INTERSPEECH 2016
September 8–12, 2016, San Francisco, USA

Modeling Time-Frequency Patterns with LSTM vs. Convolutional Architectures for LVCSR Tasks

Tara N. Sainath, Bo Li

Google, Inc. New York, NY, U.S.A
{tsainath, boboli}@google.com

Abstract

Various neural network architectures have been proposed in the literature to model 2D correlations in the input signal, including convolutional layers, frequency LSTMs and 2D LSTMs such as time-frequency LSTMs, grid LSTMs and ReNet LSTMs. It has been argued that frequency LSTMs can model translational variations similar to CNNs, and 2D LSTMs can model even more variations [1], but no proper comparison has been done for speech tasks. While convolutional layers have been a popular technique in speech tasks, this paper compares convolutional and LSTM architectures to model time-frequency patterns as the first layer in an LDNN [2] architecture. This comparison is particularly interesting when the convolutional layer degrades performance, such as in noisy conditions or when the learned filterbank is not constant-Q [3]. We find that grid-LDNNs offer the best performance of all techniques, and provide between a 1-4% relative improvement over an LDNN and CLDNN on 3 different large vocabulary Voice Search tasks.

1. Introduction

In recent years, both Convolutional Neural Networks (CNNs) [4] and Long-Short Term Memory Recurrent Neural Networks (LSTMs) [5] have become popular alternatives to Deep Neural Networks [6] for large vocabulary continuous speech recognition (LVCSR) tasks. As both CNNs, LSTMs and DNNs are limited in their individual modeling capabilities, recently a CLDNN architecture was introduced to combine the modeling benefits of all three architectures in a unified framework [2]. For example, CLDNNs have shown between a 4-6% relative improvement over LSTMs on various clean Voice Search tasks.

Convolutional layers in the CLDNN model achieve local translation invariance through local filters and pooling. This requires tuning both the filter size and pooling size, which is typically done experimentally [4]. Since pooling is done on linear shifts of the convolutional filter, it is most effective when the filterbank is constant-Q (such as log-mel) and therefore a linear shift across filters corresponds to a linear shift in pitch. Pooling is less effective when the filterbank deviates from being constant-Q, for example with raw-waveform modeling where the filterbank is learned in the network [3]. In addition, we have also observed convolution layers to hurt performance in very noisy conditions.

Frequency [7] and Time-Frequency LSTMs [8, 9] have been introduced as alternatives to CNNs to model correlations in frequency. F-LSTMs and TF-LSTMs capture translation invariance through local filters and recurrent connections. Since they do not require any pooling operation, they should be more adaptable to different types of input features. However, to the best of our knowledge no fair comparison has been done between CNNs, F-LSTM and TF-LSTM layers for speech tasks.

The first goal of this paper is to perform a good comparison among convolutional, F-LSTM and TF-LSTM architectures as the first layer in an LDNN network [2]. We compare performance for both mild and moderately noisy 2,000 hr Voice Search tasks using log-mel features. We find that for mild speech, F-LDNNs and CLDNNs match in performance, confirming the thought that an F-LSTM layer has similar modeling capabilities to a convolutional layer [1]. Furthermore, by using TF-LDNNs, which give stronger modeling capabilities, we can achieve a 1% relative improvement in WER over a strong CLDNN model. For moderate noise where convolution hurts, we find that TF-LDNNs provide a 3% relative improvement over LDNNs.

Second, alternative architectures have also been proposed to model two dimensional inputs for vision tasks. For example, Grid LSTMs [10] have separate LSTMs in both time and frequency dimensions. When the LSTMs overlap at a specific time-frequency point, the gate functions and output activations are computed using both time and frequency LSTM cell states. Grid LSTMs offer the benefit independently modeling time and frequency correlations. In addition, ReNets [1] have completely separate time and frequency LSTMs which do not include states of both during overlap. The potential benefit of ReNets is that they are easier to parallelize since the time and frequency LSTMs move independently. We find on the noisier test set that a grid-LDNN provides a 1% relative improvement over a TF-LDNN, while ReNet-LDNN degrades performance.

The third goal of this paper is to compare convolution to 2D LSTM architectures when the filterbank is learned, specifically in a multichannel model [11]. We find that when the learned filterbank is not constant-Q, the convoutional layer degrades performance. However, the grid-LDNN provides a 3% relative improvement over the LDNN, showing the benefit of modeling time-frequency patterns with an LSTM rather than a convolutional layer when the filterbank is learned.

2. Architecture Description

This section describes the various architectures explored in this paper to model time-frequency patterns.

2.1. Baseline LDNN Architecture

The general architecture used for experiments in this paper is described as follows. First, frame x_t is passed as input to the network. For simplicity, we assume that each frame x_t is a log-mel filterbank feature. Then, this frame can be optionally processed in frequency by either CNN or LSTM layers, as denoted by the "frequency-processing" block in Figure 1. The details of this frequency processing block will be described in subsequent sections. The output of the frequency processing layer is then passed to a 256-dimensional linear low-rank layer.

The rest of the architecture is similar to the CLDNN architecture described in [2]. Specifically, the low-rank layer is passed to 3 LSTM layers with 832 cells and a 512 unit projection layer, then one DNN layer with 1,024 hidden units, and finally a softmax layer with 13,522 context-dependent state output targets. If no frequency processing is performed, x_t is passed directly to the first LSTM layer, and thus the architecture is known as an LDNN.

During training, the network is unrolled for 20 time steps for training with truncated backpropagation through time. In addition, the output state label is delayed by 5 frames, as we have observed that information about future frames improves the prediction of the current frame [2].

2.2. Frequency Processing

2.2.1. CNN

The convolutional layer explored in this paper is very similar to the CNN layer described in [4]. Specifically, we use 1 convolutional layer, with 256 feature maps. The filter size and pooling size is dependent on the input feature dimension for x_t. For example, for a 128-dimensional input, we use an 21x1 frequency-time filter for the convolutional layer. Our pooling strategy is to use non-overlapping max pooling, and pooling in frequency only is performed with a pooling size of 9.

2.2.2. F-LSTM

The LSTM architecture, as described in [12], consists of a set of recurrently connected subnetworks, referred to a *memory blocks*. Each memory block contains *memory cells* to store the temporal state of the network, as well as three multiplicative gate units to control information flow. The *input gate* controls the information passed from the input activations into the memory cells, while the *output gate* controls the information passed from the memory cells to the rest of the network. Finally, the *forget gate* adaptively resets the memory of the cell. At each step j, the LSTM model is given by the following equations:

$$i_j = \sigma(W_{ix}x_j + W_{im}m_{t-1} + W_{ic}c_{t-1} + b_i) \quad (1)$$
$$f_j = \sigma(W_{fx}x_j + W_{mf}m_{t-1} + W_{cf}c_{t-1} + b_f) \quad (2)$$
$$c_j = f_j \odot c_{t-1} + i_j \odot g(W_{cx}x_j + W_{cm}m_{t-1} + b_c) \quad (3)$$
$$o_j = \sigma(W_{ox}x_j + W_{om}m_{t-1} + W_{oc}c_t + b_o) \quad (4)$$
$$m_j = o_j \odot h(c_j) \quad (5)$$

where i_j, f_j, c_j and o_j denote the input, forget, memory cell and output gate activations at step j. m_j is the output of the LSTM layer. W are the different weight matrices, for example W_{ix} is the weight matrix from the input gate to the input. $\odot$ is an element-wise dot product. Finally, σ is the logistic sigmoid non-linearity while g and h are the cell input and output activations, which we take to be *tanh*. An LSTM is typically used to model the speech signal in time, namely the input to the LSTM at each time step t is given by $x_j = x_t$. Thus, we will refer to the above model as a time LSTM (T-LSTM) for simplicity. This type of layer is used in the LDNN architecture in Figure 1.

The frequency LSTM (F-LSTM) uses the exact same equations as given above, except that we model a sequential process in frequency. Our F-LSTM implementation models that of [7]. Specifically, given input feature $x_t \in \Re^N$, we window the first F elements from this feature, denoted by $x_0 = x_t^{0:F} \in \Re^F$

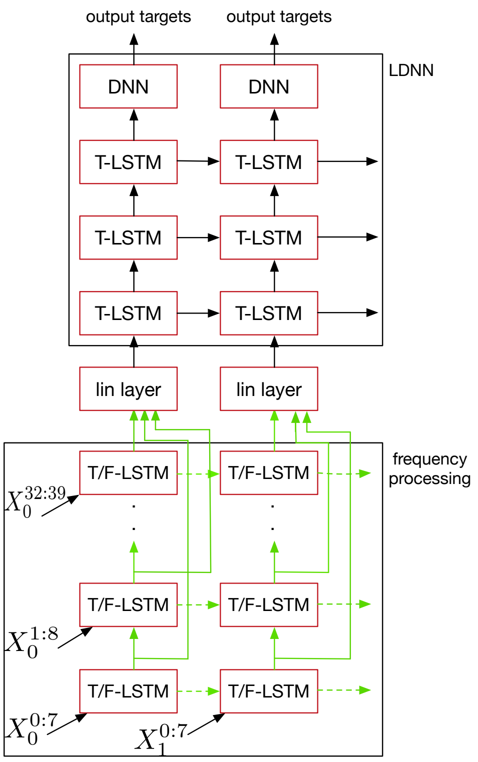

Figure 1: F-LSTM and TF-LSTM with LDNN. The dashed-lines are ignored by the F-LSTM and indicate that the output of the TF-LSTM is shared in both time and frequency.

and give this as input to the LSTM. At the next step, we stride the window over the input by S, and take the next F features, denoted by $x_1 = x_t^{S:(F+S)} \in \Re^F$, and pass this to the LSTM. Hence the input to the LSTM (Equations (1) - (5)) at each frequency step k is given by $x_j = x_k = x_t^{k*S:(F+k*S)}$. In most cases $S < F$ so the chunks have overlapping information. The F-LSTM is thus unrolled over frequency by an amount $L = (N-F)/S+1$. This process is shown pictorially in Figure 1. The output out of each F-LSTM, denoted by $\{m_0, \ldots, m_L\}$ are concatenated together and given to a linear dimensionality-reduction layer, and then given to an LDNN. We will call the F-LSTM with LDNN architecture an F-LDNN. In this work, we will take feature dimension $N = 128$, filter size $F = 24$, stride $S = 4$, and LSTM cells to be 64, to match similar filter sizes and parameters with the CNN.

The F-LSTM is similar to the convolutional layer as both models look over a small local frequency patch and share model parameters as the filter is shifted. The main difference is that the F-LSTM models frequency variations through a recurrent state that is passed from one unrolled timestep to another. However, the convolutional layer has a subsequent pooling step to achieve local translational invariance. As argued in [1], the recurrent state in F-LSTMs can emulate local competition among features similar to max pooling. However, F-LSTMs have the added benefit that there is no need to tune the pooling parameters.

2.2.3. TF-LSTM

The F-LSTM can be extended to model the sequential process of the signal in both time and frequency jointly with a time-frequency LSTM (TF-LSTM) [8]. Since speech has correlations in both time and frequency, we believe that TF-LSTMs should perform as good if not better than the F-LSTM. For each frequency step k and time step t, the TF-LSTM is given by the following equations:

$$\begin{aligned} i_{t,k} &= \sigma(W_{ix}x_{t,k} + W^{(t)}_{im}m_{t-1,k} + W^{(k)}_{im}m_{t,k-1} + \\ &\quad W_{ic}c_{t-1,k} + b_i) \end{aligned} \tag{6}$$

$$\begin{aligned} f_{t,k} &= \sigma(W_{fx}x_{t,k} + W^{(t)}_{fm}m_{t-1,k} + W^{(k)}_{fm}m_{t,k-1} + \\ &\quad W_{cf}c_{t-1,k} + b_f) \end{aligned} \tag{7}$$

$$\begin{aligned} c_{t,k} &= f_{t,k} \odot c_{t-1,k} + i_{t,k} \odot g(W_{cx}x_{t,k} + W^{(t)}_{cm}m_{t-1,k} + \\ &\quad W^{(k)}_{cm}m_{t,k-1} + b_c) \end{aligned} \tag{8}$$

$$\begin{aligned} o_{t,k} &= \sigma(W_{ox}x_{t,k} + W^{(t)}_{om}m_{t-1,k} + W^{(k)}_{om}m_{t,k-1} + \\ &\quad W_{oc}c_{t,k} + b_o) \end{aligned} \tag{9}$$

$$m_{t,k} = o_{t,k} \odot h(c_{t,k}) \tag{10}$$

In comparison to the F-LSTM and T-LSTM in Equations (1)-(5), the TF-LSTM uses the output from the previous frequency step (i.e., $m_{t,k-1}$) and the previous time step (i.e., $m_{t-1,k}$). These previous outputs are weighted by separate weights $W^{(t)}_{*m}$ and $W^{(k)}_{*m}$ respectively. Notice the TF-LSTM still produces one output, $m_{t,k}$, at each time-frequency (k, t) step. In Figure 1, this shared output is denoted by a dotted line.

Given input feature $x_t \in \Re^N$, our TF-LSTM still windows the input feature with a filter size of F and a stride of S, similar to the F-LSTM. At each time step t, the output out of each TF-LSTM, denoted by $\{m_{t,0}, \dots, m_{t,L}\}$ are concatenated together and given to a linear dimensionality-reduction layer, and then the LDNN.

2.2.4. *Grid-LSTM*

A grid-LSTM [10] is very similar to a TF-LSTM except there are separate LSTMs which move in time and frequency. However, at a current time-frequency bin, the grid frequency LSTM (gF-LSTM) uses the state of the grid time LSTM (gT-LSTM) from the previous timestep, and similarly the gT-LSTM uses the state of the gF-LSTM from the previous frequency step. The motivation for looking at the grid-LSTM is to explore benefits of having separate LSTMs to model the correlations in time and frequency. The grid-LSTM in dimension j is given by the following equations at each time-frequency step (t, k):

$$\begin{aligned} i^{(j)}_{t,k} &= \sigma(W^{(j)}_{ix}x_{t,k} + W^{(t)}_{im}m^{(t)}_{t-1,k} + W^{(k)}_{im}m^{(k)}_{t,k-1} + \\ &\quad W^{(t)}_{ic}c^{(t)}_{t-1,k} + W^{(k)}_{ic}c^{(k)}_{t,k-1} + b^{(j)}_i) \end{aligned} \tag{11}$$

$$\begin{aligned} f^{(j)}_{t,k} &= \sigma(W^{(j)}_{fx}x_{t,k} + W^{(t)}_{mf}m^{(t)}_{t-1,k} + W^{(k)}_{mf}m^{(k)}_{k-1,t} + \\ &\quad W^{(t)}_{cf}c^{(t)}_{t-1,k} + W^{(k)}_{cf}c^{(k)}_{t,k-1} + b^{(j)}_f) \end{aligned} \tag{12}$$

$$\begin{aligned} c^{(t)}_{t,k} =& f^{(t)}_{t,k} \odot c^{(t)}_{t-1,k} + i^{(t)}_{t,k} \odot g(W^{(t)}_{cx}x_{t,k} + W^{(t)}_{cm}m^{(t)}_{t-1,k} + \\ & W^{(k)}_{cm}m^{(k)}_{t,k-1} + b^{(t)}_c) \end{aligned} \tag{13}$$

$$\begin{aligned} c^{(k)}_{t,k} =& f^{(k)}_{t,k} \odot c^{(k)}_{t,k-1} + i^{(k)}_{t,k} \odot g(W^{(k)}_{cx}x_{t,k} + W^{(t)}_{cm}m^{(t)}_{t-1,k} + \\ & W^{(k)}_{cm}m^{(k)}_{t,k-1} + b^{(k)}_c) \end{aligned} \tag{14}$$

$$\begin{aligned} o^{(j)}_{t,k} &= \sigma(W^{(j)}_{ox}x_{t,k} + W^{(t)}_{om}m^{(t)}_{t-1,k} + W^{(k)}_{om}m^{(k)}_{t,k-1} + \\ &\quad W^{(t)}_{oc}c^{(t)}_{t,k} + W^{(k)}_{oc}c^{(k)}_{t,k} + b^{(j)}_o) \end{aligned} \tag{15}$$

$$m^{(j)}_{t,k} = o^{(j)}_{t,k} \odot h(c^{(j)}_{t,k}) \tag{16}$$

In these equations, replacing j with t gives the gT-LSTM and with k gives the gF-LSTM. Besides the dimension dependent weight parameters $W^{(j)}_{**}$, the major difference from TF-LSTM is how each gate uses the previous output from both the gF-LSTM $m^{(k)}_{t,k-1}$ and gT-LSTM $m^{(t)}_{t-1,k}$. When the weights between two gT-LSTM and gF-LSTM are independent to each other, as in [10], the computation cost of grid-LSTM is slightly higher than TF-LSTM because of the separate weight matrices that multiply the cell states in both time and frequency. When the weight matrices are shared among the time and frequency cells (for example $W^{(t)}_{ic} = W^{(k)}_{ic}$ in Equation 11), the computation of the grid-LSTM becomes similar to that of TF-LSTMs. We will explore the performance of grid-LSTMs only with the shared state to keep computation similar to the TF-LSTM.

At each time step t, the output of the gT-LSTM, denoted by $\{m^{(t)}_{t,0}, \dots, m^{(t)}_{t,L}\}$ and gF-LSTM, denoted by $\{m^{(k)}_{t,0}, \dots, m^{(k)}_{t,L}\}$ are concatenated together and given to the linear dimensionality reduction layer, followed by the LDNN.

2.2.5. *ReNet-LSTM*

A ReNet LSTM [1] is an F-LSTM unrolled in frequency and T-LSTM unrolled in time. These two LSTMs are treated completely independently, meaning the recurrent state is not shared when the F-LSTM and T-LSTMs overlap. The benefit of ReNet is computational efficiency, as the two LSTMs can be run in parallel. Since the LSTMs produce independent outputs, we concatenate the outputs of both before giving them to a linear dimensionality-reduction layer and then the LDNN.

3. Experimental Details

Our single channel experiments are conducted on ∼2,000 hours of mild-noise training set consisting of 3 million English utterances. This data set is created by artificially corrupting clean utterances using a room simulator, adding varying degrees of noise and reverberation such that the overall SNR is between 5dB and 30dB. The noise sources are from YouTube and daily life noisy environmental recordings. To understand the behavior of 2D-LSTMs on different data sets, we also run experiments training on a moderate-noise 2,000-hour data set, where clean utterances are corrupted with both reverberation and additive noise at SNRs ranging from 0 to 20 dB. Finally, multichannel experiments are conducted on a training set with similar noise configurations to moderate, using a 2 channel linear mic with a 14cm spacing. Results are reported in matched conditions, meaning models trained in mild-noise conditions are evaluated on a mild-noise 20 hour test set of 30,000 utterances. All data sets are anonymized and hand-transcribed, and are representative of Google's voice search traffic.

All neural networks are trained with the cross-entropy criterion, using asynchronous stochastic gradient descent (ASGD) optimization [13]. The weights for all CNN and DNN layers are initialized using the Glorot-Bengio strategy described in [14], while all LSTM layers are uniform randomly initialized to be between -0.02 and 0.02.

4. Results

4.1. Analysis with Log-Mel Features

First, we compare LDNNs, CLDNNs, F-LSTMs and TF-LSTMs on the mild noise train/test set using log-mel features. Table 1 shows the WER of these models. First, notice that the

CLDNN outperforms the LDNN. Second, the F-LDNN, which has been argued to capture similar properties to convolution [1] matches the performance of the CLDNN. Finally, the TF-LSTM gives small improvements over the F-LSTM/CLDNN, showing that it is better to model correlations in both time and frequency.

Model	WER
LDNN	16.4
CLDNN	16.1
F-LDNN	16.0
TF-LDNN	**15.9**

Table 1: WER, Mild Noise

Second, we explore behavior of these models for the moderate noise train/test set, as shown in Table 2. First, notice that in the noisier conditions, the convolution layer in the CLDNN degrades performance compared to the LDNN. Since translational invariance in the CNN is captured through local filters and pooling, it is possible that as noise corrupts the signal, it becomes more difficult to make these local decisions. However, notice that the F-LDNN shows improvements over the CLDNN and LDNN. In fact, the 2% relative gain observed by the CLDNN over the LDNN for clean speech in Table 1, is achieved by the F-LDNN in noisier conditions. Since the F-LDNN captures translational invariance through the recurrent connections, it can better model the evolutionary path of the signal in frequency compared to the CNN. Finally, again notice that by modeling correlations in time and frequency with the TF-LDNN, we can achieve improvements over all other models. Compared to the baseline LDNN, the TF-LDNN has roughly a 3% relative improvement in WER. As stated before, to our knowledge this is the first thorough comparison between convolution, F-LSTMs and TF-LSTMs.

Method	WER
LDNN	23.0
CLDNN	23.4
F-LDNN	22.7
TF-LDNN	**22.3**

Table 2: WER, Moderate Noise

4.2. Alternative Architectures

In this section, we futher compare the TF-LSTM with other alternative architectures for modeling the time-frequency correlations, that have been explored for computer vision [1, 10] but never for speech. First, Table 3 shows that grid-LDNNs offer an additional 1% relative improvement in mild and moderate noise over the TF-LDNN, with a similar computational cost since the weights of the cell state are tied between the gT-LSTM and gF-LSTM. This shows that separate modeling of time and frequency correlations, while sharing the states, is important. However, performance with ReNet-LDNNs degrades for moderate noise conditions, showing the importance of sharing the state between T-LSTMs and F-LSTMs. We did not repeat the ReNet-LDNN experiment for mild noise given the degradation for moderate noise.

4.3. Performance with Learned Filterbank Features

Finally, we explore the behavior of grid LSTMs when we learn a filterbank directly from the raw-waveform. We specifically look at multichannel speech processing, where raw-waveform processing is necessary to preserve the fine time structure of the signal [15]. We explore grid LSTMs with the neural adaptive beamforming (NAB) model [11]. We refer the reader to [11] for a detailed description of the NAB model architecture, and describe it at a high-level only here. In the first layer of this model, for each frame, an "adaptive layer" takes a 35ms input raw-waveform signal for each channel and uses a T-LSTM to predict filter coefficients for each channel. These filters are convolved with the raw-waveform from each channel and the outputs are summed, mimicing filter and sum beamforming. Then, a "spectral decomposition" is performed on the single channel raw-waveform, using a time-convolution layer with max-pooling, very similar to [3]. This layer essentially learns a filterbank which is not constant-Q, and produces a 128-dimensional frequency feature at each frame. This feature is then given to a CLDNN/LDNN acoustic model.

Model	Moderate Noise	Mild Noise
LDNN	23.0	16.4
CLDNN	23.4	16.1
TF-LDNN	22.3	15.9
grid-LDNN	**22.1**	**15.7**
ReNet-LDNN	22.9	-

Table 3: WER For Different Architectures

We believe the NAB model suffers not only from the frequency convolution that sits above the spectral decomposition layer, not only because the learned filterbank is not constant-Q, but also because the filter into the spectral layer changes every frame because of the adaptive LSTM. Thus, we explore if further improvements can be obtained replacing the CLDNN or LDNN with a grid-LDNN.

Table 4 shows that the convolution layer in the CLDNN degrades performance. However, the grid-LDNN gives an 3% relative improvement in WER over the LDNN. This demonstrates that modeling time-frequency correlations with a 2D-LSTM is much more robust compared to convolution, even with a different feature representation.

	WER
LDNN	21.3
CLDNN	21.6
grid-LDNN	**20.7**

Table 4: WER with NAB model

5. Conclusions

In this paper, we presented different 2D-LSTM approaches (F-LSTM, TF-LSTM, grid-LSTM, ReNet) to model time-frequency correlations, and compared this to convolutional layers. We found that grid-LSTMs in the LDNN architecture (grid-LDNN) offered the best performance of all 2D-LSTMs, particularly when the input is noisy or the filterbank is learned. Overall, the grid-LDNN shows between a 1-4% relative improvement over an LDNN and CLDNN for a mild noise task, moderate noise task, and multichannel task.

6. Acknowledgements

The authors would like to thank Eugene Brevdo, Oriol Vinyals and Rafal Jozefowicz for discussions related to 2D-LSTMs.

7. References

[1] F. Visin, K. Kastner, K. Cho, M. Matteucci, A. C. Courville, and Y. Bengio, “Renet: A recurrent neural network based alternative to convolutional networks,” *CoRR*, vol. abs/1505.00393, 2015.

[2] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, “Convolutional, Long Short-Term Memory, Fully Connected Deep Neural Networks,” in *Proc. ICASSP*, 2015.

[3] T. N. Sainath, R. J. Weiss, K. W. Wilson, A. Senior, and O. Vinyals, “Learning the Speech Front-end with Raw Waveform CLDNNs,” in *Proc. Interspeech*, 2015.

[4] T. N. Sainath, B. Kingsbury, A. Mohamed, G. Dahl, G. Saon, H. Soltau, T. Beran, A. Aravkin, and B. Ramabhadran, “Improvements to Deep Convolutional Neural Networks for LVCSR,” in *Proc. ASRU*, 2013.

[5] H. Sak, A. Senior, and F. Beaufays, “Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling,” in *Proc. Interspeech*, 2014.

[6] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, “Deep Neural Networks for Acoustic Modeling in Speech Recognition,” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[7] J. Li, A. Mohamed, G. Zweig, and Y. Gong, “LSTM Time and Frequency Recurrence for Automatic Speech Recognition,” in *Proc. ASRU*, 2015.

[8] A. Graves, S. Fernandez, and J. Schmidhuber, “Multi-Dimensional Recurrent Neural Networks,” in *Proc. ICANN*, 2007.

[9] J. Li, A. Mohamed, G. Zweig, and Y. Gong, “Exploring Multidimensional LSTMs for Large Vocabulary ASR,” in *Proc. ICASSP*, 2016.

[10] N. Kalchbrenner, I. Danihelka, and A. Graves, “Grid Long Short-Term Memory,” in *to appear in Proc. ICLR*, 2016.

[11] B. Li, T. N. Sainath, R. J. Weiss, K. W. Wilson, and M. Bacchiani, “Neural Network Adaptive Beamforming for Robust Multichannel Speech Recognition,” in *to appear in Proc. Interspeech*, 2016.

[12] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735 – 1780, 1997.

[13] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. Le, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Ng, “Large Scale Distributed Deep Networks,” in *Proc. NIPS*, 2012.

[14] X. Glorot and Y. Bengio, “Understanding the Difficulty of Training Deep Feedforward Neural Networks,” in *Proc. AISTATS*, 2014.

[15] T. N. Sainath, R. J. Weiss, K. W. Wilson, A. Narayanan, and M. Bacchiani, “Factored Spatial and Spectral Multichannel Raw Waveform CLDNNs,” in *in Proc. ICASSP*, 2016.