To Improve the Robustness of LSTM-RNN Acoustic Models Using Higher-order Feedback From Multiple Histories

Hengguan Huang and *Brian Mak*

Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
{hhuangaj,mak}@cse.ust.hk

Abstract

This paper investigates a novel multiple-history long short-term memory (MH-LSTM) RNN acoustic model to mitigate the robustness problem of noisy outputs in the form of mis-labeled data and/or mis-alignments. Conceptually, after an RNN is unfolded in time, the hidden units in each layer are re-arranged into ordered sub-layers with a master sub-layer on top and a set of auxiliary sub-layers below it. Only the master sub-layer generates outputs for the next layer whereas the auxiliary sub-layers run in parallel with the master sub-layer but with increasing time lags. Each sub-layer also receives higher-order feedback from a fixed number of sub-layers below it. As a result, each sub-layer maintains a different history of the input speech, and the ensemble of all the different histories lends itself to the model's robustness. The higher-order connections not only provide shorter feedback paths for error signals to propagate to the farther preceding hidden states to better model the long-term memory, but also more feedback paths to each model parameter and smooth its update during training. Phoneme recognition results on both real TIMIT data as well as synthetic TIMIT data with noisy labels or alignments show that the new model outperforms the conventional LSTM RNN model.

Index Terms: long short-term memory, recurrent neural network, multiple histories

1. Introduction and related works

Recurrent neural network (RNN), especially when it is coupled with deep learning, can be a very powerful model for sequential signals. It has been successfully deployed in many state-of-the-art systems for various applications such as automatic speech recognition (ASR) [1], language modeling [2], machine translation [3], computer vision [4], etc. RNN, when unfolded in time, may be considered as a very deep neural network (DNN). Thus, in principle, it can model long-span temporal dependency in sequential signals through simple recurrent network connections. However, in practice, training RNNs using the back-propagation through time (BPTT) algorithm [5] can be difficult due to the well-known vanishing or exploding gradient problem [6]. The problem may be sometimes solved by simple heuristic such as gradient clipping, or by carefully designed initialization and momentum scheme [7]. A principled solution is to replace the simpler stochastic gradient descent (SGD) algorithm by the second-order Hessian-free optimization algorithm [8]. Another approach is to bridge the gap between an RNN hidden state and its preceding states by the addition of direct feedback paths to the latter so that gradients can reach the preceding states more readily. The earliest work in this approach is the higher-order NARX RNN [9]; a similar higher-order RNN (HO-RNN) [10] is recently proposed for language modeling. On the other hand, in clockwork RNN (CW-RNN) [11], hidden states are partitioned into groups which run at different clock speeds and only groups running at lower speeds feedback to groups running at faster speed. CW-RNN was shown to outperform long short-term memory (LSTM) in some tasks. A CW-RNN variant called multi-timescale LSTM (MT-LSTM) [12], which allows both slow-to-fast and fast-to-slow state feedback, has shown better performance than other neural-network based models in 4 text classification tasks. In ASR, perhaps the most adopted solution is the LSTM RNN [13]. LSTM solves the problem by using a set of self-learning gates to control the amount of short-term and long-term information to forget, to retain, and to further propagate. In fact, the more powerful deep bi-directional LSTM RNN [14] is used in many state-of-the-art ASR systems.

It is also found that LSTM RNN is more robust to input noises — additive noises, channel noises and reverberation noises for ASR [15]. In this paper, we would like to investigate a different robustness issue: misalignment and mislabeling noises that frequently occur in training data. In training DNN-based acoustic models, the training state labels are usually produced by forced aligning the training utterances with an inferior GMM-HMM acoustic model, and their alignments are not optimal for the DNN. Moreover, during semi-supervised training of acoustic models for large-vocabulary ASR, most of the training data are not manually transcribed and their (phone or word) labels are again obtained from the recognition results of other models, and decoding errors are inevitable. Inspired by the multiple feedback paths suggested in NARX RNN and CW-RNN, we propose a multiple-history LSTM (MH-LSTM) in which different LSTM units maintain different histories of what they 'hear' and are connected with higher-order feedback. The ensemble of MH-LSTM units help smooth out the misalignment and mislabeling noises in the training targets.

2. Multiple-history LSTM RNN

In the following discussion, we assume that an RNN only has one hidden layer on which a softmax layer is stacked upon to model the posterior probabilities of the training targets. Its extension to deep MH-LSTM RNN is straight-forward. Moreover, when the RNN unit (simple neuron or LSTM cell) is immaterial for the discussion, we will use the term "multiple-history RNN" (MH-RNN) instead of MH-LSTM which, though, is the final model we used in the reported ASR experiments.

2.1. Overview of multiple-history RNN

Inspired by the use of multiple feedback paths in CW-RNN, we also group the hidden units into sub-layers. The top sub-layer will be called the *master* sub-layer, while the remaining ones are called *auxiliary* sub-layers. Only the master sub-layer is

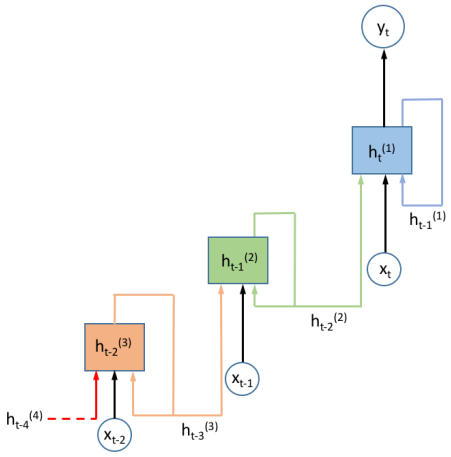


Figure 1: *A 2nd-order multiple-history RNN.*

directly connected to the output layer, and the sub-layers are arranged in order of increasing time lags that they run with. That is, the 1^{st} master sub-layer runs with no time delay; the 2^{nd} sub-layer runs with a time lag of one unit; the 3^{rd} sub-layer runs with a time lag of 2 units, and so on. All units in all sub-layers are initialized differently and randomly. As a result, each sub-layer maintains a different history of the training data, and an MH-RNN with H sub-layers will maintain H different histories. Fig. 1 depicts a 2nd-order MH-RNN: the rightmost unit is the master while the remaining ones are auxiliary. In the example, each unit also gets inputs from its last recurrent state plus the recurrent state from the following auxiliary unit (to its left).

2.2. Comparison with Simple RNN and HO-RNN

MH-RNN is similar to HO-RNN as they both use higher-order feedback. Formally, these RNNs are described by the following recurrence relations:

$$\text{SRNN:} \quad \mathbf{h}_t = \sigma\big(\mathbf{W}_x\mathbf{x}_t + \mathbf{W}_{h(1)}\mathbf{h}_{t-1} + \mathbf{b}_h\big)$$

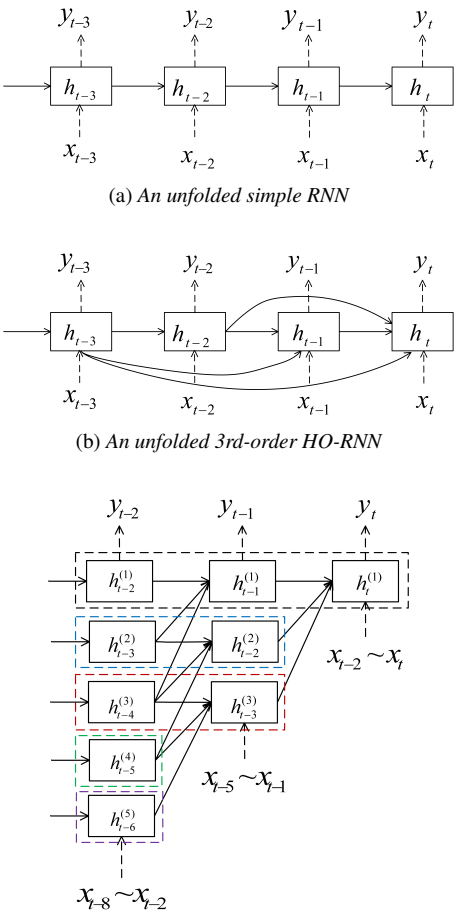
$$\text{HO-RNN:} \quad \mathbf{h}_t = \sigma\big(\mathbf{W}_x\mathbf{x}_t + \sum_{k=1}^{p}\mathbf{W}_{h(k)}\mathbf{h}_{t-k} + \mathbf{b}_h\big)$$

$$\text{MH-RNN:} \quad \mathbf{h}_t^{(m)} = \sigma\big(\mathbf{W}_x\mathbf{x}_t + \sum_{k=1}^{p}\mathbf{W}_{h(k)}\mathbf{h}_{t-k}^{(m+k-1)} + \mathbf{b}_h^{(m)}\big)$$

$$\mathbf{y}_t = \text{softmax}\big(\mathbf{W}_o\mathbf{h}_t + \mathbf{b}_o\big)$$

where SRNN represents a simple RNN; p is the model order of HO-RNN or MH-RNN; m indicates the sub-layer index which serves also as the label for the history in MH-RNN; $\mathbf{x}_t$, $\mathbf{h}_t$ and $\mathbf{y}_t$ are the input, hidden and output vectors at time t; $\mathbf{h}_t^{(m)}$ is the hidden vector of the mth sub-layer at time t; $\mathbf{W}_x$ and $\mathbf{W}_o$ are the weight matrices of the input and output layers; $\mathbf{W}_{h(k)}$ is the recurrent weight matrix of the preceding kth hidden state; $\mathbf{b}_h$ and $\mathbf{b}_o$ are the biases of the hidden and output layer; σ is the sigmoid function.

Fig. 2 illustrates the difference between a simple RNN, an HO-RNN, and an MH-RNN when they are unfolded in time. In Fig. 2(c), similar to a pth-order HO-RNN, each hidden unit in a sub-layer of a pth-order MH-RNN receives recurrent inputs from itself and from $(p-1)$ auxiliary sub-layers below it. However, unlike the HO-RNN, the preceding states of an MH-RNN come from hidden units from different auxiliary sub-layers, each having a different history. In fact, if all the auxiliary



(a) *An unfolded simple RNN*

(b) *An unfolded 3rd-order HO-RNN*

(c) *An unfolded 3rd-order MH-RNN*

Figure 2: *Various RNNs when they are unfolded in time. In MH-RNN in (c), dotted boxes represent sub-layers which are numbered from 1 at the top, and they run with increasing time lags with the top master sub-layer running at zero time lag.*

sub-layers actually come from the same history, an MH-RNN is degenerated to an HO-RNN. Similarly, if there is only 1 sub-layer, then an MH-RNN is reduced to a simple RNN. Thus, MH-RNN is a generalization of simple RNN. It is more powerful and robust than simple RNN or HO-RNN as it has a larger state space that helps extract different views of the temporal information in a training sequence. We believe that MH-RNN have the following advantages over the other RNNs:

(a) Similar to HO-RNN and CW-RNN, an error signal from the output layer δ_y^t propagates back in more feedback paths to each hidden state in the unfolded RNN. For example, δ_y^t will propagate back to the hidden states at $(t-3)$ as follows. To $\mathbf{h}_{t-3}^{(1)}$ via the path, $P1$:

$$\mathbf{h}_t^{(1)} \to \mathbf{h}_{t-1}^{(1)} \to \mathbf{h}_{t-2}^{(1)} \to \mathbf{h}_{t-3}^{(1)};$$

to $\mathbf{h}_{t-3}^{(2)}$ via the paths, $P2.1$ and $P2.2$:

$$\mathbf{h}_t^{(1)} \to \mathbf{h}_{t-1}^{(1)} \to \mathbf{h}_{t-3}^{(2)} \text{ or } \mathbf{h}_t^{(1)} \to \mathbf{h}_{t-2}^{(2)} \to \mathbf{h}_{t-3}^{(2)},$$

and to $\mathbf{h}_{t-3}^{(3)}$ via the direct path, $P3$:

$$\mathbf{h}_t^{(1)} \to \mathbf{h}_{t-3}^{(3)}.$$

(b) Compared with simple RNN, both HO-RNN and MH-RNN provide some shorter path for the error signal to reach preceding states more easily and thus, may model the longer-term temporal dependency better.

(c) For simple RNN, the error signal δ_y^t only reaches $\mathbf{h}_{t-3}$ via path $P1$. In HO-RNN, all the $\mathbf{h}_{t-3}^{(*)} \equiv \mathbf{h}_{t-3}$ are equivalent as all the sub-layers actually come from the same of hidden state, representing a single history. As a result, the error signal δ_y^t from $\mathbf{y}_t$ will reach $\mathbf{h}_{t-3}$ via multiple paths which will be weighted by $\mathbf{h}_{t-3}$ and provides different contributions (partial gradients) to the update of the various weight matrices $\mathbf{W}_{h(1)}, \mathbf{W}_{h(2)}, \ldots$ The multiple feedback paths help smooth out the model update, and we expect that HO-RNN can handle some noises in the training targets caused by mislabeling and misalignment. MH-RNN further expands the hidden state space to provide multiple versions of $\mathbf{h}_t$ so that the multiple feedback paths will be weighted differently. Analogous to an ensemble of classifiers, the ensemble of hidden state of the same time-stamp but from different histories in MH-LSTM is expected to give better and more stable model update that may better mitigate the problem caused by mislabeling and misalignment noises.

(d) Although both MH-RNN and HO-RNN or CW-RNN allows multiple feedback paths, the latter two models only maintain one single history while MH-RNN uses multiple histories to enhance its modeling power, especially in its capability of handling mislabeling and misalignment noises.

2.3. Update formula

Let $\delta_{h(m)}^t$ be the delta (error) signal propagated back to $\mathbf{h}_t^{(m)}$, the hidden state on the mth sub-layer at time t. The recurrent weight matrix of the kth preceding state of an MH-RNN with H histories (or sub-layers), where $1 \leq k \leq p$, is updated as follows:

$$\mathbf{W}_{h(k)}^{(new)} = \mathbf{W}_{h(k)} + \gamma \boldsymbol{\Delta}\mathbf{W}_{h(k)} \ , \tag{1}$$

where γ is the learning rate, and

$$\boldsymbol{\Delta}\mathbf{W}_{h(k)} = \sum_{t=2}^{T} \delta_{h(1)}^t \mathbf{h}_{t-k}^{'(k)} + \sum_{m=2}^{H-k+1} \sum_{t=3-m}^{T-H} \delta_{h(m)}^t \mathbf{h}_{t-k}^{'(m+k-1)} \ , \tag{2}$$

where T is the BPTT stepsize and any delta term with a -ve time index has to be copied from the last BPTT step; $'$ is the vector/matrix transpose operator.

2.4. Remarks

Although MH-RNN is described above, we actually used MH-LSTM RNN in the experimental evaluation of the model. It is trivial to adapt the update formulas, Eq. 1 and Eq. 2, for each of the gate matrices and cell state matrix of the LSTM. Moreover, in the actual MH-LSTM implementation, the number of histories H will affect its memory size and training speed.

3. TIMIT Experiments

We evaluated the proposed MH-LSTM RNN model on real and synthetic noisy speech data based on the TIMIT corpus.

3.1. The TIMIT speech corpus

The standard NIST training set which consists of 3,696 utterances from 462 speakers was used to train the various DNN and RNN models. A separate development data set, consisting of 400 utterances from 50 speakers, was used for early stopping, and the standard core test set, consisting of 192 utterances spoken by 24 speakers, was used for evaluation. We followed the standard TIMIT protocol and collapsed the original 61 phonetic labels in the corpus into the standard set of 39 phonemes for reporting the recognition performance in terms of phoneme error rate (PER). Phoneme recognition was performed using Viterbi decoding with a phone bigram language model estimated from the TIMIT training transcriptions using the Kaldi toolkit [16].

Table 1: *Summary of TIMIT phoneme recognition performance. F = input context in number of frames; L = number of hidden layers; N = number of nodes per hidden layer; p = model order.*

Model	F	L	N	PER %
DNN	11	4	1024	21.8
LSTM	1	3	512	21.0
LSTM	5	3	512	20.6
HO-LSTM ($p = 2$)	5	3	512	20.3
MH-LSTM ($p = 5$)	5	3	256	**19.8**

3.2. Feature extraction and model training procedure

Acoustic hidden Markov models (HMM) based on Gaussian-mixture model (GMM), deep neural network (DNN), LSTM-RNN, HO-LSTM RNN, and MH-LSTM RNN were built. GMM models employed fMLLR-adapted 39-dimensional MFCC features, while all neural-network based models used 40 mel-filterbank coefficients without their derivatives. Inputs to DNN/RNNs were normalized to have zero mean and unit variance.

The GMM-HMM was trained using the standard Kaldi TIMIT recipe and there were 1940 tied context-dependent states. It was then used to derive the state targets for subsequent DNN/RNN training through forced alignment. All DNN/RNN models were trained by our own codes developed using Theano. Inputs of DNN consisted of the current frame together with its 5 left and 5 right contextual frames. The number of hidden layers, the number of hidden nodes per layer, and the model order for HO-LSTM were varied from 2–4, 128–1024, and 2–3 respectively to find the best DNN/RNN architecture for the task. Both DNN and RNNs were trained by optimizing the target cross entropies, using BP and BPTT respectively and SGD.

3.3. Baseline results

Table 1 shows the TIMIT phoneme recognition performance of the baseline DNN, LSTM RNN, and HO-LSTM RNN. It can be seen that LSTM performs much better than DNN by 0.8% or 1.2% absolute when inputs of 1 or 5 contextual frames are used. Since more contexts give better LSTM performance, all ensuing LSTM experiments employed an input context of 5 frames. The addition of higher-order recurrences in the 2^{nd}-order HO-LSTM, which otherwise shares the same architecture as the baseline LSTM, help reduce the PER by 0.3% absolute.

3.4. Experiments: clean TIMIT data

Since MH-LSTM is similar to HO-LSTM, based on the best configuration of HO-LSTM in Table 1, we tried to find the best MH-LSTM configuration by varying the number of hidden layers between 2–3 and number of hidden nodes per layer from 128 to 512. The number of histories H and the model order p were

Table 2: *Performance of MH-LSTM with different model configurations. $H = 11$ and p = 5.*

#Layers, L	#Nodes/Layer, N	PER%
2	128	21.6
2	256	19.9
2	512	21.2
3	128	21.3
3	**256**	**19.8**
3	512	21.2

Table 3: *Effect of the number of histories and model order of MH-LSTM RNN.*

#Histories, H	Order, p	PER %
11	3	20.1
11	**5**	**19.8**
11	7	20.1
21	**5**	**19.8**

fixed to 11 and 5 in all experiments unless otherwise stated. Table 2 gives the MH-LSTM performance under different network configurations. Both in the development set and training set, an MH-LSTM with 3 hidden layers and 256 nodes per layer gave the best recognition results.

We further checked the importance of multiple histories H and the effect of the model order p by varying their values. The results are shown in Table 3. It is found that for this relatively simple task, a model order of 5 is optimal and 11 multiple histories is sufficient though increasing H would not hurt performance.

In summary, a 5th-order MH-LSTM with a 5-frame input context and 11 multiple histories improves the LSTM baseline PER of 20.6% by 0.8% absolute to 19.8%.

3.5. Experiments: synthetic TIMIT data with noisy targets

Based on the best configurations found in Section 3.4, we repeated the recognition experiments with target noises.

3.5.1. Robustness against mis-alignments

Mis-alignments in the targets readily occur since the state targets are usually generated by an inferior acoustic model, e.g., a GMM-HMM in our case. We further simulated mis-alignment noises in TIMIT training data by randomly perturbing a proportion of their state boundaries (produced by the baseline GMM-HMM) by $\pm$1–3 frames. The perturbed TIMIT data were used to re-train the LSTM, HO-LSTM and MH-LSTM as before, and their phoneme recognition performances are shown in Table 4. It is found that when the amount of perturbed state boundaries increases to 40%, the PER of the baseline LSTM also increases steadily by 1.1% absolute from 20.6% to 21.7%. The PER of HO-LSTM also increases by the same amount of 1.1%. On the other hand, MH-LSTM increases only by 0.7% absolute from 19.8% to 20.5%.

3.5.2. Robustness against mislabeling

Wrong target labels are also common when no manual transcriptions are available for the training data, and they have to generated, again, by some ASR systems. Here, we simulated mislabeling by randomly replacing a proportion of correct TIMIT phone labels in the training data by wrong labels. The baseline GMM-HMM was used to forced-align the TIMIT data using the wrong phonetic transcriptions, and the various LSTM models were re-trained using the wrong targets. Note that this mislabeling simulation produces much worse target noises then the last experiment since a single wrong label may result in many frames of wrong state targets for LSTM training.

Table 4: *Effect of state mis-alignments on TIMIT PER %.*

Model	State Mis-alignments		
	0%	20%	40%
LSTM	20.6	21.1	21.7
HO-LSTM ($p = 2$)	20.3	21.2	21.4
MH-LSTM ($p = 5$)	19.8	20.0	20.5

Table 5: *Effect of phoneme mislabeling on TIMIT PER %.*

Model	Mislabeled Phonemes			
	0%	5%	10%	20%
LSTM	20.6	21.6	22.8	24.9
HO-LSTM ($p = 2$)	20.3	21.6	22.5	23.7
MH-LSTM ($p = 5, H = 11$)	19.8	21.3	21.8	24.6
MH-LSTM ($p = 5, H = 21$)	19.8	20.7	21.3	23.4

The TIMIT phoneme recognition performances of the ensuing re-trained models are given in Table 5. When the amount of wrong labels is less than 20%, although all models perform worse, the performance of HO-LSTM is as bad as the baseline LSTM's whereas MH-LSTM's performace degrades more slowly. However, when the amount of wrong labels reaches 20%, the MH-LSTM suddenly performs very poorly, almost as bad as the baseline LSTM. Nonetheless, when we re-trained the MH-LSTM with more histories ($H = 21$), the new MH-LSTM performs much better than the old MH-LSTM with only 11 histories. The results prove that the multiple histories can mitigate the adverse effect of target noise, and greater target noise can be handled with more histories in the MH-LSTM RNN.

4. Conclusions and future work

We introduce a novel model called *multiple-history LSTM* (MH-LSTM), which is a generalization of LSTM and higher-order LSTM (HO-LSTM). Through a series of carefully designed experiments using both clean and synthetic noisy TIMIT data, we show that, compared with the conventional LSTM or HO-LSTM, MH-LSTM can better capture the long-term temporal dependencies between the input frames and output targets and is more resilient to mis-alignments and wrong labels in the training data. We also believe that the proposed MH-LSTM is more robust against input noises, and we will verify this experimentally in the future work. An issue with MH-LSTM is that it has a larger hidden state space and as a result, its training is slower. How to speed up its training will be another future work.

5. Acknowledgements

The work described in this paper was partially supported by grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project Nos. HKUST16206714 and HKUST16215816).

6. References

[1] A. Graves, A. r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 2013, pp. 6645–6649.

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

[3] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *International Conference on Learning Representations*, 2015. [Online]. Available: http://arxiv.org/abs/1409.0473

[4] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[5] 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.

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

[7] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the International Conference on Machine Learning*, 2013, pp. 1139–1147.

[8] J. Martens, "Deep learning via Hessian-free optimization," in *Proceedings of the International Conference on Machine Learning*, 2010.

[9] T. Lin, B. G. Horne, P. Tino, and C. L. Giles, "Learning long-term dependencies in NARX recurrent neural networks," *IEEE Transactions on Neural Networks*, vol. 7, no. 6, pp. 1329–1338, 1996.

[10] R. Soltani and H. Jiang, "Higher order recurrent neural networks," *CoRR*, vol. abs/1605.00064, 2016. [Online]. Available: http://arxiv.org/abs/1605.00064

[11] J. Koutn´ik, K. Greff, F. Gomez, and J. Schmidhuber, "A clockwork RNN," in *Proceedings of the International Conference on Machine Learning*, 2014, pp. 1863–1871.

[12] P. Liu, X. Qiu, X. Chen, S. Wu, and X. Huang, "Multi-timescale long short-term memory neural network for modelling sentences and documents," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing, Lisbon*, 2015, pp. 2326–2335.

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

[14] A. Graves, N. Jaitly, and A.-r. Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," in *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop*, 2013, pp. 273–278.

[15] J. T. Geiger, Z. Zhang, F. Weninger, B. Schuller, and G. Rigoll, "Robust speech recognition using long short-term memory recurrent neural networks for hybrid acoustic modelling," in *Proceedings of Interspeech*, 2014, pp. 631–635.

[16] D. Povey *et al.*, "The Kaldi speech recognition toolkit," in *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop*, 2011.

[17] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.