

# Parallelizing Linear Recurrent Neural Nets Over Sequence Length

Eric Martin and Chris Cundy

Fill in institutes

## 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.

## Background

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  $x_t$ ,  $\lambda_t$  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.

3	1	5	0	2	4	2	6	1
3	4	9	0	2	6	2	8	9
			+9			+9+6		
3	4	9	9	11	15	17	23	24

Figure 1: Cumulative sum parallelized 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)**

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

## 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  $\tilde{h} = \text{GILR}(x)$

## 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  $k$ .

batch size	LSTM								
	1	2	4	8	16	32	64	128	256
seq length	256	63.6	124	226	323	390	448	467	464
	512	118	227	304	388	460	477	474	471
	1024	189	304	375	435	475	467	463	459
	2048	311	405	469	491	502	497	491	
	4096	412	471	503	509	517	509		
	8192	478	509	514	521	520			
	16384	510	512	520	527				
	32768	510	510	517					
	65536	509	511						
	131072	503							
GILR-LSTM									

**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.

## Synthetic Long-Term Dependency

We tested the GILR-LSTM by comparing to the CuDNN LSTM implementation on a synthetic memorisation task (problem 2b from We compared the performance on variants with different values of  $n$ . We obtained speedups of over 6x measured in wall clock time to convergence. Further, the GILR-LSTM attained convergence when the time dependence of the problem had a length of **one million time steps**.

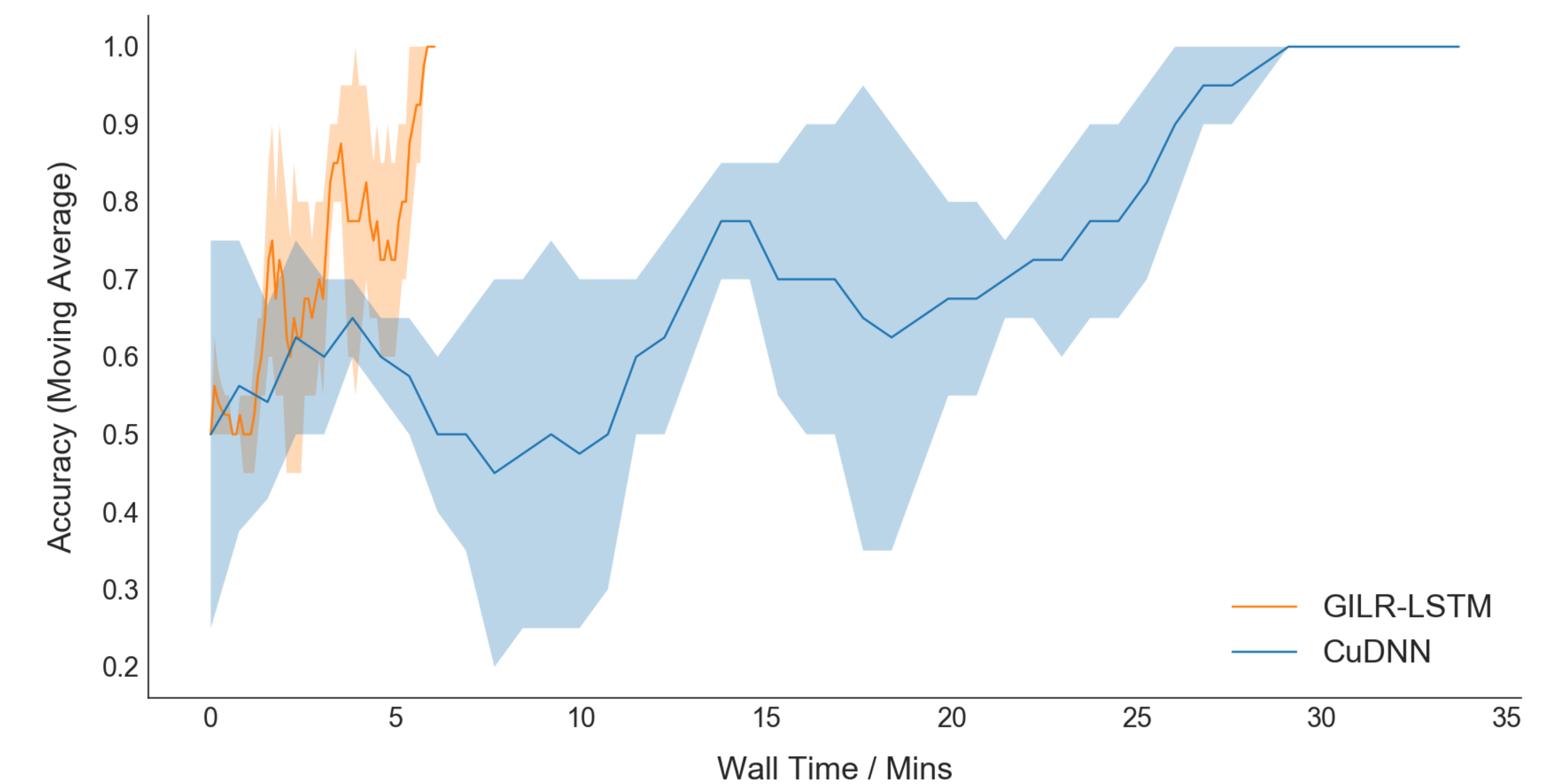


Figure 3: Accuracy on the memorisation task with 8,192 sequence length

## Conclusion

Nunc tempus venenatis facilisis. **Curabitur suscipit** consequat eros non porttitor. Sed a massa dolor, id ornare enim. Fusce quis massa dictum tortor **tincidunt mattis**. Donec quam est, lobortis quis pretium at, laoreet scelerisque lacus. Nam quis odio enim, in molestie libero. Vivamus cursus mi at *nulla elementum sollicitudin*.

## References

## Contact Information

- Web: <http://www.university.edu/smithlab>
- Email: [john@smith.com](mailto:john@smith.com)
- Phone: +1 (000) 111 1111

PLACEHOLDER  
LOGO

PLACEHOLDER  
LOGO