Parallelizing Linear Recurrent Neural Nets Over Sequence Length

Eric Martin and Chris Cundy*

*UC Berkeley

Abstract

RNN training and inference generally takes time linear in the sequence length because of non-linear sequential dependencies. We show the training and inference of RNNs with only linear sequential dependencies can be parallelized over the sequence length using the parallel scan algorithm, leading to rapid training on long sequences even with small minibatch size. We use this insight and a parallel linear recurrence CUDA kernel to accelerate several state of the art RNN architectures by up to 9x and to solve a synthetic sequence classification task with a one million time step dependency.

Introduction

Large minibatches are necessary for computational performance but create large memory requirements and damage model generalization ability.

Linear RNNs and convolutional models such as strongly typed RNNs, Wavenet, Bytenet, Quasi-RNNs, and simple recurrent units have achieved state of the art results on many sequential tasks with rapid training times.

Given x_t, λ_t can compute $h_t = \lambda_t h_{t-1} + x_t$ for $t = 1 \dots T$ on p processors in $O(T/p + \log(p))$ with the classic **parallel scan algorithm**. Backpropagation of gradient can also be parallelized with the same algorithm. We implemented a parallel linear recurrence operation in CUDA and integrated it with TensorFlow.

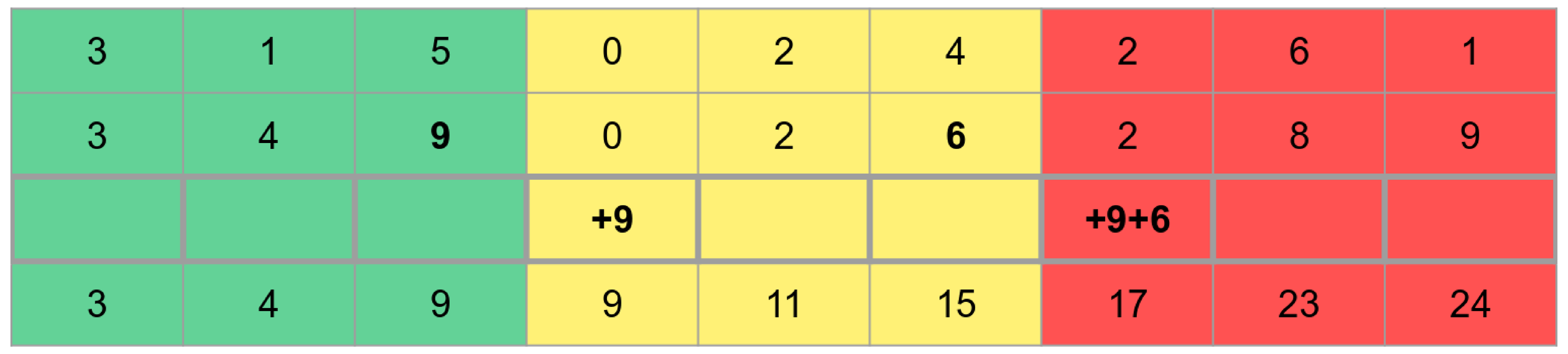

Figure 1: Example of parallelizing cumulative sum over 3 processors

Gated Impulse Linear Recurrence

Given a fast algorithm for evaluating linear recurrences, we introduce a new linear recurrent layer called **gated impulse linear recurrence (GILR)**

$$
\begin{aligned}
g_t &= \sigma(U x_t + b_g) \\
i_t &= \tau(V x_t + b_z) \\
h_t &= g_t \odot h_{t-1} + (1 - g_t) \odot i_t
\end{aligned}
$$

Linear Surrogate RNNs

RNNs have a transition function $s_t = f(s_{t-1}, x_t)$. s_t serves dual roles as a summary of the past as well as the output of the unit. Non-linear f in units such as vanilla RNN and LSTM prevents parallelization over sequence length.

Replacing the summary of the past s_{t-1} with a linear surrogate $\tilde{s}_{t-1}$ allows the easy adaption of any existing RNN architecture for parallel computation. Several recent linear RNNs can be viewed as linear surrogate RNNs.

The state of an LSTM consists of (c_t, h_t). c_t is already computed by linear recurrence, so a linear surrogate LSTM must only compute a linear $\tilde{h}_t$. A **GILR-LSTM** uses $\tilde{h} = \mathrm{GILR}(x)$

Training Runtime Results

Seq. Len.	SRU	QRNN(2)	QRNN(10)	GILR-LSTM
16	0.28	0.38	0.78	0.61
256	0.84	0.86	0.99	0.91
4,096	1.38	1.18	1.05	0.98
65,536	9.21	6.68	2.05	1.41

Table 1: Parallel kernel speedup compared to serial linear recurrence for a variety of LS-RNNs All models use two stacked RNN layers with 256 hidden units, keeping the GPU memory usage constant by fixing $bT = 65,536$ for minibatch size b and sequence length T. QRNN(k) refers to a QRNN with filter size k.

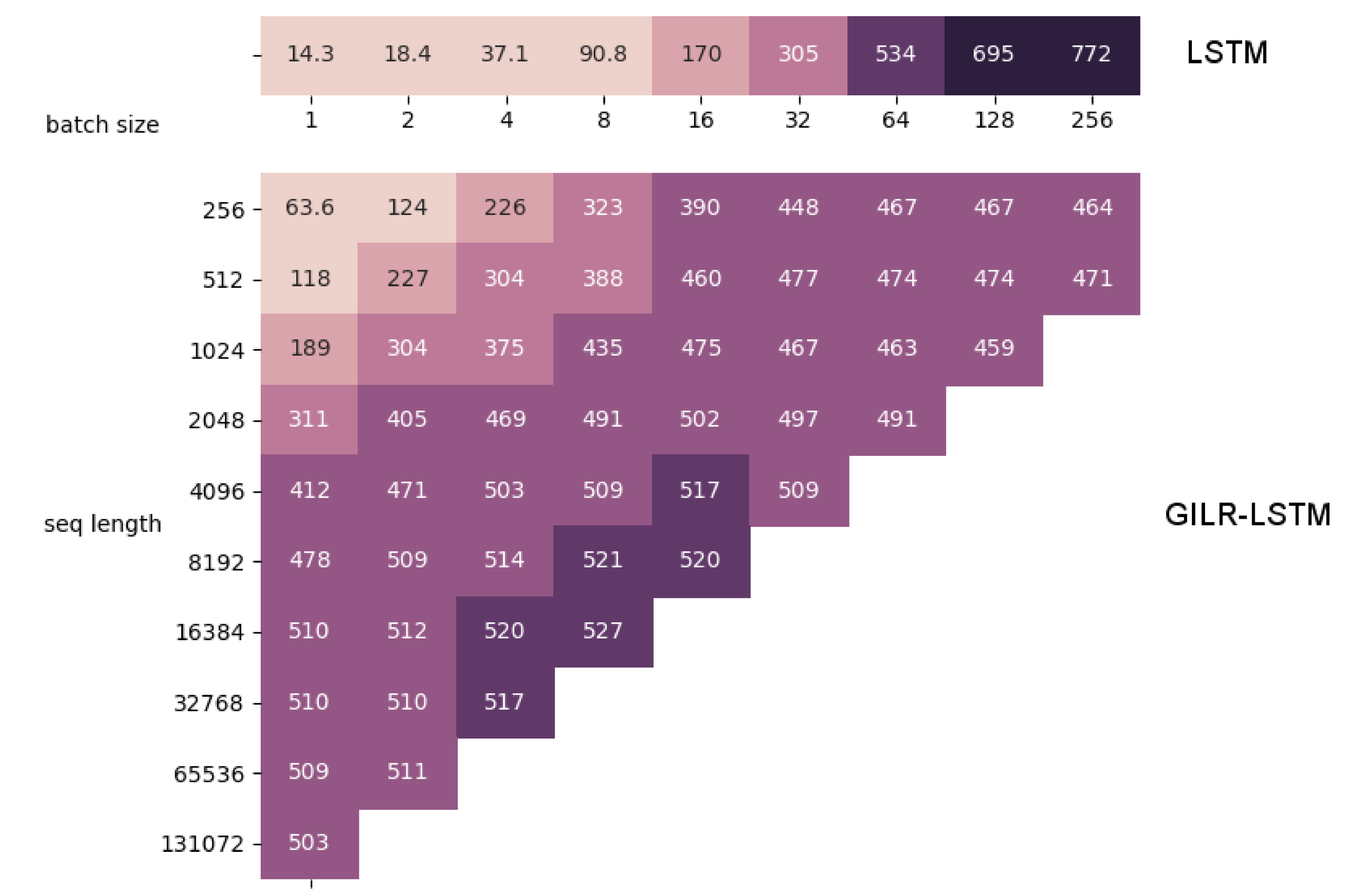

Figure 2: Throughput (thousand steps/s) for 2 layer 256 unit cuDNN LSTM and GILR-LSTM as a function of batch size and sequence length. LSTM throughput is independent of sequence length. GILR-LSTM can achieve much greater throughput at small batch sizes.

Learning Long-Term Dependencies

Task: Learn to remember 1 bit of information for T time steps.

We measured time until convergence for a 2 layer GILR-LSTM and LSTM for T ranging from 1,000 to 1,000,000.

The GILR-LSTM converged in over **6x** less wall time. We demonstrated a GILR-LSTM could learn a **one million time step** sequential dependency, which is at least a 100x longer dependency than previously learned.

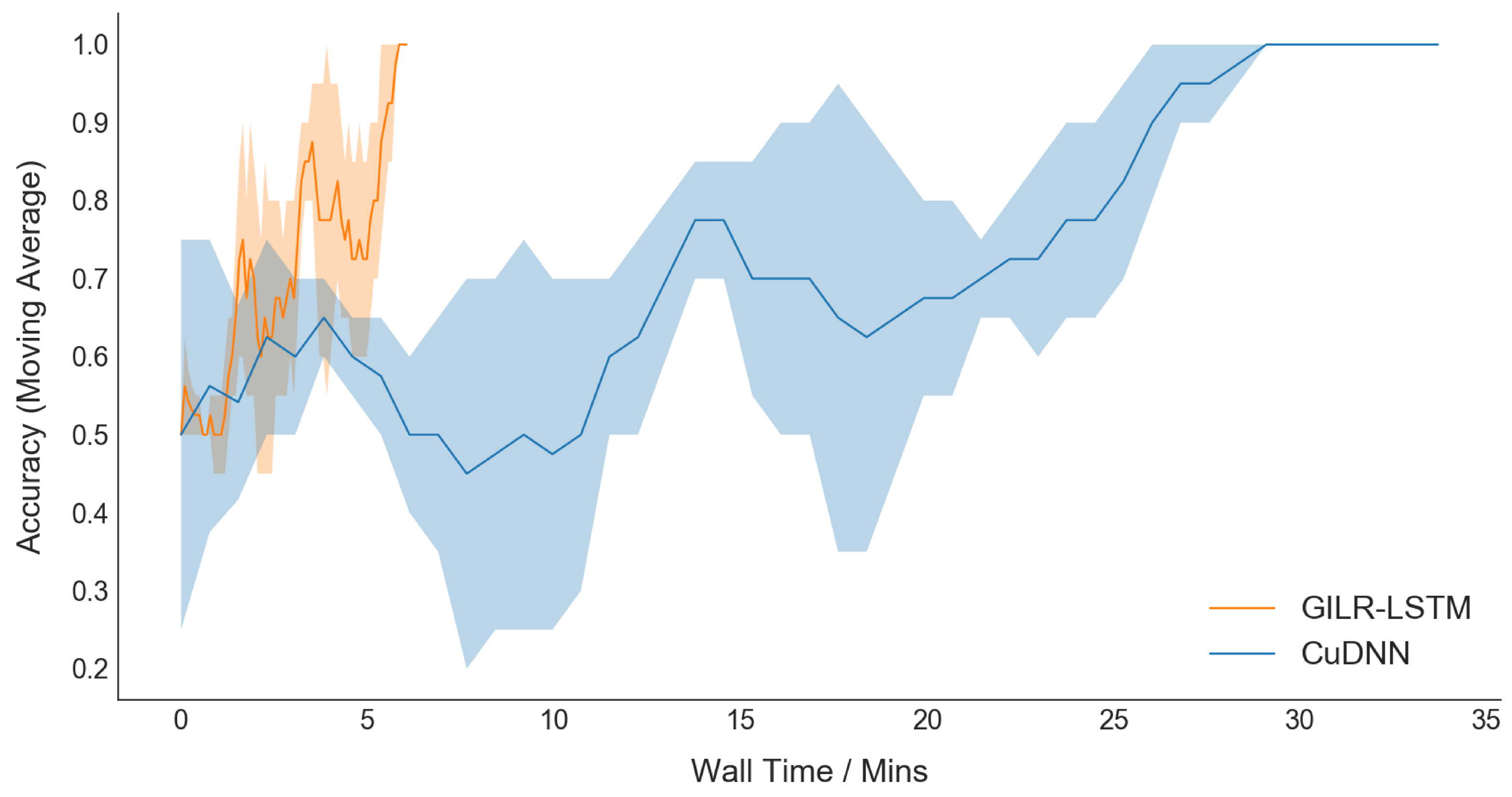

Figure 3: Accuracy on the memorization task with 8,192 sequence length

Conclusion

Parallel linear recurrence enables rapid learning on extremely long sequences at small minibatch sizes. A significant portion of deep learning's current success can be attributed to highly efficient matrix multiplication and convolution kernels. We hope that parallel linear recurrence can join these algorithms and be to large scale sequence modelling what fast convolution is to image recognition.

Possible future work includes parallel training of memory augmented models, applications to autoregressive flows, and replacing decay vector λ_t with structured matrix Λ_t.

References

D. Balduzzi and M. Ghifary. Strongly-typed recurrent neural networks.
G. E. Blelloch. Prefix sums and their applications.
J. Bradbury, S. Merity, C. Xiong, and R. Socher. Quasi-recurrent neural networks.
N. Kalchbrenner, et al. Neural machine translation in linear time.
T. Lei, Y. Zhang, Training RNNs as fast as CNNs.
A. van den Oord, et al. Wavenet: A generative model for raw audio.