Benchmarking of LSTM Networks

Thomas M. Breuel
Google, Inc.
tmb@google.com

Abstract

LSTM (Long Short-Term Memory) recurrent neural networks have been highly successful in a number of application areas. This technical report describes the use of the MNIST and UW3 databases for benchmarking LSTM networks and explores the effect of different architectural and hyperparameter choices on performance. Significant findings include: (1) LSTM performance depends smoothly on learning rates, (2) batching and momentum has no significant effect on performance, (3) softmax training outperforms least square training, (4) peephole units are not useful, (5) the standard non-linearities (tanh and sigmoid) perform best, (6) bidirectional training combined with CTC performs better than other methods.

1 Introduction

LSTM networks [1, 2, 3] have become very popular for many sequence classification tasks. This note presents the results of large scale benchmarking with a wide range of parameters to determine the effects of learning rates, batch sizes, momentum, different non-linearities, and peepholes. The two datasets used for benchmarking are MNIST and the UW3 text line OCR task. The questions we are addressing are:

- Generally, how do LSTMs behave for different hyperparameters?
- How reproducible are training results based on hyperparameters?
- What are the effects of batching and momentum on error rates?
- How do different choices of non-linearities affect performance?
- Are peepholes useful?
- What are the effects of bidirectional methods?
- What are the effects of using CTC?

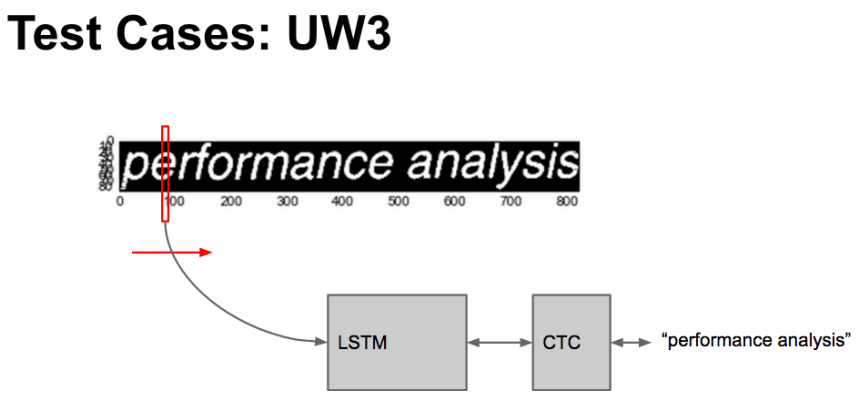

Figure 1: Application of LSTMs to OCR.

2 Input Data

We use two kinds of input data in our experiments: MNIST [6] and UW3 [5]. MNIST is a widely used benchmark on isolated digit handwriting classification. UW3 is an OCR evaluation database.

We transform both the MNIST and the UW3 inputs into a sequence classification problem by taking the binary image input and scanning it left to right using vertical slices to the image. MNIST images are 28x28 pixels large, so this yields a sequence of 28 bit vectors of dimension 28. UW3 images are variable size, but they are size-normalized and deskewed to a height of 48 pixels; they still have variable width.

3 MNIST Performance by Learning Rates and Network Size

In the initial experiments, LSTM models were trained on MNIST data with between 50 and 500 states and learning rates between 10^{-6} and 10^{-1}. Figure 2 shows the performance generally across different combination of number of states and learning rates. The figure shows that test set error rate depends quite smoothly on hyperparameters. As in other neural network learning models, training diverges above some upper limit for the learning rate. Error rates also increase for large numbers of hidden units and low learning rates. This is mostly due to learning being very slow, not overtraining. A look at the top 10 test set error rates (Figure 3) shows that it is fairly easy to achieve error rates between 0.8% and 0.9%.

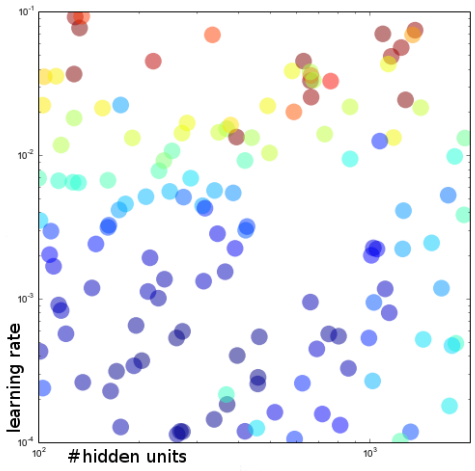

Figure 2: The parameter space explored during MNIST training of LSTMs. Error rates are indicated by color, representing a range of approximately 0.8% to about 2% error. Note that performance is quite consistent across the parameter space: similar learning rates and numbers of hidden units give similar error rates.

	err	exp	hidden	lrate	n	t
5011	0.0082	logs.46.clstm-mnist-2.tmb.19913335654.1.h5	205	0.000369	1560000	33069.80
6803	0.0083	logs.63.clstm-mnist-2.tmb.19913335702.1.h5	398	0.000401	5040000	352731.00
3813	0.0083	logs.39.clstm-mnist-2.tmb.19913335647.1.h5	371	0.000183	3720000	218506.00
1465	0.0084	logs.24.clstm-mnist-2.tmb.19978303889.1.h5	261	0.000531	2400000	76628.60
705	0.0086	logs.2.clstm-mnist-2.tmb.19913335610.1.h5	460	0.000283	1620000	141344.00
4923	0.0087	logs.45.clstm-mnist-2.tmb.19913335653.1.h5	272	0.000591	1860000	49120.80
3587	0.0087	logs.38.clstm-mnist-2.tmb.19967383652.1.h5	172	0.000311	3180000	48967.90
5610	0.0087	logs.49.clstm-mnist-2.tmb.19913335660.1.h5	272	0.000118	3900000	134446.00
7153	0.0090	logs.8.clstm-mnist-2.tmb.19971360572.1.h5	464	0.000542	900000	81540.20
6159	0.0090	logs.56.clstm-mnist-2.tmb.19993315493.1.h5	339	0.000144	1380000	76838.20
5510	0.0091	logs.48.clstm-mnist-2.tmb.19913335657.1.h5	459	0.000254	1680000	156782.00
1821	0.0092	logs.27.clstm-mnist-2.tmb.19982794897.1.h5	752	0.000567	1080000	236841.00
4762	0.0092	logs.44.clstm-mnist-2.tmb.19913335652.1.h5	268	0.000119	5520000	180688.00
3384	0.0092	logs.37.clstm-mnist-2.tmb.19920851987.1.h5	197	0.000650	1440000	30313.30

Figure 3: These are the best performing networks among the 660 LSTMs trained in the experiments described in Section 3

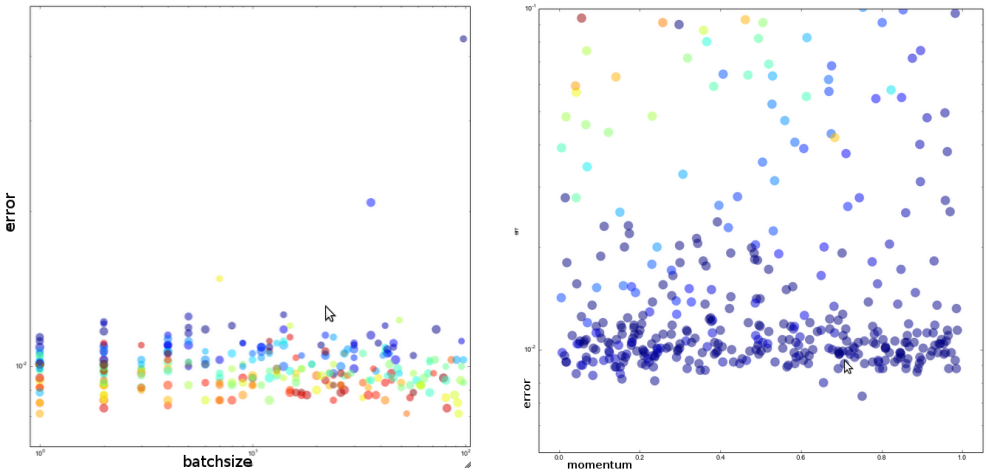

Figure 4: LSTM performance on MNIST is approximately independent of batchsize and momentum.

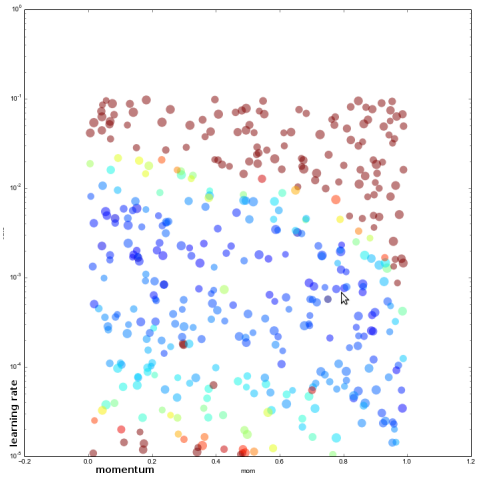

Figure 5: The optimal learning rate scales as $\frac{1}{1-\mu}$ witih momentum μ.

4 Effects of Batchsize and Momentum

Stochastic gradient descent often is performed with minibatches, computing gradients from a small set of training samples rather than individual samples. Such methods are supposed to "smooth out" the gradient. They also allow greater parallelization of the SGD process. Closely related to minibatches is the use of *momentum*, which effectively also averages gradients over multiple training samples.

For regular neural networks trained with stochastic gradient descent, batch sizes interact in complex ways with learning rates and nonlinearities. In particular, for sigmoidal nonlinearities, beyond a certain batch size, the learning rate needs to be scaled by the inverse of the batch size in order to avoid divergence; as a secondary effect, large batch sizes generally fail to achieve the same minimum error rates that single sample updates achieve. For ReLU (rectified linear, $f(x) = \max(0, x)$) nonlinearities, we don't observe the same kind of batchsize dependencies, however.

To test whether batch size dependencies exist for LSTM networks, 427 networks were trained with batch sizes ranging from 20 to 2000, momentum parameters between 0 and 0.99, and learning rates between 10^{-5} and 10^{-1}. The results are shown in Figures 4 and 5. These results show that there is no significant effect of either batch size or momentum parameter on error rates.

In addition, for a momentum parameter μ, the optimal learning rate is seen to scale as $\frac{1}{1-\mu}$; the reason is that with momentum, the same sample contributes to the update of the gradient effectively that many times.

We note that in these experiments, we obtained the best performance of LSTM networks on MNIST, with a test set error rate of 0.73%.

5 Different LSTM Types applied to MNIST

LSTM networks involve a number of choices of non-linearities and architecture. These are illustrated in Figure 6. For regular deep neural networks, we had observed that logistic and softmax output layers give significantly different results and wanted to see whether that carries over to LSTM. We had observed that peephole connections may not help recognition in preliminary experiments and wanted to verify this. ReLU nonlinearities appear to give significantly better results on other neural networks, so we investigated whether they also work in the context of LSTMs.

To test these ideas experimentally, 2101 LSTM networks in 12 different configurations were trained with learning rates between 10^{-6} and 10^{-1}. The results from these experiments are shown in Figure 7. From these results, we can draw the following conclusions:

- Peepholes do not seem to have a significant effect on error rate.
- Logistic vs softmax outputs makes no significant difference.
- Variants with linear outputs or ReLU units perform much worse.

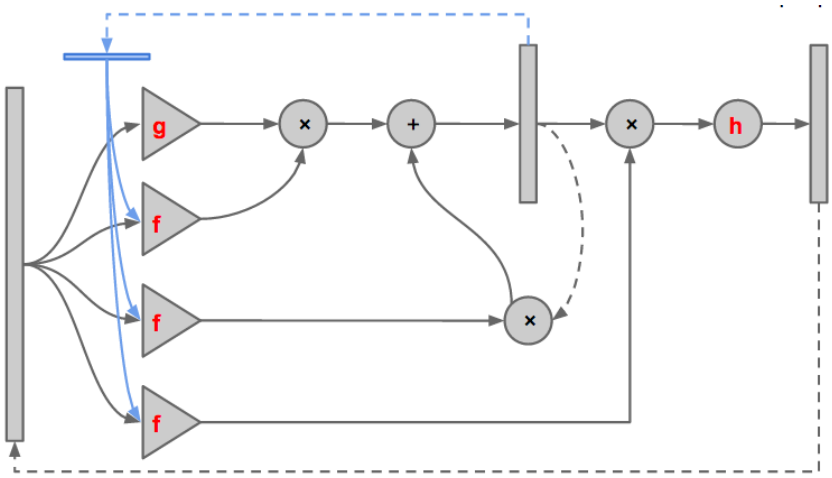

Figure 6: The structure of the LSTM model. The model involves three different choices of nonlinearities (f, g, and h), plus peephole connections (blue). Different LSTM structures used in the experiments: LINLSTM, LSTM, NPLSTM, RELU2LSTM, RELULSTM, RELUTANHLSTM.

Overall, the standard LSTM architecture without peephole connections seems to be a good choice based on these results.

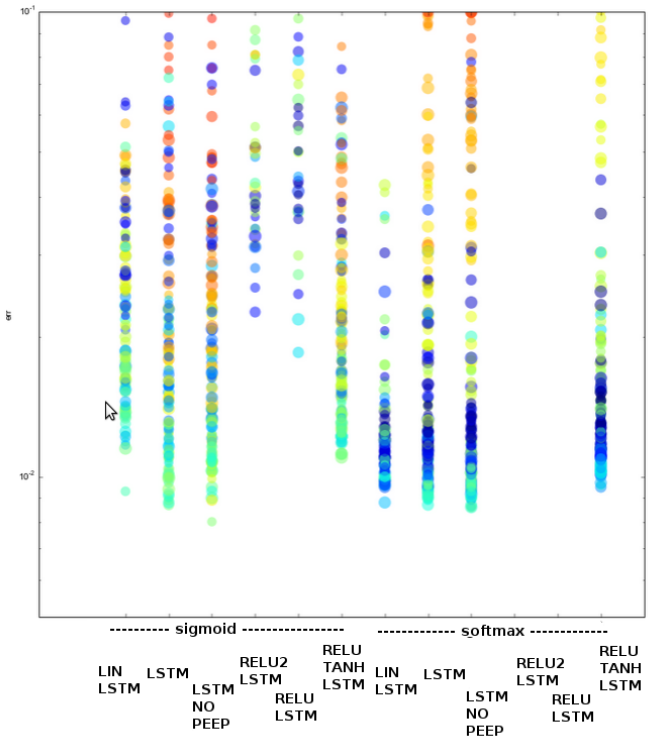

Figure 7: The distribution of error rates by LSTM type. The vertical axis represents error, the horizontal axis LSTM type. Each dot is one LSTM results, the best test error achieved during a training run. Dot size indicates number of hidden units and dot color indicates learning rate. The left six models use sigmoidal outputs and mean-squared error at the output, while the right six models use a softmax layer for output. Within each group, the results are, from left to right, for LINLSTM (h = linear), LSTM, NPLSTM, RELU2LSTM (g, h = ReLU), RELULSTM (g = ReLU, h = linear), RELUTANHLSTM (g = ReLU, h = tanh). The best performing variant is NPLSTM, with either logistic or softmax outputs.

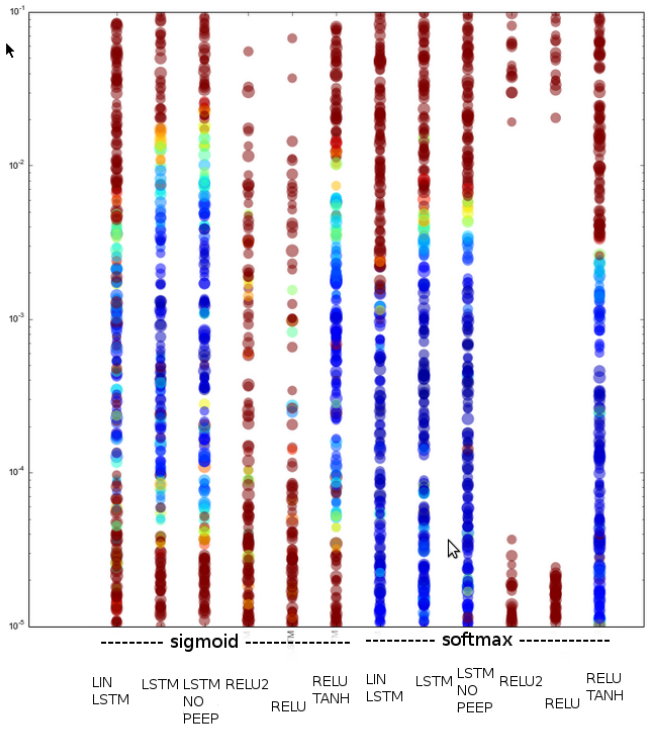

Figure 8: Error rates (color) by type and learning rate. The figure verifies that the range of error rates tried covers the upper range of convergent learning rates. (RELU2LSTM and RELULSTM both have a gap in learning rates where the networks all diverged within the first epoch.)

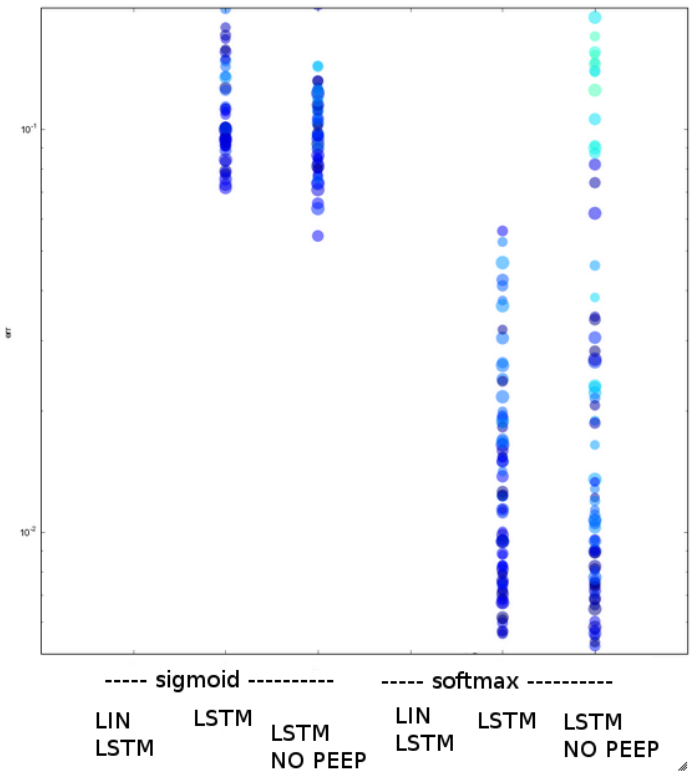

Figure 9: Error rates on the UW3 task for different network types. The left three networks are logistic output variants, the right three are softmax output variants. The LSTM types within each group are LINLSTM, LSTM, and NPLSTM. Unlike MNIST, in these experiments, logistic outputs perform much worse than softmax outputs. Networks without peephole connections perform slightly better than networks containing such connections.

6 Different LSTM Types applied to OCR

The experiments with different LSTM types were also carried out on the UW3 input data to verify the results on a more complex task. The biggest difference betweeh MNIST and UW3 is that the input and output sequences are variable size in UW3, there there are an order of magnitude more class labels, and that classes are highly unbalanced. The ReLU variants were not tested on this dataset.

The most striking difference between UW3 and MNIST results is that on UW3, logistic outputs perform much worse than softmax outputs. We verified that in all cases, the range of learning rates covered the region of convergence. If we look at the training curves for logistic vs softmax output, we see that with logistic output units, training starts off slower and "gets stuck" on a number of

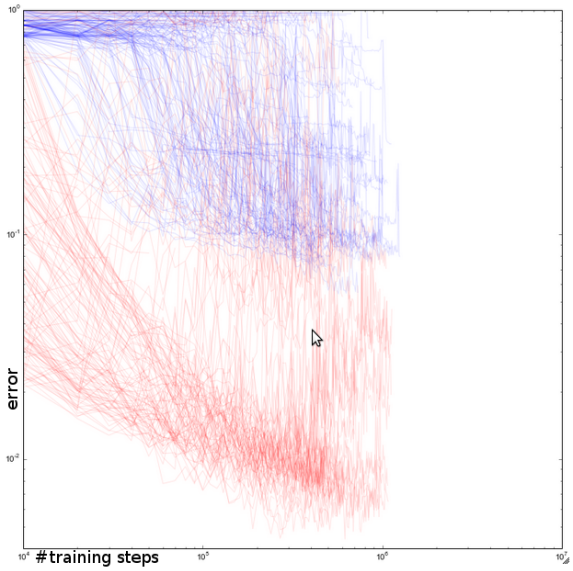

Figure 10: Training curves for UW3 LSTM training using logistic output units (blue) and softmax outputs (red).

distinct plateaus. The cause of this still remains to be investigated.

7 Bidirectional vs. Unidirectional, CTC vs. non-CTC

LSTM networks can be combined into various more complex network architectures. The two most common architectures are shown above. The first uses a single LSTM layer followed by a logistic or softmax output layer. For bidirectional LSTM training, the input sequence is processed both in forward and reverse, and the combined outputs of the forward and reverse processing at each time step are then combined into a final output. Bidirectional LSTM networks can take into account both left and right context in making decisions at any point in the sequence, but they have the disadvantage that they are not causal and cannot be applied in real time. Note that bidirectional networks have twice the number of internal states and slightly more than twice the number of weights than corresponding unidirectional networks with the same number of states.

LSTM networks learn sequence-to-sequence transformations, where input and output sequences are of equal lengths. In tasks like OCR, input signals are transformed into shorter sequences of symbols. Usually, a transcript ABC is represented as the symbolic output $\epsilon^+A^+\epsilon^+B^+\epsilon^+C^+\epsilon^+$, augmenting the original set of classes by an ϵ symbol. The LSTM network then predicts a vector of posterior probabilities in this augmented set of classes. The non-ϵ outputs

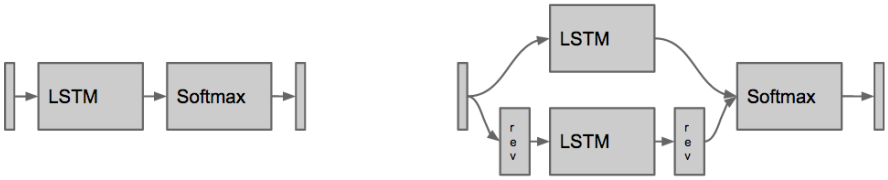

Figure 11: LSTM networks can be combined into various more complex network architectures. The two most common architectures are shown above: unidirectional LSTM (left) and bidirectional LSTM (right).

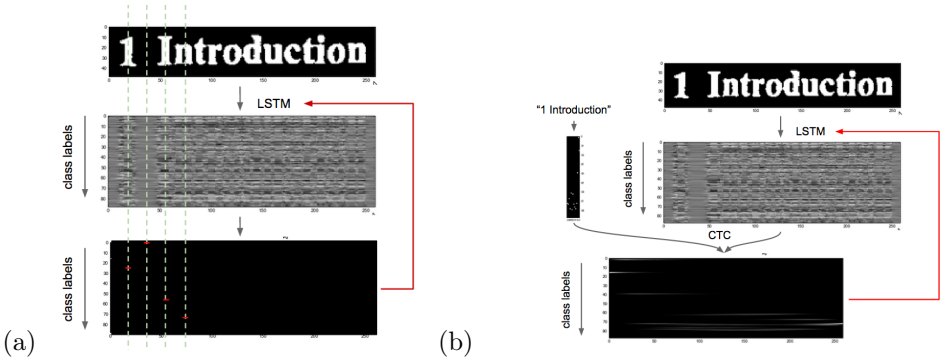

Figure 12: In non-CTC training, the target for LSTM training is constructed by assuming a fixed relationship between the images of input symbols and the apperance of the symbol in the output sequence. In CTC training, the target for LSTM training is constructed by aligning (using the forward-backward algorithm) the target sequence with the actual output of the LSTM.

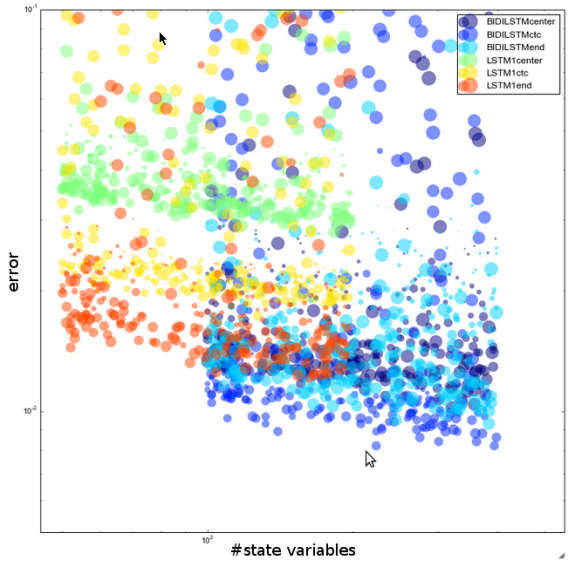

Figure 13: Performance of different network types and training modalities by number of states.

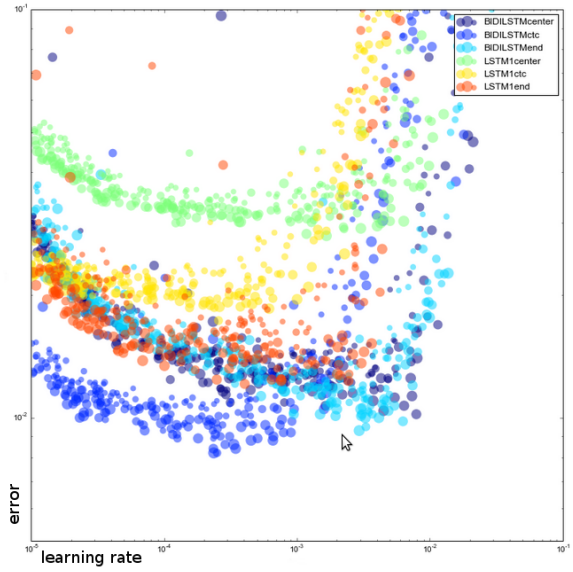

Figure 14: A plot of error rate by learning rate on MNIST for different network architectures and training modalities.

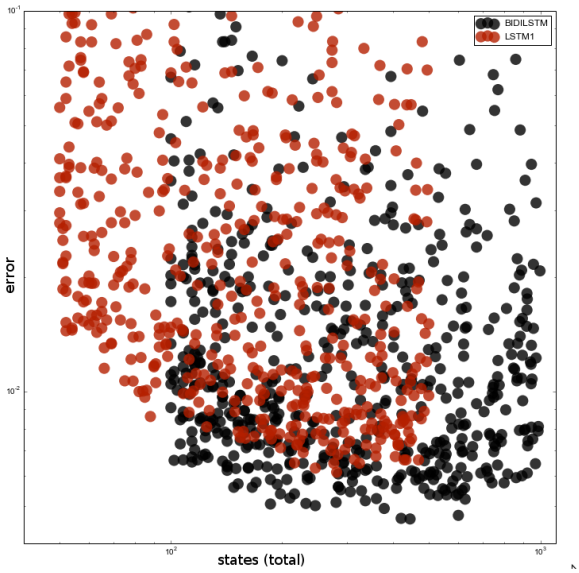

Figure 15: Unidirectional vs bidirectional LSTM training on the UW3 database.

correspond to the "location" of the corresponding symbol, but there are many possible choices: the location might be consistently at the beginning, center, or end of the symbol, or it might differ from symbol to symbol. In non-CTC training, the location of the output symbol is fixed based. In CTC training, the location of the output symbol is determined via the forward backward algorithm.

In the experiments, we compare unidirectional ("LSTM1") and bidirectional ("BIDI") networks. Since MNIST contains only a single output symbol per input, for non-CTC versions, we simply select a constant column (time step) in the output sequence where the network needs to output the symbol; the two time steps tested are in the middle of the output sequence ("center") and at the end of the output sequence ("end"). In addition, CTC was used witih both kinds of networks ("ctc").

The results of MNIST benchmarks on these six conditions are shown in Figure 13. Among BIDI networks, CTC training performed best. Among LSTM1 networks, placing the label at the end performed best. Not surprisingly, LSTM1center performed worse, because the unidirectional network could only take into account half the information from the input image before outputting a label. Surprisingly, LSTM1ctc performed worse than LSTM1center; that is, CTC training performed worse than placing the label explicitly at the end of the sequence for unidirectional training.

The error rate by learning rates and network type is shown in Figure 14. Surprisingly, the different networks achieve their lowest error rates at learning rates that are different by more than an order of magnitude. This can be

partially explained by the observation that CTC has a different output class distribution from non-CTC training (in particular, CTC has more non-ϵ class labels in the output distribution). But even among the non-CTC training runs, the optimal learning rates differ depending on network type and location of the target class in time.

Overall, these results confirm the hypothesis that BIDIctc training yields the best results. However, the results also caution us against simple benchmarks, since learning rates, network structure, and use of CTC interact in complex, non-monotonic ways.

For UW3, only bidirectional vs. unidirectional training was compared (since the consistent assignment of a location to characters in a text line is difficult to achieve). The results, shown in Figure 15, are consistent with the MNIST results: bidirectional LSTM significantly outperforms unidirectional LSTM networks at all network sizes. Note that in light of the MNIST result that CTC performs worse than non-CTC training with unidirectional training, it is possible that unidirectional training could be improved with a careful manual choice of target locations.

8 Eventual Divergence

In all experiments where training was continued long enough, we observed eventual slow divergence of the test set error. This is shown in Figure 16. Furthermore, the lowest learning rates resulted in the longest time to divergence This divergence is qualitatively different from the fast divergence we observe when the learning rates are too high. Furthermore, it is also represented in the training set error, so it does not represent overtraining.

Our interpretation of this phenomenon is that LSTM networks internally perform two separate, competing learning processes. If we think of an LSTM network as roughly analogous to a Hidden Markov Model (HMM), these two processes correspond roughly to structural learning and parameter learning. We postulate that structural learning is a slow process that explores different structures, with fast parameter learning overlaid on top of this process. Eventually, (after about one million training steps) in this example, the network has an optimal structure for the task, and further optimization results in changes to the structure that are deleterious to overall performance.

9 Discussion and Conclusions

Let us summarize the results:

- LSTM networks give excellent performance on MNIST digit recognition; this also represents a simple and useful test case for checking whether an LSTM implementation is performing correctly.

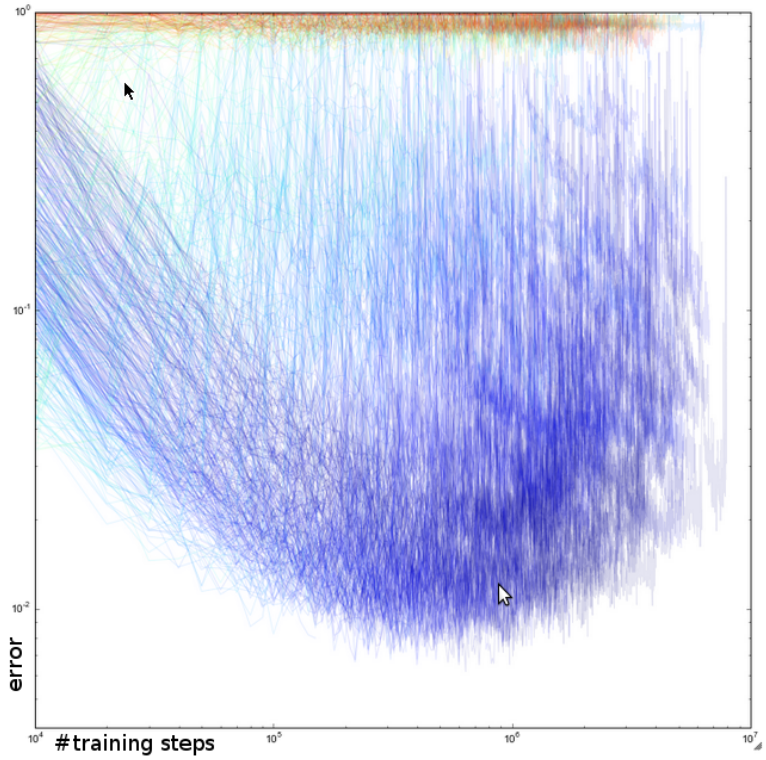

Figure 16: A plot of test-set error vs training steps for unidirectional LSTM training on the UW3 database. Note the logarithmic axes. Each curve represents one test set error training curve; the plot represents a total of 942 training runs with a range of learning rates and number of states. Color indicates learning rate, from blue (low) to red (high).

- LSTM network performance is fairly reproducible between training runs and test set error has broad flat minima in hyperparameter space (i.e., hyperparameter optimization is fairly simple).
- The best performing networks in all experiments were "standard" LSTM networks with no peephole connections.
- Peephole connections never resulted in any improved performance.
- Momentum and batchsize parameters had no observable effect on LSTM performance; this means that batching may often be a good method for parallelizing LSTM training.
- LSTM networks failed to converge to low error rate solutions with logistic outputs and MSE training for the OCR task; softmax training resulted in the lowest error rates overall.
- CTC and bidirectional networks generally perform better than fixed outputs and/or unidirectional networks.
- LSTM test set error rates seem to invariably diverge eventually.

These results agree with other, recently published results on LSTM peformance; in particular, [4] also found that standard LSTM architectures with the usual nonlinearities perform best, and that peephole connections do not improve performance. The other results reported above have not been previously obtained.

Numerous other issues remain to be explored experimentally. For example, we do not know what effect different choices of weight initialization have. Also, a number of other LSTM-like architectures have been proposed.

We believe that for exploring and benchmarking such architectural variants, the use of the MNIST and size-normalized UW3 data sets as used in this technical report form a good basis for comparison, since they are sufficiently difficult datasets to be interesting, yet still fairly easy to train on.

Appendix

The source code used in the experiments is available from `http://github.com/tmbdev/clstm`. The MNIST and UW3-derived datasets are available from `http://tmbdev.net` (in HDF5 format).

References

[1] Felix Gers, Jürgen Schmidhuber, et al. Lstm recurrent networks learn simple context-free and context-sensitive languages. *Neural Networks, IEEE Transactions on*, 12(6):1333–1340, 2001.

[2] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. Learning precise timing with lstm recurrent networks. *The Journal of Machine Learning Research*, 3:115–143, 2003.

[3] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Bidirectional lstm networks for improved phoneme classification and recognition. In *Artificial Neural Networks: Formal Models and Their Applications–ICANN 2005*, pages 799–804. Springer, 2005.

[4] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *arXiv preprint arXiv:1503.04069*, 2015.

[5] Robert M. Haralick. Uw-iii english/technical document image database, 1995.

[6] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998.