Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling

Haşim Sak, Andrew Senior, Françoise Beaufays

Google, USA

{hasim,andrewsenior,fsb@google.com}

Abstract

Long Short-Term Memory (LSTM) is a specific recurrent neural network (RNN) architecture that was designed to model temporal sequences and their long-range dependencies more accurately than conventional RNNs. In this paper, we explore LSTM RNN architectures for large scale acoustic modeling in speech recognition. We recently showed that LSTM RNNs are more effective than DNNs and conventional RNNs for acoustic modeling, considering moderately-sized models trained on a single machine. Here, we introduce the first distributed training of LSTM RNNs using asynchronous stochastic gradient descent optimization on a large cluster of machines. We show that a two-layer deep LSTM RNN where each LSTM layer has a linear recurrent projection layer can exceed state-of-the-art speech recognition performance. This architecture makes more effective use of model parameters than the others considered, converges quickly, and outperforms a deep feed forward neural network having an order of magnitude more parameters.

Index Terms: Long Short-Term Memory, LSTM, recurrent neural network, RNN, speech recognition, acoustic modeling.

1. Introduction

Speech is a complex time-varying signal with complex correlations at a range of different timescales. Recurrent neural networks (RNNs) contain cyclic connections that make them a more powerful tool to model such sequence data than feed-forward neural networks. RNNs have demonstrated great success in sequence labeling and prediction tasks such as handwriting recognition and language modeling. In acoustic modeling for speech recognition, however, where deep neural networks (DNNs) are the established state-of-the-art, recently RNNs have received little attention beyond small scale phone recognition tasks, notable exceptions being the work of Robinson [1], Graves [2], and Sak [3].

DNNs can provide only limited temporal modeling by operating on a fixed-size sliding window of acoustic frames. They can only model the data within the window and are unsuited to handle different speaking rates and longer term dependencies. By contrast, recurrent neural networks contain cycles that feed the network activations from a previous time step as inputs to the network to influence predictions at the current time step. These activations are stored in the internal states of the network which can in principle hold long-term temporal contextual information. This mechanism allows RNNs to exploit a dynamically changing contextual window over the input sequence history rather than a static one as in the fixed-sized window used with feed-forward networks. In particular, the *Long Short-Term Memory* (LSTM) architecture [4], which overcomes some modeling weaknesses of RNNs [5], is conceptually attractive for the task of acoustic modeling.

LSTM and conventional RNNs have been successfully applied to various sequence prediction and sequence labeling tasks. In language modeling, a conventional RNN has obtained significant reduction in perplexity over standard n-gram models [6] and an LSTM RNN model has shown improvements over conventional RNN LMs [7]. LSTM models have been shown to perform better than RNNs on learning context-free and context-sensitive languages [8]. Bidirectional LSTM (BLSTM) networks that operate on the input sequence in both directions to make a decision for the current input have been proposed for phonetic labeling of acoustic frames on the TIMIT speech database [9]. For online and offline handwriting recognition, BLSTM networks used together with a Connectionist Temporal Classification (CTC) layer and trained from unsegmented sequence data, have been shown to outperform a state-of-the-art Hidden-Markov-Model (HMM) based system [10]. Similar techniques with a deep BLSTM network have been proposed to perform grapheme-based speech recognition [11]. BLSTM networks have also been proposed for phoneme prediction in a multi-stream framework for continuous conversational speech recognition [12]. In terms of architectures, following the success of DNNs for acoustic modeling [13, 14, 15, 16], a deep BLSTM RNN combined with a CTC output layer and an RNN transducer predicting phone sequences has been shown to reach state-of-the-art phone recognition accuracy on the TIMIT database [17].

Deep BLSTM RNNs have recently been shown to perform better than DNNs in the hybrid speech recognition approach [2]. Using the hybrid approach, we have recently shown that an LSTM architecture with a recurrent projection layer outperforms DNNs and conventional RNNs for large vocabulary speech recognition, considering moderately-sized models trained on a single machine [3]. In this paper, we explore LSTM RNN architectures for large-scale acoustic modeling using distributed training. We show that a two-layer deep LSTM RNN where each LSTM layer has a linear recurrent projection layer outperforms a strong baseline system using a deep feed-forward neural network having an order of magnitude more parameters.

2. LSTM Network Architectures

2.1. Conventional LSTM

The LSTM contains special units called *memory blocks* in the recurrent hidden layer. The memory blocks contain memory cells with self-connections storing the temporal state of the network in addition to special multiplicative units called gates to control the flow of information. Each memory block in the original architecture contained an *input gate* and an *output gate*. The input gate controls the flow of input activations into the

Copyright © 2014 ISCA

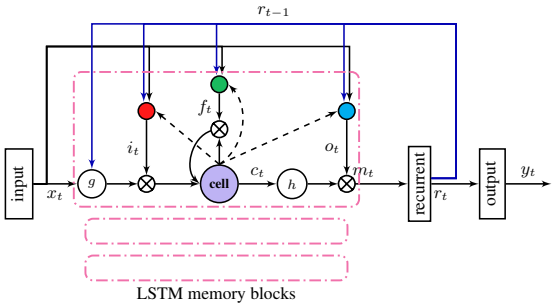


Figure 1: LSTMP RNN architecture. A single memory block is shown for clarity.

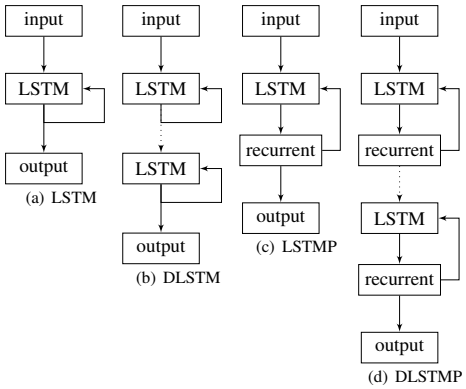


Figure 2: LSTM RNN architectures.

memory cell. The output gate controls the output flow of cell activations into the rest of the network. Later, the *forget gate* was added to the memory block [18]. This addressed a weakness of LSTM models preventing them from processing continuous input streams that are not segmented into subsequences. The forget gate scales the internal state of the cell before adding it as input to the cell through the self-recurrent connection of the cell, therefore adaptively forgetting or resetting the cell's memory. In addition, the modern LSTM architecture contains *peephole connections* from its internal cells to the gates in the same cell to learn precise timing of the outputs [19].

An LSTM network computes a mapping from an input sequence $x = (x_1, ..., x_T)$ to an output sequence $y = (y_1, ..., y_T)$ by calculating the network unit activations using the following equations iteratively from $t = 1$ to T:

$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1} + W_{ic}c_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{fx}x_t + W_{fm}m_{t-1} + W_{fc}c_{t-1} + b_f) \quad (2)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g(W_{cx}x_t + W_{cm}m_{t-1} + b_c) \quad (3)$$

$$o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1} + W_{oc}c_t + b_o) \quad (4)$$

$$m_t = o_t \odot h(c_t) \quad (5)$$

$$y_t = \phi(W_{ym}m_t + b_y) \quad (6)$$

where the W terms denote weight matrices (e.g. W_{ix} is the matrix of weights from the input gate to the input), W_{ic}, W_{fc}, W_{oc} are diagonal weight matrices for peephole connections, the b terms denote bias vectors (b_i is the input gate bias vector), σ is the logistic sigmoid function, and i, f, o and c are respectively the input gate, forget gate, output gate and cell activation vectors, all of which are the same size as the cell output activation vector m, $\odot$ is the element-wise product of the vectors, g and h are the cell input and cell output activation functions, generally and in this paper $tanh$, and ϕ is the network output activation function, softmax in this paper.

2.2. Deep LSTM

As with DNNs with deeper architectures, deep LSTM RNNs have been successfully used for speech recognition [11, 17, 2]. Deep LSTM RNNs are built by stacking multiple LSTM layers. Note that LSTM RNNs are already deep architectures in the sense that they can be considered as a feed-forward neural network unrolled in time where each layer shares the same model parameters. One can see that the inputs to the model go through multiple non-linear layers as in DNNs, however the features from a given time instant are only processed by a single nonlinear layer before contributing the output for that time instant. Therefore, the depth in deep LSTM RNNs has an additional meaning. The input to the network at a given time step goes through multiple LSTM layers in addition to propagation through time and LSTM layers. It has been argued that deep layers in RNNs allow the network to learn at different time scales over the input [20]. Deep LSTM RNNs offer another benefit over standard LSTM RNNs: They can make better use of parameters by distributing them over the space through multiple layers. For instance, rather than increasing the memory size of a standard model by a factor of 2, one can have 4 layers with approximately the same number of parameters. This results in inputs going through more non-linear operations per time step.

2.3. LSTMP - LSTM with Recurrent Projection Layer

The standard LSTM RNN architecture has an input layer, a recurrent LSTM layer and an output layer. The input layer is connected to the LSTM layer. The recurrent connections in the LSTM layer are directly from the cell output units to the cell input units, input gates, output gates and forget gates. The cell output units are also connected to the output layer of the network. The total number of parameters N in a standard LSTM network with one cell in each memory block, ignoring the biases, can be calculated as $N = n_c \times n_c \times 4 + n_i \times n_c \times 4 + n_c \times n_o + n_c \times 3$, where n_c is the number of memory cells (and number of memory blocks in this case), n_i is the number of input units, and n_o is the number of output units. The computational complexity of learning LSTM models per weight and time step with the stochastic gradient descent (SGD) optimization technique is $O(1)$. Therefore, the learning computational complexity per time step is $O(N)$. The learning time for a network with a moderate number of inputs is dominated by the $n_c \times (4 \times n_c + n_o)$ factor. For the tasks requiring a large number of output units and a large number of memory cells to store temporal contextual information, learning LSTM models become computationally expensive.

As an alternative to the standard architecture, we proposed the Long Short-Term Memory Projected (LSTMP) architecture to address the computational complexity of learning LSTM models [3]. This architecture, shown in Figure 1 has a separate linear projection layer after the LSTM layer. The recurrent connections now connect from this recurrent projection layer to the input of the LSTM layer. The network output units are connected to this recurrent layer. The number of parameters in this model is $n_c \times n_r \times 4 + n_i \times n_c \times 4 + n_r \times n_o + n_c \times n_r + n_c \times 3$,

Table 1: *Experiments with LSTM and LSTMP RNN architectures showing test set WERs and frame accuracies on development and training sets. L indicates the number of layers, for shallow (1L) and deep (2,4,5,7L) networks. C indicates the number of memory cells, P the number of recurrent projection units, and N the total number of parameters.*

C	P	Depth	N	Dev (%)	Train (%)	WER (%)
840	-	5L	37M	67.7	70.7	10.9
440	-	5L	13M	67.6	70.1	10.8
600	-	2L	13M	66.4	68.5	11.3
385	-	7L	13M	66.2	68.5	11.2
750	-	1L	13M	63.3	65.5	12.4
6000	800	1L	36M	67.3	74.9	11.8
2048	512	2L	22M	68.8	72.0	10.8
1024	512	3L	20M	69.3	72.5	10.7
1024	512	2L	15M	69.0	74.0	10.7
800	512	2L	13M	69.0	72.7	10.7
2048	512	1L	13M	67.3	71.8	11.3

where n_r is the number of units in the recurrent projection layer. In this case, the model size and the learning computational complexity are dominated by the $n_r \times (4 \times n_c + n_o)$ factor. Hence, this allows us to reduce the number of parameters by the ratio $\frac{n_r}{n_c}$. By setting $n_r < n_c$ we can increase the model memory (n_c) and still be able to control the number of parameters in the recurrent connections and output layer.

With the proposed LSTMP architecture, the equations for the activations of network units change slightly, the m_{t-1} activation vector is replaced with r_{t-1} and the following is added:

$$r_t = W_{rm} m_t \tag{7}$$

$$y_t = \phi(W_{yr} r_t + b_y) \tag{8}$$

where the r denote the recurrent unit activations.

2.4. Deep LSTMP

Similar to deep LSTM, we propose deep LSTMP where multiple LSTM layers each with a separate recurrent projection layer are stacked. LSTMP allows the memory of the model to be increased independently from the output layer and recurrent connections. However, we noticed that increasing the memory size makes the model more prone to overfitting by memorizing the input sequence data. We know that DNNs generalize better to unseen examples with increasing depth. The depth makes the models harder to overfit to the training data since the inputs to the network need to go through many non-linear functions. With this motivation, we have experimented with deep LSTMP architectures, where the aim is increasing the memory size and generalization power of the model.

3. Distributed Training: Scaling up to Large Models with Parallelization

We chose to implement the LSTM RNN architectures on multi-core CPU rather than on GPU. The decision was based on CPU's relatively simpler implementation complexity, ease of debugging and the ability to use clusters made from commodity hardware. For matrix operations, we used the Eigen matrix library [21]. This templated C++ library provides efficient implementations for matrix operations on CPU using vectorized instructions. We implemented activation functions and gradient calculations on matrices using SIMD instructions to benefit from parallelization.

We use the truncated backpropagation through time (BPTT) learning algorithm [22] to compute parameter gradients on short subsequences of the training utterances. Activations are forward propagated for a fixed step time T_{bptt} (e.g. 20). Cross entropy gradients are computed for this subsequence and back-propagated to its start. For computational efficiency each thread operates on subsequences of four utterances at a time, so matrix multiplies can operate in parallel on four frames at a time. We use asynchronous stochastic gradient descent (ASGD) [23] to optimize the network parameters, updating the parameters asynchronously from multiple threads on a multi-core machine. This effectively increases the batch size and reduces the correlation of the frames in a given batch. After a thread has updated the parameters, it continues with the next subsequence in each utterance, preserving the LSTM state, or starts new utterances with reset state when one finishes. Note that the last subsequence of each utterance can be shorter than T_{bptt} but is padded to the full length, though no gradient is generated for these padding frames.

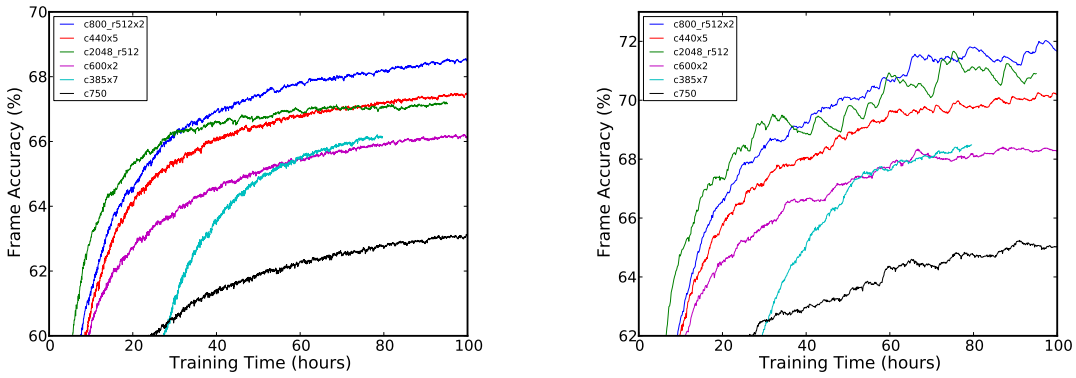
This highly parallel single machine ASGD framework described in [3] proved slow for training models of the scale we have used for large scale ASR with DNNs (many millions of parameters). To scale further, we replicate the single-machine workers on many (e.g. 500) separate machines, each with three, synchronized, computation threads. Each worker communicates with a shared, distributed parameter server [23] which stores the LSTM parameters. When a worker has computed the parameter gradient on a minibatch (of $3 \times 4 \times T_{bptt}$ frames), the gradient vector is partitioned and sent to the parameter server shards which each add the gradients to their parameters and respond with the new parameters. The parameter server shards aggregate parameter updates completely asynchronously. For instance, gradient updates from workers may arrive in different orders at different shards of the parameter server. Despite the asynchrony, we observe stable convergence, though the learning rate must be reduced, as would be expected because of the increase in the effective batch size from the greater parallelism.

4. Experiments

We evaluate and compare the performance of LSTM RNN architectures on a large vocabulary speech recognition task – the Google Voice Search task. We use a hybrid approach [24] for acoustic modeling with LSTM RNNs, wherein the neural networks estimate hidden Markov model (HMM) state posteriors. We scale the state posteriors by the state priors estimated as the relative state frequency from the training data to obtain the acoustic frame likelihoods. We deweight the silence state counts by a factor of 2.7 when estimating the state frequencies.

4.1. Systems & Evaluation

All the networks are trained on a 3 million utterance (about 1900 hours) dataset consisting of anonymized and hand-transcribed utterances. The dataset is represented with 25ms frames of 40-dimensional log-filterbank energy features computed every 10ms. The utterances are aligned with a 85 million parameter DNN with 14247 CD states. The weights in all the networks are initialized to the range (-0.02, 0.02) with a uniform distribution. We try to set the learning rate specific to a network architecture and its configuration to the largest value



(a) Held-out data. (b) Training data.

Figure 3: *Frame accuracy vs training time.*

that results in a stable convergence. As a result, the learning rates for stable convergence of different networks range from 5e-06 to 1e-05. The learning rates are exponentially decayed during training. With the correct learning rate, the training of LSTM RNNs results in a stable convergence. Apart from clipping the activations of memory cells to range [-50, 50], we do not limit the activations of other units, the weights or the estimated gradients.

During training, we evaluate frame accuracies (i.e. phone state labeling accuracy of acoustic frames) on a held-out development set of 200,000 frames. The trained models are evaluated in a speech recognition system on a test set of 22,500 hand-transcribed, anonymized utterances. For all the decoding experiments, we use a wide beam to avoid search errors. After a first pass of decoding using the LSTM RNNs with a 5-gram language model heavily pruned to 23 million n-grams, lattices are rescored using a 5-gram language model with 1 billion n-grams.

The input to the LSTM RNNs is log-filterbank energy features, with no frame stacking. Since the information from the future frames helps making better decisions for the current frame, we delay the output HMM state label by 5 frames. We do not calculate the errors for the first 5 frames of each utterance during backpropagation and we repeat the last frame of each utterance for 5 more time steps.

4.2. Results

In Table 1, we summarize the results for various LSTM and LSTMP RNN architectures. We observe that the conventional LSTM RNNs with a single layer do not perform very well for this large scale acoustic modeling task. With two layers of LSTM RNNs, the performance improves but still it is not very good. The LSTM RNN with five layers approaches the performance of the best model. We see that training an LSTM RNN with seven layers is hard, the model starts converging after a day of training. From the table, one can see that the LSTMP RNN models with a single layer and a large number of memory cells tends to overfit the training data. Increasing the number of LSTMP RNN layers seems to alleviate this problem of memorization and to result in better generalization to held-out data. The LSTMP RNN models give slightly better results than the LSTM RNN model with 5 layers. We see that increasing the number of parameters in the LSTMP RNN models more than 13M by having more layers or more memory cells does not give performance improvements.

Figure 3 compares the frame accuracies on training and held-out sets for various LSTM and LSTMP architectures. The overfitting problem with LSTMP RNN architecture with large number of memory cells (2048) can be seen clearly. We observe that LSTMP RNN architectures converges faster than LSTM RNN architectures. It is clear that having more layers helps generalization but makes training harder and convergence slower.

Table 2 shows how the performance of networks with deep LSTMP RNN architecture changes with the depth and number of model parameters. We see that increasing the number of parameters over 13M does not improve performance. We can also decrease the number of parameters substantially without hurting performance much. The deep LSTMP RNN architecture with two layers each with 800 cells and 512 recurrent projection units converges mostly in 48 hours and gives 10.9% WER on the independent test set. Training this model for 100 hours improves the WER to 10.7% and for 200 hours to 10.5%. In comparison, our best DNN models with 85M parameters gives 11.3% at the same beam and training takes a few weeks.

Table 2: *Experiments with LSTMP RNN architectures.*

C	P	Depth	N	WER (%)
1024	512	3L	20M	10.7
1024	512	2L	15M	10.7
800	512	2L	13M	10.7
700	400	2L	10M	10.8
600	350	2L	8M	10.9

5. Conclusions

We showed that deep LSTM RNN architectures achieve state-of-the-art performance for large scale acoustic modeling. The proposed deep LSTMP RNN architecture outperforms standard LSTM networks and DNNs and makes more effective use of the model parameters by addressing the computational efficiency needed for training large networks. We also show for the first time that LSTM RNN models can be quickly trained using ASGD distributed training.

6. References

[1] A. J. Robinson, G. D. Cook, D. P. W. Ellis, E. Fosler-Lussier, S. J. Renals, and D. A. G. Williams, "Connectionist speech recognition of broadcast news," *Speech Commun.*, vol. 37, no. 1-2, pp. 27–45, May 2002.

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

[3] H. Sak, A. Senior, and F. Beaufays, "Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition," *ArXiv e-prints*, Feb. 2014.

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[5] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *Neural Networks, IEEE Transactions on*, vol. 5, no. 2, pp. 157–166, 1994.

[6] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur, "Recurrent neural network based language model," in *Proceedings of INTERSPEECH*, vol. 2010, no. 9. International Speech Communication Association, 2010, pp. 1045–1048.

[7] M. Sundermeyer, R. Schlüter, and H. Ney, "Lstm neural networks for language modeling." in *INTERSPEECH*, 2012, pp. 194–197.

[8] F. A. Gers and J. Schmidhuber, "LSTM recurrent networks learn simple context free and context sensitive languages," *IEEE Transactions on Neural Networks*, vol. 12, no. 6, pp. 1333–1340, 2001.

[9] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, vol. 12, pp. 5–6, 2005.

[10] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 5, pp. 855–868, 2009.

[11] F. Eyben, M. Wollmer, B. Schuller, and A. Graves, "From speech to letters using a novel neural network architecture for grapheme based ASR," in *Automatic Speech Recognition & Understanding, 2009. ASRU 2009. IEEE Workshop on*. IEEE, 2009, pp. 376–380.

[12] M. Wollmer, F. Eyben, B. Schuller, and G. Rigoll, "A multi-stream asr framework for blstm modeling of conversational speech," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, May 2011, pp. 4860–4863.

[13] A. Rahman Mohamed, G. E. Dahl, and G. E. Hinton, "Acoustic modeling using deep belief networks," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 20, no. 1, pp. 14–22, 2012.

[14] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 20, no. 1, pp. 30–42, Jan. 2012. [Online]. Available: http://dx.doi.org/10.1109/TASL.2011.2134090

[15] N. Jaitly, P. Nguyen, A. Senior, and V. Vanhoucke, "Application of pretrained deep neural networks to large vocabulary speech recognition," in *Proceedings of INTERSPEECH*, 2012.

[16] G. Hinton, L. Deng, D. Yu, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, G. Dahl, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, November 2012.

[17] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of ICASSP*, 2013.

[18] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[19] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with LSTM recurrent networks," *Journal of Machine Learning Research*, vol. 3, pp. 115–143, Mar. 2003.

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

[21] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," 2010. [Online]. Available: http://eigen.tuxfamily.org

[22] R. J. Williams and J. Peng, "An efficient gradient-based algorithm for online training of recurrent network trajectories," *Neural Computation*, vol. 2, pp. 490–501, 1990.

[23] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. W. Senior, P. A. Tucker, K. Yang, and A. Y. Ng, "Large scale distributed deep networks." in *NIPS*, 2012, pp. 1232–1240.

[24] H. Bourlard and N. Morgan, *Connectionist speech recognition*. Kluwer Academic Publishers, 1994.