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

Reducing the Computational Complexity of Two-Dimensional LSTMs

Bo Li, Tara N. Sainath

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

Abstract

Long Short-Term Memory Recurrent Neural Networks (LSTMs) are good at modeling temporal variations in speech recognition tasks, and have become an integral component of many state-of-the-art ASR systems. More recently, LSTMs have been extended to model variations in the speech signal in two dimensions, namely time and frequency [1, 2]. However, one of the problems with two-dimensional LSTMs, such as Grid-LSTMs, is that the processing in both time and frequency occurs sequentially, thus increasing computational complexity. In this work, we look at minimizing the dependence of the Grid-LSTM with respect to previous time and frequency points in the sequence, thus reducing computational complexity. Specifically, we compare reducing computation using a bidirectional Grid-LSTM (biGrid-LSTM) with non-overlapping frequency sub-band processing, a PyraMiD-LSTM [3] and a frequency-block Grid-LSTM (fbGrid-LSTM) for parallel time-frequency processing. We find that the fbGrid-LSTM can reduce computation costs by a factor of four with no loss in accuracy, on a 12,500 hour Voice Search task.

1. Introduction

Sequence modeling tasks often utilize Long Short-Term Memory Recurrent Neural Network (LSTMs) [4] architectures. For example in speech recognition, LSTMs are most commonly used to model temporal variations, and have produced state-of-the-art results on a variety of tasks [5, 6].

Recently, time-LSTMs have been extended to model the speech signal in both time and frequency [1, 2]. Two dimensional (2D) LSTMs [7] are unrolled across both time and frequency, and can model both spectral and temporal variations through local filters and recurrent connections. In fact, 2D-LSTMs have been shown to outperform CNNs on a Voice Search task [2].

However, one of the issues with 2D-LSTMs is that at any given time frame, the LSTM is unrolled in frequency and the recurrent state is passed between unrolled steps (i.e., sub-bands). Thus 2D-LSTMs are much slower compared to time-LSTMs which are not unrolled in frequency. In addition, 2D-LSTMS are also slower than CNNs, where the convolution in each frequency sub-band does not depend on other sub-bands and can be parallelized.

The purpose of this paper is to look at improving the speed of 2D-LSTMs without sacrificing loss in word error rate (WER). For the purposes of this paper, we focus on a variant of 2D-LSTMs known as Grid-LSTMs, which have shown superior performance compared to other 2D-LSTM methods [2]

The first method we look at for speeding up Grid-LSTMs is to run a bi-directional Grid-LSTM (biGrid-LSTM). Typically, when the Grid-LSTM is unrolled in frequency, we stride it such that the filter overlaps with the previous sub-band [2]. As we will show in this paper, if we make the stride too large, we can reduce computational complexity, but also degrade the WER since we cannot capture frequency variations well. However, since the LSTM spatial processing introduces directionality when it is unrolled and the recurrent state is passed, running the Grid-LSTM from lower frequency sub-band towards the higher ones should generate different activations when unrolling from higher sub-bands to the lower ones. Thus, we hypothesize that with a biGrid-LSTM, we could potentially increase the stride size and still model frequency variations appropriately. More importantly, the two Grid-LSTMs could be run in parallel, which would not incurr extra time delay with sufficient computational power compared to uni-directional Grid-LSTM. To maintain the model's capability of online processing, we only adopt the bi-directional modeling in the frequency direction, not in the time direction.

Another approach introduced in the literature to mitigate the computational complexity of unrolling the Grid-LSTM in frequency is a PyraMiD-LSTM [3]. In this approach, at each time-frequency block, instead of taking in the LSTM state from the previous frequency block of the current time frame, the frequency block of the previous time frame is used. In this case, all the blocks of the current window can be computed in parallel with the compromise of the delayed time information.

Finally, we propose a frequency block Grid-LSTM (fbGrid-LSTM) method to reduce computation. The passing of the recurrent state as the Grid-LSTM is unrolled in frequency helps to model frequency variations. However, the frequency variations we need to model occur only within a small sub-band range [8], akin to the small range that convolutional filters are pooled over in a CNN. Therefore, we explore if we can run multiple Grid-LSTMs on smaller frequency sub-bands, where we only unroll the LSTM to the length of the smaller sub-band. This allows us to run multiple Grid-LSTMs in parallel, thereby reducing computational complexity. Our experiments comparing different Grid-LSTM speed-ups are conducted on a 12,500 hour Voice Search task. We find that the fbGrid-LSTM allows for a reduction in parallel computation cost of more than 70% relative with no loss in accuracy. We also find that while the biGrid-LSTM and the PyraMiD-LSTM can reduce computation, they also degrade the performance.

The rest of this paper is as follows. In Section 2 we describe the various Grid-LSTM speed-up methods. The experimental setup is described in Section 3 and results comparing the different speed-up methods are presented in Section 4. Finally, Section 5 concludes the paper and discusses future work.

2. Architecture Description

This section describes the various Grid-LSTM speed-up architectures investigated in this paper. Note that in this paper, the first layer of our model is a Grid-LSTM. The output of this is fed to a LDNN architecture [6], consisting of time-LSTM and DNN layers.

http://dx.doi.org/10.21437/Interspeech.2017-1164

2.1. LSTM

We review the basic LSTM model as described in [4] for easy comparisons with the Grid-LSTMs in the following sections. The LSTM consists of a set of recurrently connected subnetworks, referred to as *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 the 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. In this study, we remove the peephole connections as we didn't find gains by using them. Specifically, at each time step t, the LSTM model is given by:

$$q_u = W_{um} m_{t-1}, \quad u \in \{i, f, c, o\} \tag{1}$$
$$i_t = \sigma(W_{ix} x_t + q_i + b_i) \tag{2}$$
$$f_t = \sigma(W_{fx} x_t + q_f + b_f) \tag{3}$$
$$c_t = f_j \odot c_{t-1} + i_t \odot g(W_{cx} x_t + q_c + b_c) \tag{4}$$
$$o_t = \sigma(W_{ox} x_t + q_o + b_o) \tag{5}$$
$$m_t = o_t \odot h(c_t) \tag{6}$$

where i_t, f_t, c_t and o_t denote the input, forget, memory cell and output gate activations at step t. m_t is the output of the LSTM layer. W_{**} are the different weight matrices, for example W_{ix} is the weight matrix from the feature to the input gate. $\odot$ is an element-wise dot product. Finally, σ is the logistic sigmoid nonlinearity 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, each step corresponds to a frame.

2.2. Grid-LSTM

A Grid-LSTM [9] is very similar to a TF-LSTM [1] except that there are separate LSTMs which move in time and frequency. However, at a time-frequency bin, the grid frequency LSTM (gF-LSTM) also uses the state of the grid time LSTM (gT-LSTM) from the previous timestep, and the same to the gT-LSTM. The motivation for looking at the Grid-LSTM is to explore benefits of modeling the correlations in time and frequency. The Grid-LSTM in dimension $s \in \{t, k\}$ is given by the following equations at each time-frequency step (t, k):

$$q_{u,t,k} = W_{um}^{(t)} m_{t-1,k}^{(t)} + W_{um}^{(k)} m_{t,k-1}^{(k)}, \quad u \in \{i, f, c, o\} \tag{7}$$
$$i_{t,k}^{(s)} = \sigma(W_{ix}^{(s)} x_{t,k} + q_{i,t,k} + b_i^{(s)}) \tag{8}$$
$$f_{t,k}^{(s)} = \sigma(W_{fx}^{(s)} x_{t,k} + q_{f,t,k} + b_f^{(s)}) \tag{9}$$
$$c_{t,k}^{(t)} = f_{t,k}^{(t)} \odot c_{t-1,k}^{(t)} + i_{t,k}^{(t)} \odot g(W_{cx}^{(t)} x_{t,k} + q_{c,t,k} + b_c^{(t)}) \tag{10}$$
$$c_{t,k}^{(k)} = f_{t,k}^{(k)} \odot c_{t,k-1}^{(k)} + i_{t,k}^{(k)} \odot g(W_{cx}^{(k)} x_{t,k} + q_{c,t,k} + b_c^{(k)}) \tag{11}$$
$$o_{t,k}^{(s)} = \sigma(W_{ox}^{(s)} x_{t,k} + q_{o,t,k} + b_o^{(s)}) \tag{12}$$
$$m_{t,k}^{(s)} = o_{t,k}^{(s)} \odot h(c_{t,k}^{(s)}) \tag{13}$$

In these equations, replacing s with t gives the gT-LSTM and with k gives the gF-LSTM. Besides the dimension dependent weight parameters $W_{**}^{(s)}$, the major difference from TF-LSTM is that each gate uses the previous output from both the gF-LSTM $m_{t,k-1}^{(k)}$ and gT-LSTM $m_{t-1,k}^{(t)}$. In this paper, similar to [2] we explore sharing the weight matrices between the time and frequency cells (for example $W_{ix}^{(t)} = W_{ix}^{(k)}$ in Equation (8)) to limit the increase in the number of model parameters.

To extract features using the Grid-LSTM to feed to the LDNN, we follow the same approach used in [2]. Specifically, given an input feature vector $v_t \in \Re^N$, we window the first F elements from this feature, denoted by $x_{t,0} = v_t^{0:F} \in \Re^F$ and give this as input to the Grid-LSTM. At the next step, we stride the window over the input by S, and take the next F features, denoted by $x_{t,1} = v_t^{S:(F+S)} \in \Re^F$, and pass this to the Grid-LSTM. Hence the input to the LSTM (Equations (2) - (6)) at each time step t and frequency step k is given by $x_{t,k} = v_t^{k*S:(F+k*S)}$. In most cases $S < F$ so the chunks have overlapping information. The Grid-LSTM is thus unrolled over frequency by an amount $L = (N - F)/S + 1$.

At each time step t, the output of the gT-LSTM, denoted by $\{m_{t,0}^{(t)}, \ldots, m_{t,L-1}^{(t)}\}$ and gF-LSTM, denoted by $\{m_{t,0}^{(k)}, \ldots, m_{t,L-1}^{(k)}\}$ are concatenated together and given to a linear dimensionality reduction layer, followed by the LDNN.

2.3. Bi-directional Grid-LSTM (biGrid-LSTM)

The bi-directional Grid-LSTM (biGrid-LSTM) consists of a forward Grid-LSTM and a backward Grid-LSTM. The forward processing ('`fwd`') is extactly the same as the Grid-LSTM shown in Equations (7) - (13). While for the backward processing ('`bwd`'), instead of using the previous frequency block's, *i.e.* $(k-1)$-th, LSTM output $m_{t,k-1}^{(bwd,k)}$ and cell state $c_{t,k-1}^{(bwd,k)}$, the next frequency block's, *i.e.* $(k+1)$-th, LSTM output $m_{t,k+1}^{(bwd,k)}$ and cell state $c_{t,k+1}^{(bwd,k)}$ are used. Specifically, the corresponding equations for Equations (7) and Equation (11) in the backward processing are:

$$q_{u,t,k}^{(\texttt{bwd})} = W_{um}^{(\texttt{bwd},t)} m_{t-1,k}^{(bwd,t)} + W_{um}^{(\texttt{bwd},k)} m_{t,k+1}^{(bwd,k)} \tag{14}$$
$$c_{t,k}^{(\texttt{bwd},k)} = f_{t,k}^{(\texttt{bwd},k)} \odot c_{t,k+1}^{(\texttt{bwd},k)} + i_{t,k}^{(\texttt{bwd},k)} \odot g(W_{cx}^{(\texttt{bwd},k)} x_{t,k} + q_{c,t,k}^{(\texttt{bwd})} + b_c^{(\texttt{bwd},k)}) \tag{15}$$

with others simply replacing the weight parameters with backward ones. The final output at each time-frequency block (t, k) is a concatenation of the forward and backward activations:

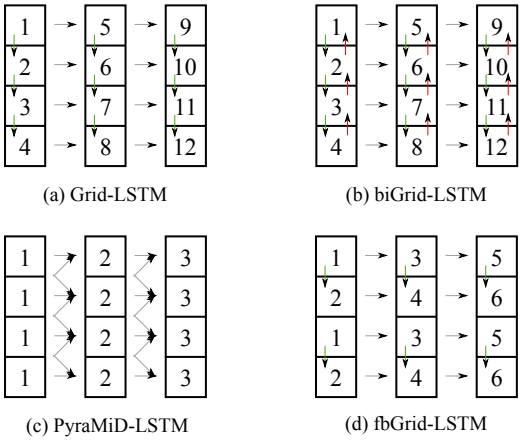
$$m_{t,k}^{(s)} = [m_{t,k}^{(\texttt{fwd},s)T} \quad m_{t,k}^{(\texttt{bwd},s)T}]^T \tag{16}$$

Thus, at each time step t, we concatenate the output of the biGrid-LSTM $m_{t,k}^{(s)}$ for all $k \in \{0 \ldots L-1\}$, namely $\{m_{t,0}^{(s)}, \ldots, m_{t,L-1}^{(s)}\}$ and give it to a linear dimensionality reduction layer, followed by the LDNN.

2.4. PyraMiD-LSTM

In the PyraMiD-LSTM [3], the frequency dependency is on the previous time frame instead of the current time frame. Hence, at each time frame, different frequency sub-bands are independent to each other and there is not state passing across frequency sub-bands. The time and frequency interaction is modelled through concatenating the neighbouring sub-bands at previous time frame as inputs. This enables the parallel processing of different frequency sub-bands at the current time step. Different from other variants, there is no actual LSTM unrolling in the frequency direction. Specifically, the main difference from Grid-LSTM is Equation (7), which for the PyraMiD-LSTM is:

$$q_{u,t,k} = W_{um}^{(k-1)} m_{t-1,k-1} + W_{um}^{(k)} m_{t-1,k} + W_{um}^{(k+1)} m_{t-1,k+1} \tag{17}$$



(a) Grid-LSTM (b) biGrid-LSTM

(c) PyraMiD-LSTM (d) fbGrid-LSTM

Figure 1: *Comparisons of processing flows of different Grid-LSTM variations. The black arrows indicate time dependence and the the green and red ones are for the forward and backward frequency dependence.*

Besides, no superscript $s \in \{t, k\}$ is needed for PyraMiD-LSTM. Similarly, for the output we concatenate the PyraMiD-LSTM $m_{t,k}$ at all $k \in \{0 \dots L-1\}$ sub-bands, namely $\{m_{t,0}, \dots, m_{t,L-1}\}$ and give it to the linear dimensionality reduction layer, followed by the LDNN [6].

2.5. Frequency Block Grid-LSTM (fbGrid-LSTM)

As discussed in [10], the passing of the recurrent state as the grid LSTM unrolled in frequency models frequency variations, and is akin to pooling in convolution. In speech recognition, the behavior in low-frequency regions and high-frequency regions is very different, and therefore it is really within a small frequency region we care about reducing variations. Thus, we explore if it is necessary to pass the recurrent state along the entire frequency space, or if we can unroll in smaller frequency blocks. This gives us the benefit that we can run multiple Grid-LSTMs in parallel. For each block $b \in \{0 \dots B-1\}$, the Grid-LSTM computation represented by Equations (7) - (13) is applied and different blocks have different weight parameters.

For the fbGrid-LSTM case, at each time step t, we concatenate the output of the fbGrid-LSTM $m_{t,k}^{(s,b)}$ for all blocks $b \in \{0 \dots B-1\}$ and all $k \in \{0 \dots L_b - 1\}$, namely $\{m_{t,0}^{(s,0)}, \dots, m_{t,L_0-1}^{(s,0)}, \dots m_{t,0}^{(s,B-1)}, \dots, m_{t,L_{B-1}-1}^{(s,B-1)}\}$. This concatenated output is given to a linear dimensionality reduction layer, followed by the LDNN [6].

2.6. Processing Flow comparison

To have an intuitive understanding of the speed-up brought by the different Grid-LSTM variations, the processing flow of different architectures are illustrated in Figure 1. For standard LSTM, the processing steps only depends on the number of time frames as it only unrolls in time. While for Grid-LSTM (Figure 1 (a)), the unrolling in both time and frequency greatly increases the number of processing steps. By replacing the dependency on the current frame to the previous frame, PyraMiD-LSTM has the same processing steps as the standard LSTM. With fbGrid-LSTM (Figure 1 (d)), the number of processing steps reduces to the $1/B$-th of the basic Grid-LSTM, where B is the total number of blocks used.

3. Experimental Details

Our experiments are conducted on about 12,500 hours of noisy training data consisting of 15 million English utterances. This data set is created by artificially corrupting clean utterances using a room simulator, adding varying degrees of noise and reverberation. The clean utterances are anonymized and hand-transcribed voice search queries, and are representatives of Google's voice search traffic. Noise signals, which include music and ambient noise sampled from YouTube and recordings of "daily life" environments, are added to the clean utterances at SNRs ranging from 0 to 20 dB, with an average of about 12 dB. Reverberation is simulated using the image model [11] with room dimensions and microphone array positions that are randomly sampled from 100 possible room configurations with T_{60}s ranging from 400 to 900 ms, with an average of about 600 ms. Our test set consists of a separate set of about 30,000 utterances (over 20 hours). We artificially add noise to the clean utterances using a room simulator. The degree of noise and reverberation is similar to the training set.

Similar to the low frame rate (LFR) experiments in [12]. The acoustic features used for all experiments are 80-dimensional log-mel filterbank energies, computed using a 25ms window every 10ms. At the current frame t, these features are stacked with 2 frames to the left and downsampled at a 30ms frame rate, to produce a 240-dimensional feature vector. All networks consist of a Grid-LSTM layer variant, which is followed by an LDNN model [6]. The LDNN consists of a low-rank layer, followed by 5 time-LSTM layers with 700 cells, then one DNN layer with 1,024 hidden units, and finally a softmax layer with 8,192 context-dependent phone output targets. All networks are trained with the cross-entropy criterion, using asynchronous stochastic gradient descent (ASGD) [13]. The weights for DNN layers are initialized using the Glorot-Bengio strategy described in [14], while all LSTM parameters are uniformly initialized to lie between -0.02 and 0.02. We use an exponentially decaying learning rate, which starts at 1.5e-4 and has a decay rate of 0.9 over 1 billion frames.

4. Results

In this section, we present our experimental results comparing different Grid-LSTM variations.

4.1. Grid-LSTM

First, we investigate the effect of filter size F, stride size S and LSTM cell size C on the standard uni-directional Grid-LSTM. Table 1 shows the change in WER as we vary the number of cells C, keeping both the filter size F and stride S constant. Notice that increasing the cells from 64 to 128 reduces WER, though comes at a cost of increased multiplies and adds (M+A).

One way to reduce computation is to increase the filter stride S. Table 1 shows that for a cell size of $C = 128$, increasing the stride to 8, we can reduce the computational cost by a factor of four but with small loss in accuracy. When we increase the stride to 16, we take a larger hit in accuracy. As the stride controls the frequency variation we can model, it makes sense that increasing the stride to be too large will degrade performance.

4.2. biGrid-LSTM

As the previous section showed, increasing the stride size S can reduce computation cost, but comes at the cost of increas-

Table 1: *WER for Grid-LSTMs with different configurations (F, S and C for filter, stride and LSTM cell sizes).*

Model			M+A	WER
F	S	C	(M)	
16	2	64	6.0	19.0
16	2	96	12.2	18.9
16	2	128	20.6	18.3
16	8	128	5.6	18.6
16	16	128	3.1	19.2

ing WER. To address the degradation, we investigate the use of bi-directional processing for the gF-LSTM of the Grid-LSTM, which we hope can make up for the performance degradation when the stride is large by running an LSTM from low-to-high frequency and the other way around. One benefit of the biGrid-LSTM is that the two gF-LSTMs can be run in parallel. Table 2 shows the biGrid-LSTM performance for different choices of stride S. Firstly, with non-overlapping filters of size 8 and 16, performance degradation was observed for both cases. Filter size 8 is much better than 16, which seems to suggest smaller strides and filters are preferred for biGrid-LSTM. So far the biGrid-LSTM does the same windowing of inputs for the two different directions. Alternatively, we can shift the backward processing a bit to explicitly incur more variations during training. The experimental result (Opt. "O4" in Table 2) shows that simple offseting is not sufficient to improve the robustness. Lastly, we un-tie the gT-LSTM and gF-LSTM (Opt. "TF" in Table 2), which doubles the number of parameters but has the same computation cost. However the performance degradation is still not resloved. These experiments indicate that the use of bi-directional processing in frequency direction is not sufficient to enable non-overlapping filter processing on our dataset.

Table 2: *WER for bi-directional Grid-LSTMs with different configurations (F, S and C for filter, stride and LSTM cell sizes).*

Model				M+A (M)		WER
F	S	C	Opt.	Total	Parallel	
16	16	128	-	6.2	3.1	20.3
8	8	128	-	11.5	5.7	18.7
8	8	128	O4	11.5	5.7	19.9
8	8	128	TF	11.5	5.7	19.2

4.3. PyraMiD-LSTM

Next, we explore the performance of PyraMiD-LSTM with non-overlapping filters. With ($F = 4, S = 4, C = 64$) and ($F = 8, S = 8, C = 128$), we could achieve a huge computation reduction, but WERs increased a lot and smaller stride seems more preferrable for PyraMiD-LSTM. With the same configruation as the Grid-LSTM, namely $F = 16, S = 2, C = 128$, there is still an 8% relative increase in WER. A possible explanation is that without frequency direction unrolling, PyraMiD-LSTM can only model limited variations in neighbouring sub-bands. Specially, variations within only the previous, current and next frequency sub-bands are captured besides the time-delayed states. To address this, we can increase the length of the frequency dependence window. However, in that way, the extreme case would be to take in the whole frame vector, which degenerates to a time-LSTM. Instead, we borrowed the idea of reducing frequency sub-band dependence and proposed an fbGrid-LSTM, which will be discussed next.

Table 3: *WER for PyraMiD-LSTMs with different configurations (F, S and C for filter, stride and LSTM cell sizes).*

Model			M+A (M)		WER
F	S	C	Total	Parallel	
4	4	64	2.9	0.4	20.6
8	8	128	5.7	1.7	21.2
16	2	128	20.6	1.9	19.8

4.4. fbGrid-LSTM

Finally, we explore the performance of splitting the frequency bands into independent blocks and run Grid-LSTM on each block in parallel, known as a fbGrid-LSTM. While this does not reduce the overall number of computations, it allows us to run multiple Grid-LSTMs in each frequency block in parallel. Experimental results for different configurations of the fbGrid-LSTM are tabulated in Table 4. The table shows that if we split the frequency axis into 4 independent blocks and run a separate Grid-LSTM in each one ($F = 10$), we take a small hit in WER compared to not splitting the frequency axis at all. This is likely because making hard decisions on where to split the frequency axis means we cannot model frequency variations across these splits. However, if we split the axis into 4 blocks such that there is overlap between the blocks ($F = 16$), we can achieve similar performance to not splitting the frequency axis at all, with a 70% relative speed-up as the 4 blocks can be run in parallel. Notice however if we increase the number of blocks to be too large (i.e., $B = 7$) such that we unroll the LSTM state less in each block, we cannot model as much frequency variation since we pass the recurrent state less, and hence the WER increases.

Table 4: *WER for frequency block Grid-LSTMs with different configurations (B, F, S and C for number of blocks, filter, stride and LSTM cell sizes).*

Model				M+A (M)		WER
B	F	S	C	Total	Parallel	
1	16	2	128	20.6	20.6	18.3
4	10	2	128	14.1	3.5	18.9
4	16	2	128	22.5	5.6	18.0
7	15	1	128	25.9	3.7	18.3

5. Conclusions

Time-frequency modeling using LSTMs, especially Grid-LSTMs, have been shown to yield better robustness. In this paper, we further investigated several Grid-LSTM variations, namely bi-directional Grid-LSTM, PyraMiD-LSTM and freuqency block Grid-LSTM (fbGrid-LSTM). Experimental results have shown that we can improve the Grid-LSTM's parallelization capability with independent frequency block processing, namely the proposed frequency block Grid-LSTM. It eliminates the frequency sub-band dependence and processes each block with a Grid-LSTM in parallel. The fbGrid-LSTM has been shown to not only reduce the computation cost by a factor of four, but also to achieve slightly better recognition performance compared to the conventional Grid-LSTM.

6. References

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

[2] T. N. Sainath and B. Li, "Modeling Time-Frequency Patterns with LSTM vs. Convolutional Architectures for LVCSR Tasks," in *Proc. Interspeech*, 2016.

[3] M. F. Stollenga, W. Byeon, M. Liwicki, and J. Schmidhuber, "Parallel Multi-Dimensional LSTM, With Application to Fast Biomedical Volumetric Image Segmentation," in *Proc. NIPS*, 2015.

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

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

[7] A. Graves, S. Fernández, and J. Schmidhuber, "Multi-dimensional recurrent neural networks," in *in Proc. CoRR*, 2007.

[8] J. B. Allen, "How do humans process and recognize speech?" *IEEE Transactions on speech and audio processing*, vol. 2, no. 4, pp. 567–577, 1994.

[9] N. Kalchbrenner, I. Danihelka, and A. Graves, "Grid Long Short-Term Memory," in *in Proc. ICLR*, 2016.

[10] F. Visin, K. Kastner, K. Cho, M. Matteucci, A. C. Courville, and Y. Bengio, "ReNet: A Recurrent Neural Network Based Alternative to Convolutional Networks," in *in Proc. CoRR*, 2015.

[11] J. B. Allen and D. A. Berkley, "Image Method for Efficiently Simulation Room-Small Acoustics," *Journal of the Acoustical Society of America*, vol. 65, no. 4, pp. 943 – 950, April 1979.

[12] G. Pundak and T. N. Sainath, "Lower Frame Rate Neural Network Acoustic Models," in *Proc. Interspeech*, 2016.

[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.