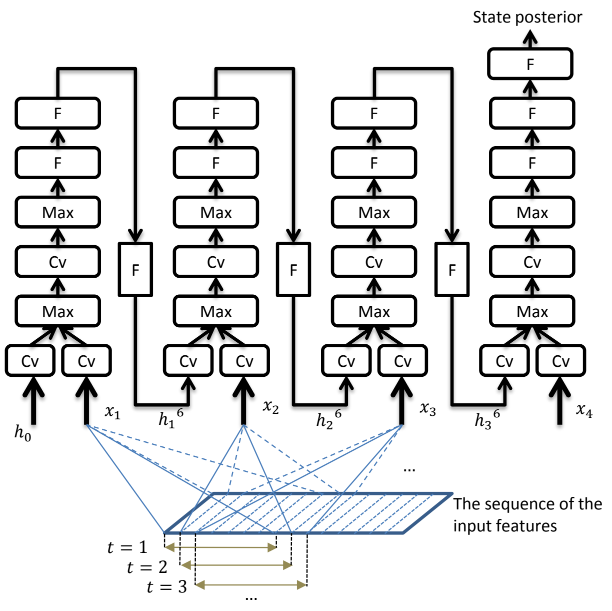

Figure 2: *Unfolded recurrent convolutional neural network (URCNN). The $Cv, Max, F, C$ mean a convolutional, a max-pooling, a fully connected layers and connection layer, respectively*

## 3. Proposed approach

### 3.1. Unfolded deep recurrent convolutional neural network

Our unfolded deep recurrent convolutional neural network (UDRCNN) is inspired by the unfolded recurrent neural network [2]. Fig. 2 shows the architecture of the network. Note that sequence of the input features in Fig. 2 is the context window (e.g. 11 frames). The sequence of the input features is divided into $T + 1$ overlapping frame blocks with a smaller number of frames $M$ ($M \ll T$) where the next frame block is shifted 1 frame to the right of the current frame block. Frame blocks are denoted as $x_0, x_1, x_2...x_T$.

The CNN has $K = 7$ layers, i.e. 2 conv layers each followed by a max pooling layer, and 3 fully connected layers on top of the last max pooling layer. The CNN travels thought the input speech features within the context window and it processes the sequence of the input speech features sequentially. At each time step $t$, the CNN inputs a block of $M$ frames (e.g. $x_t$) and the output of the hidden layer for the preceding block at time step $t - 1$ to compute the new output of its hidden layer. The first convolutional layer has two convolutional operations *conv*, one to process the block of speech features and one to process the past hidden states such as:

$$h_t^1 = \sigma(conv(x_t) + conv(g(h_{t-1}^{K-1}))) \quad (7)$$

where $h_t^k$ is the output of the $k^{th}$ hidden layer at time step $t$, $\sigma$ denotes a sigmoid non-linear activation function, and $g$ consists of a linear transformation followed by a sigmoid activation function. $g$ is used here to reduce the dimension of the output of the hidden layer $h_{t-1}^{K-1}$. One connection layer is used to feed the output of the last hidden layer $h_{t-1}^{K-1}$ back to the first convolutional layer of the CNN. The output vector of the connection layer is reformed so that it has a similar shape to one block of speech features. The output of the last block of the context window predicts the state posterior probabilities.

In our architecture, by using this more complex and deeper network such as the CNN instead of just a single matrix, we hope that it will provide a better model of the short-term dependencies behavior. Note that we use dropout during training for all fully connected layers.

### 3.2. UDRCNN with jump ahead connections

To process a sequence of input features that has a longer context window, and thus a larger number of frame blocks, we use UDRCNN in combination with jump ahead connections. To exploit longer term dependencies, more convolutional operations are used at the first convolutional layer where each convolutional operation is associated with a specific time step in the history. Therefore, the first convolutional layer becomes:

$$h_t^1 = \sigma(conv(x_t) + \sum_{\tau \subset \Lambda, \tau < t} conv(g(h_\tau^{K-1}))) \quad (8)$$

where $\Lambda$ denotes a set of associated time steps $\tau$. Here we consider only past recurrent connections, but the same principle could be used with future connections in a similar way as for bi-directionnal RNNs. $g$ is used here to reduce the dimension purpose.

## 4. Relation to prior works

Our method is similar to that reported in [3]. However, while [3] uses system combination to combine the output of RNN and CNN systems, our approach investigates a more sophisticated network combination, which allows us to jointly train the entire system by using the SGD approach. In addition, by using jump ahead connections, our network can be trained to directly model a larger context as shown in the Section 5.8.2.

In [8], a CNN was also used with RNNs, but the CNN was simply used as a feature extraction, whereas we introduce the recurrent connection within the CNN. In addition, by using jump ahead connections, our network is potentially more effective if we use a larger context.

Although the two groups functioned independently, we found that our work exhibited certain similarities with [18] since both exploit jump ahead connections for RNN based acoustic modeling. But our work differs from [18] in its use of jump ahead connections. In our network, jump ahead connections are used to connect between time steps whereas this is not the case in [18] does not. In [18], jump ahead connections are used to connect a hidden recurrent layer to the output layer. Therefore, our work might be somewhat more effective at learning long-term dependencies.

This work sharing some similarity with the works reported in [19, 20, 21]. The main difference from [19, 20, 21] is that we use the complex CNN to model the temporal dependency instead of using simply a hidden-hidden transition matrix.

## 5. Experiments

We performed experiments using the CHiME-3 corpus [22], which consists of real speech recordings collected in four different environments, i.e. cafe (CAF), street junction (STR), public transport (BUS), and pedestrian area (PED). The corpus also includes simulated training and test data sets. In this study, we used the real data for evaluation. The corpus consists of read speech, where the prompts are taken from the WSJ0 corpus. The training set comprises 1600 real and 7138 simulated utterances, which amounts to 18 hours of speech. The development and evaluation sets for the real recordings consist of 1640 and 1320 utterances, respectively, spoken by four different speakers.

These features are extracted every 10-msec by using a 25-msec sliding window. Each speech features consist of 40 log mel filterbank coefficients appended with $\Delta$ and $\Delta\Delta$ coefficients. The sequence of input speech features has 11 frames. The speech features were processed with utterance level cepstral mean normalization, and further normalized using mean and variance normalization parameters calculated on the training data. A soft-max layer which has 5976 output units is used to compute the HMM state posteriors. The acoustic model was trained using a mini-batch SGD approach to minimize the cross entropy criterion $\mathbf{J}$. Only one channel of audio data from channel 5 is used to train the model. Network parameters were randomly initialized. A learning rate of 0.08, a momentum of 0.9 and a batch size of 128 were used for the training. The learning rate was gradually decreased when the frame accuracy did not improve for a cross validation set. We used 40 epochs to train each configuration. The dropout rate was set at 0.5 for all configurations. We used a trigram language model for decoding. In the following, we carried out experiments with different unfolded RNN configurations to confirm the effectiveness of using a CNN and the jump ahead connections. The different configurations are detailed below.

### 5.1. Unfolded RNN

Our first configuration is an unfolded RNN without convolutional layers. 11 frames are arranged into 5 frame blocks in which each frame block has 7 frames. Our unfolded RNN has 1024 hidden neurons for the hidden layer and the output layer. One additional fully connected layer is used on top of the output layer which has 5976 neurons at the output. Sigmoid functions are used for the nonlinear activation function. Dropout is used for both recurrent and non-recurrent connections.

### 5.2. LSTM

We used LSTM network which has 3 LSTM layers. Each LSTM layer has 1000 units. One fully connected layer is used on top of the last LSTM layers. We applied the dropout operator only to the non-recurrent connections and input features. The network is trained using Back-propagation Though Time with truncated time step of 7.

### 5.3. Unfolded deep RNN

Our second configuration is an unfolded deep RNN (dRNN). Fig. 3 shows the dRNN architecture. The dRNN has almost the same architecture as an unfolded RNN except that an MLP with 3 layers is used in place of the hidden-to-hidden matrix. Each layer has 1024 neurons.

### 5.4. Unfolded deep RNN with jump ahead connections

Our third configuration is a dRNN with jump ahead connections. It has almost the same architecture as the dRNN described in 5.3. Fig. 3 shows an example of using jump ahead connections (dash lines). In our experiment, we used one ($\Lambda = \{t-1, t-2\}$) or three jump ahead connections ($\Lambda = \{t-1, t-2, t-3, t-4\}$). We denote them as dRNN+J (J

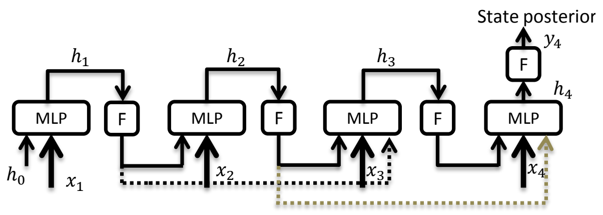

Figure 3: *An unfolded deep RNN with jump ahead connections.*

Table 1: *WERs for the CHiME3 dev and eval sets on noisy speech.*

| Methods | dev | eval |
|---|---|---|
| RNN | 18.15 | 28.10 |
| LSTM | 15.77 | 27.53 |
| dRNN | 17.48 | 27.94 |
| dRNN+J | 17.22 | 27.73 |
| dRNN+M | 17.46 | 27.74 |
| CNN | 12.83 | 21.29 |
| uCNN-RNN | 11.91 | 20.50 |
| UDRCNN | 11.50 | 19.38 |
| UDRCNN+J | 11.32 | 19.18 |
| UDRCNN+M | 11.46 | 19.03 |

means jump ahead connection) and dRNN+M (M means multiple jump ahead connections).

### 5.5. CNN baseline

To determine the degree to which our UDRCNN outperform single CNN baseline, we also evaluated our CNN baseline which has 7 layers. The first convolutional layer uses $(5 \times 11)$ filters, 3 input and 180 output feature maps. The second convolutional layer uses $(1 \times 5)$ filters, 180 input and 180 output feature maps. After each convolution layer, the resolution of the output feature map is reduced using max-pooling. Three fully connected layers with 2048 output nodes are used.

### 5.6. UDRCNN

We used a UDRCNN network as our fifth configuration. In our UDRCNN network, each CNN sub-network we used has almost the same architecture as the CNN described in Section 5.5, except for the first convolutional layer. Here, the first convolutional layer has performs two convolutional operations. Each convolutional operation uses $(5 \times 7)$ filters, 3 input and 180 output feature maps. This is because the CNN processes 7 rather than 11 frames. The connection layer is one fully connected layer that has 2048 neurons at the input and 840 neurons at the output. In addition, there are one ($\Lambda = \{t-1, t-2\}$) or three jump ahead connections ($\Lambda = \{t-1, t-2, t-3, t-4\}$). We denote them as UDRCNN+J and UDRCNN+M, respectively.

### 5.7. Unfolded recurrent convolutional neural network (uRNN-CNN)

Our sixth configuration is a uRNN-CNN. We conducted an experiment with this configuration to determine the importance of CNN based recurrent connection. In our uRNN-CNN, the network construction is almost the same as UDRCNN except that the recurrent connection was made at the last hidden layer ($K = 6$). This means that instead of connecting the hidden layer to the input of the convolutional layer, the output of the hidden layer $h^6_{t-1}$ is fed to the same hidden layer $h^6_t$. The connection layer is one fully connected layer that has 2048 neurons at the input and 2048 neurons at the output.

Table 2: *WERs for the CHiME3 dev and eval sets on noisy speech. The context window is 15 frames.*

| Methods | dev | eval |
|---|---|---|
| RNN | 20.62 | 32.43 |
| dRNN+M | 16.56 | 27.12 |
| UDRCNN | 18.15 | 27.76 |
| UDRCNN+M | 11.22 | **18.87** |

### 5.8. Results

#### 5.8.1. From simple to complex sub-network

Experimentally, we found that by using a more complex and deeper network to model the recurrent connection, we obtain much better performance as seen in Table 1. Our LSTM network obtained a 27.53% WER which is slightly better than that of simple unfolded RNN. Specifically, with UDRCNN, we obtained a 19.38% WER. Interestingly, with almost the same number of parameters, using CNNs at the recurrent connection is slightly better than using CNNs at the feature extraction. The UDRCNN obtains a 5.4% relative WER reduction over the uCNN-RNN network.

#### 5.8.2. With and without jump ahead connections

We observed that when we used an unfolded RNN in combination with jump ahead connections, the unfolded RNNs obtained significantly better ASR performance. However, the use multiple jump ahead connections that are too dense might involve a risk (e.g. jump ahead connections are one time step away from each other), especially in the small context window (e.g. 11 frames with 5 corresponding time steps). As seen in Table 1, a dRNN with multiple jump ahead connections does not offer any improvement over a single jump ahead connection(e.g. 27.73% WER and 27.74% WER on the evaluation set). A UDRCNN in combination with multiple jump ahead connections provided slightly worse results with the development (e.g. 11.46% WER) set but slightly better results with the evaluation set (e.g. 19.03% WER). We hypothesize that using multiple jump ahead connections that are too dense within a small context window might not be needed.

Then, we conducted an experiment with a larger context (e.g. 15 frames) as seen in Table 2. There, jump ahead connections provided a larger improvement. No surprisingly, we observed that the performance of the dRNN and UDRCNN degraded considerably. However, using a dRNN and UDRCNN in combination with jump ahead connections (e.g. $\Lambda = \{t-1, t-2, t-3, t-4, t-5\}$) resulted in a better WER score compared with the results shown in Table 1.

In the end, our network obtained a 18.87% WER, namely a 32.4%, a 11.3% and a 7.9% relative WER reduction over an unfolded RNN, CNN and an unfolded recurrent convolutional neural network, respectively.

## 6. Conclusion

This paper investigated how to design a different unfolded deep RNN architecture in which recurrent connections used a convolutional neural network (CNN) to model dependencies between hidden states. We also investigated an unfolded deep RNN in combination with jump ahead connections and confirmed the effectiveness of our approach. We plan to exploit such kinds of architectures in combination with an LSTM network. Moreover, we also plan to replace the simple CNN with a more complex deeper CNNs and with a larger context window to further improve performance [23, 24].

## 7. References

[1] 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 fours research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[2] G. Saon, H. Soltau, A. Emami, and M. Picheny, "Unfolded recurrent neural networks for speech recognition," in *INTERSPEECH*, 2014, pp. 343–347.

[3] G. Saon, T. Sercu, S. Rennie, and H. J. Kuo, "The IBM 2016 english conversational telephone speech recognition system," in *arXiv preprint arXiv:1604.08242*, 2016.

[4] T. Robinson, "An application of recurrent nets to phone probability estimation," *IEEE Transaction on Neural Networks*, vol. 5, no. 2, pp. 298 – 305, 1994.

[5] C. Weng, D. Yu, S. Watanabe, and B. F. Juang, "Recurrent deep neural networks for robust speech recognition," in *ICASSP*, 2014, pp. 5532–5536.

[6] A. Maas, Q. V. Le, T. M. ONeil, O. Vinyals, P. Nguyen, and A. Y. Ng, "Recurrent neural networks for noise reduction in robust asr," in *INTERSPEECH*, 2012.

[7] O. Vinyals, S. Ravuri, and D. Povey, "Revisiting recurrent neural networks for robust ASR," in *ICASSP*, March 2012, pp. 4085–4088.

[8] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *ICASSP*, April 2015, pp. 4580–4584.

[9] A. Graves, A. R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *ICASSP*, May 2013, pp. 6645–6649.

[10] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *INTERSPEECH*, 2014, pp. 338–342.

[11] A. R. Mohamed, F. Seide, D. Yu, J. Droppo, A. Stoicke, G. Zweig, and G. Penn, "Deep bi-directional recurrent networks over spectral windows," in *ASRU*, Dec 2015, pp. 78–83.

[12] S. Zhang, Y. Wu, T. Che, Z. Lin, R. Memisevic, R. Salakhutdinov, and Y. Bengio, "Architectural complexity measures of recurrent neural networks," in *NIPS*, December 2016.

[13] T. Lin, B. G. Horne, P. Tino, and C. L. Giles, "Learning long term dependencies is not as difficult with NARX recurrent neural network," *IEEE Transaction on Neural Networks*, vol. 7, no. 6, pp. 1329 – 1338, 1996.

[14] Y. Zhang, G. Chen, D. Yu, K. Yao, S. Khudanpur, and J. R. Glass, "Highway long short-term memory RNNS for distant speech recognition," in *ICASSP*, 2016, pp. 5755–5759.

[15] J. G. Zilly, R. K. Srivastava, J. Koutnk, and J. Schmidhuber, "Recurrent highway networks," in *arXiv preprint arXiv:1607.03474*, 2016.

[16] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to construct deep recurrent neural networks," in *arXiv preprint arXiv:1312.6026*, 2014.

[17] M. Pachitariu and M. Sahani, "Regularization and nonlinearities for neural language models: when are they needed?" in *arXiv preprint arXiv:1301.5650*, 2013.

[18] G. Diamos, S. Sengupta, B. Catanzaro, M. Chrzanowski, A. Coates, E. Elsen, J. Engel, A. Hannun, and S. Satheesh, "Persistent RNNs: Stashing recurrent weights on-chip," in *ICML*, 2016, pp. 2024–2033.

[19] S. Zhang, C. Liu, H. Jiang, S. Wei, L. R. Dai, and Y. Hu, "Feedforward sequential memory networks: A new structure to learn long-term dependency," *arXiv preprint arXiv:1512.08301*, 2015.

[20] S. Zhang, H. Jiang, S. Xiong, S. Wei, and L. R. Dai, "Compact feedforward sequential memory networks for large vocabulary continuous speech recognition," in *17th Annual Conference of the International Speech Communication Association, San Francisco, CA, USA*, 2016, pp. 3389–3393.

[21] R. Soltani and H. Jiang., "Higher order recurrent neural networks," in *arXiv preprint arXiv:1605.00064*, 2016.

[22] J. Barker, R. Marxer, E. Vincent, and S. Watanabe, "The third CHiME speech separation and recognition challenge: Dataset, task and baselines," in *ASRU*, Dec 2015, pp. 504–511.

[23] T. Yoshioka, N. Ito, M. Delcroix, A. Ogawa, K. Kinoshita, M. Fujimoto, C. Yu, W. J. Fabian, M. Espi, T. Higuchi, S. Araki, and T. Nakatani, "The ntt chime-3 system: Advances in speech enhancement and recognition for mobile multi-microphone devices," in *ASRU*, 2015, pp. 436–443.

[24] J. Heymann, L. Drude, and R. Haeb-Umbach, "Wide residual blstm network with discriminative speaker adaptation for robust speech recognition," in *The 4th International Workshop on Speech Processing in Everyday Environments , San Francisco, September, 2016*, 2016.