Unfolded Recurrent Neural Networks for Speech Recognition

George Saon, Hagen Soltau, Ahmad Emami and Michael Picheny

IBM T. J. Watson Research Center, Yorktown Heights, NY, 10598

gsaon@us.ibm.com

10.21437/Interspeech.2014-81

Abstract

We introduce recurrent neural networks (RNNs) for acoustic modeling which are unfolded in time for a fixed number of time steps. The proposed models are feedforward networks with the property that the unfolded layers which correspond to the recurrent layer have time-shifted inputs and tied weight matrices. Besides the temporal depth due to unfolding, hierarchical processing depth is added by means of several non-recurrent hidden layers inserted between the unfolded layers and the output layer. The training of these models: (a) has a complexity that is comparable to deep neural networks (DNNs) with the same number of layers; (b) can be done on frame-randomized minibatches; (c) can be implemented efficiently through matrix-matrix operations on GPU architectures which makes it scalable for large tasks. Experimental results on the Switchboard 300 hours English conversational telephony task show a 5% relative improvement in word error rate over state-of-the-art DNNs trained on FMLLR features with i-vector speaker adaptation and hessian-free sequence discriminative training.

Index Terms: recurrent neural networks, speech recognition

1. Introduction

One of the current research problems in neural network acoustic modeling has to do with finding appropriate input feature representations for the networks. Regardless of the type of features, it has been observed that providing ample temporal context, either through consecutive frame stacking or through finite differences, has a positive effect on performance [1]. Instead of a DNN operating on blocks of consecutive frames (or a linear transformation thereof), one can use a recurrent neural network which, by design, is able to handle temporal context efficiently.

RNNs have one or more hidden layers with recurrent connections i.e. the output of a hidden layer at time $t-1$ is part of the input of that layer at time t. These connections endow RNNs with "memory" whereby acoustic-phonetic events detected at some earlier times are passed on via the recurrent layer for further processing. Ideally, the recurrent weights learn which acoustic events are important for classification and need to be preserved and which events can be discarded. The memory of an RNN is potentially infinite although in practice events that are farther in time have to go through more layers of processing which will attenuate their influence. This is qualitatively different from a time-delay neural network (TDNN) [2] or a convolutional neural network (CNN) with time pooling [3] where the outputs within a temporal window are equally likely to be selected regardless of time shift. All these networks have the advantage over DNNs that they can detect events independent of their position in time within the window of analysis and are less affected by temporal distortions in the input.

While obtaining good results for language modeling [4], the application of recurrent nets to acoustic modeling is still making inroads. In [5], the author trains an RNN with one hidden layer with backpropagation through time (BPTT) [6] and obtains competitive phone recognition results on the TIMIT task. The same task is considered in [7], where the authors train a bidirectional RNN with forward and backward BPTT. In [8], the authors apply second-order optimization techniques to train RNNs on the Aurora2 task. More recently, [9, 10] proposed the use of deep RNNs obtained by stacking multiple RNNs [11] with mixed results: the models have good performance on TIMIT but cannot improve over DNNs on a 14 hour subset of the Wall Street Journal corpus. In all fairness, the authors and [12] also consider a more complex architecture called deep long short-term memory (LSTM) which is able to outperform DNNs. Whether it is regular RNNs, bidirectional RNNs or LSTMs, training these models on large tasks is challenging because of the sequential nature of BPTT which makes parallelization difficult.

The model proposed in this paper is a deep hidden-to-output RNN [14] with one recurrent layer that is closest to the input followed by several non-recurrent hidden layers and an output layer. Moreover, the recurrent layer is unrolled in time for a fixed number of time steps. The resulting model is a feedforward encoding network where the weight matrices corresponding to the recurrent layer are tied. By virtue of being a feedforward network, the model can be trained using frame randomization [15] and matrix-matrix operations which can be efficiently implemented on a GPU architecture. While this paper was under review, [13] proposed a deep RNN architecture for robust ASR that is similar to ours, the main differences being that the recurrent hidden layer is in the middle and that [13] does not consider frame randomization.

The paper is organized as follows: in section 2 we describe basic RNNs and our proposed model; in section 3 we present some experimental evidence of its utility, and in section 4 we summarize our findings and propose future directions.

2. Recurrent neural networks

Let us denote by $(\mathbf{x}_1, \ldots, \mathbf{x}_T)$, an acoustic observation sequence, by $(\mathbf{h}_1, \ldots, \mathbf{h}_T)$ the corresponding hidden state activation sequence and by $(\mathbf{y}_1, \ldots, \mathbf{y}_T)$ the sequence of output activations computed by the network. A classification RNN with a single recurrent layer is governed by the following equations which are iterated from $1 \ldots T$:

$$\mathbf{h}_t = \sigma(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\tilde{\mathbf{h}}_{t-1}) \quad (1)$$

$$\mathbf{y}_t = \mathrm{softmax}(\mathbf{W}_{\mathrm{hy}}\tilde{\mathbf{h}}_{\mathrm{t}}) \quad (2)$$

where $\mathbf{W}_{xh}$, $\mathbf{W}_{hh}$, $\mathbf{W}_{hy}$ represent respectively, the input-to-hidden, hidden-to-hidden and hidden-to-output weight matrices. We use the notation $\tilde{\mathbf{h}}_t = [\mathbf{h}_t^T; 1]^T$ for extended hidden

activation vectors and σ and $\mathrm{softmax}$ are the elementwise applications of the sigmoid and soft-max functions. Given training data $\{(\mathbf{x}_1^i, \hat{\mathbf{y}}_1^i), \ldots, (\mathbf{x}_{T_i}^i, \hat{\mathbf{y}}_{T_i}^i)\}_{i=1}^N$, the parameters of the RNN can be adjusted such as to minimize

$$\mathcal{L}(\mathbf{W}_{xh}, \mathbf{W}_{hh}, \mathbf{W}_{hy}) = \frac{1}{\sum_{i=1}^N T_i} \sum_{i=1}^N \sum_{t=1}^{T_i} D(\mathbf{y}_t^i, \hat{\mathbf{y}}_t^i) \quad (3)$$

where D is a suitable loss function such as cross-entropy and $\mathbf{y}_t^i$ are computed from $\mathbf{x}_1^i \ldots \mathbf{x}_t^i$ using (1),(2). Figure 1 illustrates an RNN architecture with one hidden recurrent layer.

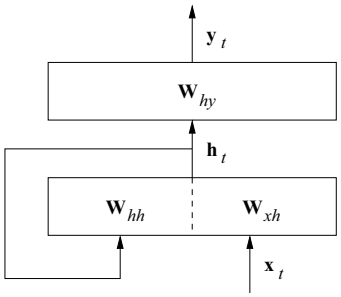

Figure 1: RNN architecture with one recurrent layer.

2.1. Deep hidden-to-output and stacked RNN

Instead of directly connecting the recurrent layer to the output layer, one can interpose L non-recurrent hidden layers. The resulting network is termed deep hidden-to-output RNN (DHO-RNN) [14] and is described by the following equations:

$$\mathbf{h}_t^0 = \sigma(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}^0 \tilde{\mathbf{h}}_{t-1}^0) \quad (4)$$

$$\mathbf{h}_t^l = \sigma(\mathbf{W}_{hh}^l \tilde{\mathbf{h}}_t^{l-1}),\ l = 1 \ldots L \quad (5)$$

$$\mathbf{y}_t = \mathrm{softmax}(\mathbf{W}_{\mathrm{hy}} \tilde{\mathbf{h}}_{\mathrm{t}}^{\mathrm{L}}) \quad (6)$$

A more general model was proposed by [11], analyzed by [16], and applied to ASR by [9] and consists in stacking multiple RNNs into a deep RNN. The recurrent layers of this model operate at different timescales which should be beneficial for speech recognition. The forward pass in a stacked RNN is given by the equations:

$$\mathbf{h}_t^0 = \sigma(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}^0 \tilde{\mathbf{h}}_{t-1}^0) \quad (7)$$

$$\mathbf{h}_t^l = \sigma(\mathbf{V}_{hh}^l \mathbf{h}_t^{l-1} + \mathbf{W}_{hh}^l \tilde{\mathbf{h}}_{t-1}^l),\ l = 1 \ldots L \quad (8)$$

$$\mathbf{y}_t = \mathrm{softmax}(\mathbf{W}_{\mathrm{hy}} \tilde{\mathbf{h}}_{\mathrm{t}}^{\mathrm{L}}) \quad (9)$$

where $\mathbf{V}_{hh}^l, \mathbf{W}_{hh}^l$ represent the hidden-to-hidden transition matrices between layers and time frames, respectively.

2.2. Unfolded RNNs

For simplicity, we focus here on RNNs with a single hidden layer and omit the bias terms. From (1), it can be seen that $\mathbf{h}_t$ depends on $\mathbf{x}_1, \ldots, \mathbf{x_t}$. Indeed,

$$\begin{aligned}\mathbf{h}_t &= \sigma(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1}) \\ &= \sigma(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\sigma(\mathbf{W}_{xh}\mathbf{x}_{t-1} + \mathbf{W}_{hh}\mathbf{h}_{t-2})) \\ &\ldots \\ &= \sigma(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\sigma(\ldots + \mathbf{W}_{hh}\sigma(\mathbf{W}_{xh}\mathbf{x}_1 + \mathbf{W}_{hh}\mathbf{h}_0)))\end{aligned} \quad (10)$$

where $\mathbf{h}_0$ is typically initialized to 0. The same computation can be carried out by a feedforward *encoding* network obtained by unrolling the recurrent layer from $t \ldots 1$. Notice that the weight matrices for the unfolded layers have to be identical (and equal to the matrix of the recurrent layer) which will place some constraints on how these networks are trained.

In this paper we propose to use encoding networks for RNNs where the unfolding is done for a fixed number of time steps. These networks are trained with a variant of the truncated backpropagation through time (BPTT) algorithm which will be explained in the next subsection.

2.3. Truncated backpropagation through time

It is helpful to organize the training data as $\mathcal{S} = \{(\mathbf{x}_{t-k+1}^i, \ldots, \mathbf{x}_t^i, \hat{\mathbf{y}}_t^i) | t = 1 \ldots T_i, i = 1 \ldots N\}$. Next, we unfold the RNN k times and create an encoding network where the recurrent weight matrix ($[\mathbf{W}_{hh}; \mathbf{W}_{xh}]$ is replicated k times ($[\mathbf{W}_{hh}^{(1)}; \mathbf{W}_{xh}^{(1)}], \ldots, [\mathbf{W}_{hh}^{(k)}; \mathbf{W}_{xh}^{(k)}]$). The resulting network architecture is illustrated in Figure 2.

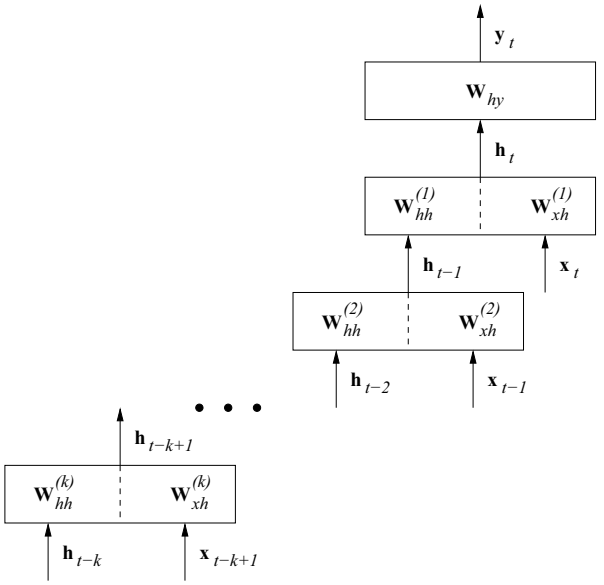

Figure 2: Feedforward encoding network for k time steps.

With these notations, the algorithm for training the encoding network can be summarized as follows.

Algorithm 1 Truncated backpropagation through time with frame randomization

Input: training data $\mathcal{S}$, initial weights $[\mathbf{W}_{hh}^{(1)}; \mathbf{W}_{xh}^{(1)}], \ldots, [\mathbf{W}_{hh}^{(k)}; \mathbf{W}_{xh}^{(k)}], \mathbf{W}_{hy}$

1: **while** convergence criterion not met **do**
2: **for** $(\mathbf{x}_{t-k+1}, \ldots, \mathbf{x}_t, \hat{\mathbf{y}}_t) \in Randomize(\mathcal{S})$ **do**
3: forward propagate $\mathbf{x}_{t-k+1}, \ldots, \mathbf{x}_t$ and obtain $\mathbf{y}_t$
4: backward propagate $\frac{\partial D(\mathbf{y}_t, \hat{\mathbf{y}}_t)}{\partial \mathbf{y}_t}$
5: update $[\mathbf{W}_{hh}^{(1)}; \mathbf{W}_{xh}^{(1)}], \ldots, [\mathbf{W}_{hh}^{(k)}; \mathbf{W}_{xh}^{(k)}], \mathbf{W}_{hy}$
6: average $[\mathbf{W}_{hh}^{(1)}; \mathbf{W}_{xh}^{(1)}], \ldots, [\mathbf{W}_{hh}^{(k)}; \mathbf{W}_{xh}^{(k)}]$
7: **end for**
8: **end while**

Output: trained weights $[\mathbf{W}_{hh}^{(1)}; \mathbf{W}_{xh}^{(1)}], \ldots, [\mathbf{W}_{hh}^{(k)}; \mathbf{W}_{xh}^{(k)}], \mathbf{W}_{hy}$

The main difference with standard (truncated) BPTT is in step 2 where the training data is randomized for every epoch. In standard BPTT, the data for utterance i is traversed in sequence from $1 \dots T_i$ (and from $T_i \dots 1$). Here, because of the randomization, the $k-1$ preceding data points have to be stored for each $\mathbf{x}_t$. The averaging step 6 ensures that the weight matrices are the same after every update. The extension to frame-randomized minibatch SGD is straightforward. In step 2 we select M tuples $(\mathbf{x}^{i_1}_{t_1-k+1}, \dots, \mathbf{x}^{i_1}_{t_1}, \hat{\mathbf{y}}^{i_1}_{t_1}), \dots, (\mathbf{x}^{i_M}_{t_M-k+1}, \dots, \mathbf{x}^{i_M}_{t_M}, \hat{\mathbf{y}}^{i_M}_{t_M})$, $i_j \in \{1, \dots, N\}$, $t_j \in \{1, \dots, T_{i_j}\}$ and group them into tuples of time-shifted matrices $([\mathbf{x}^{i_1}_{t_1-k+1}; \dots; \mathbf{x}^{i_M}_{t_M-k+1}], \dots, [\mathbf{x}^{i_1}_{t_1}; \dots; \mathbf{x}^{i_M}_{t_M}])$ which are then forward propagated to obtain $[\mathbf{y}^{i_1}_{t_1}; \dots; \mathbf{y}^{i_M}_{t_M}]$. Similarly, the matrix of the error gradient $\partial[D(\mathbf{y}^{i_1}_{t_1}, \hat{\mathbf{y}}^{i_1}_{t_1}), \dots, D(\mathbf{y}^{i_M}_{t_M}, \hat{\mathbf{y}}^{i_M}_{t_M})]/\partial \mathbf{y}_t$ is propagated backwards. Lastly, note that the averaging step 6 is performed once per minibatch as opposed to once per sample.

3. Experiments and results

The experiments were conducted on a 300 hour subset of the Switchboard English conversational telephony speech recognition task. We report results on the SWB part of the Hub5 2000 testset which contains 2.1 hours of audio and 21.4K words.

We extract 13-dimensional VTL-warped PLP cepstra every 10ms which are mean and variance normalized on a per speaker basis. Every 9 consecutive cepstral frames are concatenated and projected down to 40 dimensions using LDA. The resulting features are decorrelated by means of a global semi-tied covariance transform (STC). The LDA/STC features are further transformed with feature-space MLLR (FMLLR) per conversation side for both training and test. The FMLLR transforms are estimated using a baseline GMM-HMM system. As suggested in [15], the input to the baseline DNNs is formed by concatenating 11 consecutive FMLLR frames.

Additionally, we consider i-vector speaker adaptation [17] using 100-dimensional i-vectors which are appended to the 11×40 FMLLR features. The i-vectors are extracted using a GMM with 2048 diagonal covariance Gaussians which are estimated on the 40-dimensional FMLLR features.

3.1. Baseline DNNs

All baseline nets have 6 hidden layers with sigmoid activation functions: the first 5 with 2048 units and the last one with 256 units for parameter reduction and faster training time [18]. The output layer has 9300 softmax units that correspond to the context-dependent HMM states obtained by growing a phonetic decision tree with pentaphone crossword context. Following the recipe outlined in [15], the training data is fully randomized at the frame level within a window of 25 hours and we trained the nets with stochastic gradient descent on minibatches of 250 frames and a cross-entropy criterion. Prior to the cross-entropy training of the full network, we used layerwise discriminative pretraining [15] by running one cross-entropy sweep over the training data for the intermediate networks obtained by adding one hidden layer at a time. Cross-entropy training converged after 15 iterations.

3.2. Input features for unfolded RNNs

In theory, the input for an RNN should be the current frame. In order to provide more temporal context for acoustic-phonetic event detection, we found it beneficial to consider blocks of consecutive frames as input. For the unfolded RNNs, the blocks are shifted back in time by one frame from $t \dots t-k+1$. In Figure 3, we show a configuration where the block of frames $t \dots t+5$ is shifted back in time from $t \dots t-5$. More precisely, the inputs of an unfolded RNN with $k=6$ are given by the blocks of FMLLR frames $[t \dots t+5]$, $[t-1 \dots t+4], \dots, [t-5 \dots t]$. Such a configuration makes the results directly comparable to a DNN which is trained on inputs of the form $t-5 \dots t+5$ since both models see the same frames at time t. In the case of i-vector speaker adaptation, the 100-dimensional i-vector is appended to all the time-shifted blocks.

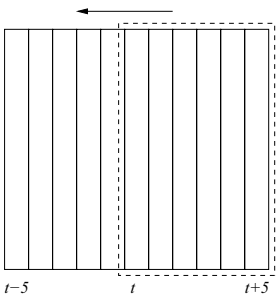

Figure 3: Input features for unfolded RNNs for $k=6$.

3.3. Training unfolded RNNs

The training data is the same as for the baseline DNNs i.e. 10 25-hour parts with minibatches of 250 samples where a sample represents a $t-5 \dots t+5$ block of FMLLR frames (concatenated with a 100-dimensional i-vector where applicable). The difference is that, at runtime, we extract the submatrices corresponding to the time-shifted blocks $[t \dots t+5]$, $[t-1 \dots t+4], \dots, [t-5 \dots t]$ which are fed as input to the unfolded RNN (see Figure 2). The RNN topology that we consider has one recurrent layer with 2048 sigmoid units, 4 hidden non-recurrent layers with 2048 sigmoid units, 1 hidden sigmoid layer with 512 units and an output layer with 9300 softmax units. The unfolded RNN is trained similarly to the baseline DNN with layerwise discriminative pre-training by running one cross-entropy sweep for the intermediate networks obtained by adding one hidden non-recurrent layer at a time. The final network is fine-tuned with 15 passes of cross-entropy SGD and each pass takes roughly 8 hours on one Tesla K20 GPU device.

In Figure 4, we compare the phone error rates on held-out data for the DNNs and unfolded RNNs without and with i-vector speaker adaptation. We observe that the unfolded RNNs exhibit consistently better error rates than the DNNs over the entire course of the optimization.

Both types of models are used directly in a hybrid decoding scenario by subtracting the logarithm of the HMM state priors from the log of the DNN/RNN output scores. The vocabulary contains 30.5K words and 32.8K pronunciation variants. The decoding language model is a 4-gram LM with 4M n-grams.

In Table 1, we compare the word error rates on Hub5'00 for DNNs and unfolded RNNs without and with i-vector speaker adaptation before and after hessian-free sequence discriminative training with a state-based minimum Bayes risk objective function as described in [19]. For sequence training, the cross-entropy trained models are used to generate numerator and denominator lattices. The latter are obtained using a weak unigram language model. We observe that the unfolded RNNs im-

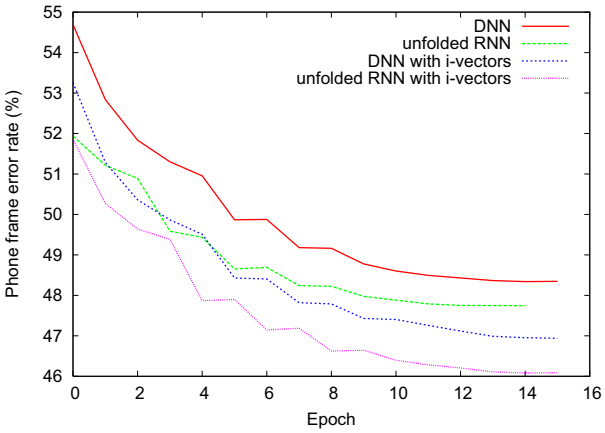

Figure 4: Phone frame error rates on heldout data for DNNs and unfolded RNNs.

prove over DNNs after i-vector speaker adaptation and that the improvement also holds after discriminative sequence training.

i-vectors	Training	DNN	uRNN
no	x-entropy	14.1%	13.5%
no	sequence	12.5%	12.0%
yes	x-entropy	13.2%	12.7%
yes	sequence	11.9%	11.3%

Table 1: Comparison of word error rates for DNNs and unfolded RNNs on Hub5'00 before and after sequence training.

3.4. Folded versus unfolded RNNs

Here we attempt to answer the question of what happens if you fold back an unfolded RNN that was trained with truncated BPTT. In other words, the network is trained as a k-unfolded feedforward net and used as a standard (folded) RNN during decoding. This question is answered in Table 2, where we compare folded and unfolded deep hidden-to-output RNNs for $k = 6$ ($t \ldots t-5$) and $k = 11$ ($t \ldots t-10$).

	unfolded RNN	folded RNN
$k = 6$	13.5%	16.2%
$k = 11$	13.8%	14.4%

Table 2: Comparison of word error rates for folded and unfolded RNNs on Hub5'00 for $k = 6$ and $k = 11$ (no i-vectors, cross-entropy training).

Two observations can be made. First, the $k = 11$ unfolded performance is slightly worse than $k = 6$ which could be due to the fact that averaging the weight matrices up to $t-10$ could result in matrices that change too little over the course of the optimization because of the "vanishing gradient" problem (the gradients become smaller with increased temporal unfolding. A better approach that we are currently investigating might be to use a larger temporal unfolding during the forward pass (say $t \ldots t-10$) in order to get a better estimate of $\mathbf{h}_{t-6}$, and a smaller unfolding during the backward pass (say $t \ldots t-5$) to cope with the small gradients. Alternatively, one can use the sum of the gradients as opposed to averaging the weight matrices which is equivalent to having a k-times larger stepsize for the unfolded layers.

Second, we note that the folded RNN performance is worse (significantly so for $k = 6$) because of the discrepancy for $\mathbf{h}_{t-k}$ (zero in training, non-zero in testing except for the first frame). Lastly, we did not provide a comparison with folded RNNs trained with full BPTT because estimating such networks on 300 hours of data is impractical. Moreover, they will likely be worse than their unfolded counterparts because of the lack of frame randomization which is clearly beneficial [15].

4. Discussion

RNNs provide an efficient mechanism to integrate temporal context for acoustic modeling. This is done by means of a recurrent hidden layer that serves as a memory of past acoustic-phonetic events. Training folded RNNs on large amounts of data directly is impractical because the original BPTT algorithm requires traversing the data for each utterance in sequence. Instead, we propose to use truncated BPTT with frame randomization and minibatch SGD training. This can be efficiently implemented with feedforward encoding networks which are unfolded RNNs for a fixed number of time steps. Similar to DNNs, we add hierarchical processing depth via multiple non-recurrent hidden layers inserted between the unfolded layers and the output layer. The proposed networks show good performance on the Switchboard task after i-vector speaker adaptation and discriminative sequence training.

Future work will deal with having different temporal unfoldings for the forward and backward passes, and with tuning the initial stepsizes for the unfolded and regular layers separately. Additionally, we plan to investigate bidirectional unfolded RNNs by unrolling the recurrent layer both backward and forward in time. Last but not least, we intend to apply some of these ideas to language modeling.

5. References

[1] H. Soltau, H.-K. Kuo, L. Mangu, G. Saon, and T. Beran, "Neural network acoustic models for the darpa rats program," in *Proc. Interspeech*, 2013.

[2] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 37, no. 3, pp. 328–339, 1989.

[3] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, 1995.

[4] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model." in *INTERSPEECH*, 2010, pp. 1045–1048.

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

[6] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[7] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *Signal Processing, IEEE Transactions on*, vol. 45, no. 11, pp. 2673–2681, 1997.

[8] O. Vinyals, S. V. Ravuri, and D. Povey, "Revisiting recurrent neural networks for robust asr," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4085–4088.

[9] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 6645–6649.

[10] A. Graves, N. Jaitly, and A.-r. Mohamed, "Hybrid speech recognition with deep bidirectional lstm," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 273–278.

[11] J. Schmidhuber, "Learning complex, extended sequences using the principle of history compression," *Neural Computation*, vol. 4, no. 2, pp. 234–242, 1992.

[12] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," *arXiv preprint arXiv:1402.1128*, 2014.

[13] C. Weng, D. Yu, S. Watanabe, and B.-H. Juang, "Recurrent deep neural networks for robust speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014.

[14] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to construct deep recurrent neural networks," *arXiv preprint arXiv:1312.6026*, 2013.

[15] F. Seide, G. Li, X. Chien, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Proc. ASRU*, 2011.

[16] M. Hermans and B. Schrauwen, "Training and analysing deep recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2013, pp. 190–198.

[17] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors," in *Proc. ASRU*, 2013.

[18] T. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *Proc. of ICASSP*, 2013.

[19] B. Kingsbury, T. Sainath, and H. Soltau, "Scalable minimum Bayes risk training of deep neural network acoustic models using distributed Hessian-free optimization," in *Proc. Interspeech*, 2012.