6. Appendix

6.1. Dataset Description & Experiment Details.

Below we describe in details all the datasets used in this study. In addition, we will add the details of the various experiments along with hyper-parameter tuning. We use a mixture of language modelling tasks (corresponds to sequence modelling regime) and terminal prediction tasks (corresponds to long range dependency tasks). Language modelling tasks include variants of the popular Penn Tree Bank (McAuley & Leskovec, 2013) dataset. Terminal prediction tasks include the synthetic Add-Task along with Sequential vision tasks. These together corresponds to the long range dependency datasets.

Add-Task. (Hochreiter & Schmidhuber, 1997) has been used to evaluate long range dependencies in RNN architectures. An example data point consists of two sequences (x_1, x_2) of length T and a target label y. x_1 contains real-valued entries drawn uniformly from $[0, 1]$, x_2 is a binary sequence with exactly two 1s, and the label y is the sum of the two entries in sequence x_1 where x_2 has 1s. Note that a naive strategy is to predict 1 as the output for every input sequence. This naive strategy has a mean squared error of 0.167, the variance of the sum of two independent uniform distributions. We follow (Arjovsky et al., 2016; Kag et al., 2020) to setup experiments on this task. For both the algorithms (BPTT and FPTT), we use episodic training where a train batch size of 128 is presented to the RNN to update its parameters and evaluated using an independently drawn test set.

Our main text uses four different sequence lengths : (a) $T = 200$ used in the ablative experiment (see Figure 2), (b) $T = 500$ used in the Toy example (see Figure 1), (c) the difficult sequence lengths $T = 750$ and $T = 1000$ shown in Figure 3. As used in earlier works (Kag et al., 2020), our experiments set hidden state size as 128. Our architecture is one-layer LSTM followed by one classifier layer. We use $1e-3$ as the learning rate for this experiment. We have used Adam optimizer for this task. For smaller sequence lengths $T = 200, 500$, we run the experiment for 5000 training iterations while for the larger sequence lengths $T = 750, 1000$ we run the experiments for 10000 training iterations. We use $K = 10$ in this task for the proposed algorithm and the default $\alpha = 0.01$ was used in this experiment.

Permute-Pixel MNIST and Pixel-CIFAR-10. are sequential variants of the popular image classification datasets: MNIST (Lecun et al., 1998) and CIFAR-10 (Krizhevsky & Hinton, 2009). MNIST consists of images of 10 digits with shape $28 \times 28 \times 1$, while CIFAR-10 consists of images with shape $32 \times 32 \times 3$. The input images are flattened into a sequence (row-wise). At each time step, 1 and 3 pixels are presented as the input for MNIST and CIFAR datsets respectively. This construction results in Pixel MNIST and CIFAR datasets with 784 and 1024 length sequences respectively. While Permute-MNIST is obtained by applying a fixed permutation on the Pixel MNIST sequence. This creates a harder problem than the Pixel setting since there are no obvious patterns to explore.

MNIST dataset has $60,000$ training and $10,000$ test images while CIFAR-10 dataset has $50,000$ training and $10,000$ test images. Neural network used for this task is one-layer LSTM followed by a classifier layer. Our LSTMs use the hidden state size as 128 and are trained for 200 epochs. We set aside 10% training data for hyper-parameter tuning and once the hyper-parameters are fixed, we use the full data for training and report the performance on the test set. We use the the grid with learning rate choices : $\{0.01, 0.005, 0.001, 0.0001, 0.0005\}$, batch sizes $\{B = 64, 128, 256\}$ and for FPTT we have α choices $\{0.1, 0.5, 0.05, 0.01, 0.001\}$. For the proposed algorithm we use $K = 20$ for this experiment. Note that we use the learning rate schedule defined in (Bai et al., 2019), i.e. we decay the learning rate by a factor of $\frac{1}{2}$ at fixed intervals, i.e. $\{60, 90, 120\}$ epochs. Our ablative experiment which use the CIFAR-10 dataset keep the same experimental setup described here. The corresponding hyper-parameter tuning details are same as the main experiment.

PTB-300. is a word level language modelling task with the difficult sequence length of 300 and has been studied in many previous works to study long range dependencies in language modeling (Zhang et al., 2018; Kusupati et al., 2018). This dataset consists of $929K$ training words, $73K$ validation words, and $82K$ test words with $10K$ vocabulary. An example in this dataset consists of a sentence, where the RNN receives an input at timestep t and has to predict the next word (which will be available on the next timestep). Perplexity is used as the evaluation metric on this dataset, where lower values corresponds to better performance. Validation set is used for tuning the hyper-parameters and once the right set of parameters have been found, evaluation is performed on the test set.

We used the known small configuration to setup this task for PTB word level task except that sequence length has been changed to 300 to model long range dependencies. We only use $1-$layer LSTM with hidden state size of 256. The embedding dimension has been set to the hidden state in accordance to earlier works (Kusupati et al., 2018; Zhang et al., 2018). We train both BPTT and FPTT to 100 epochs and learning rate is decayed by half at every epoch where validation perplexity plateaus. We use SGD optimizer as per the configuration and use initial learning rate of 20. We use $K = 10$ for

FPTT. Note that α values are tuned using the grid search with values $\{0.5, 0.1, 0.05, 0.01, 0.005, 0.001\}$.

PTB-w. is the traditional word level language modelling variant of the PTB dataset. It uses 70 as sequence length and we follow (Yang et al., 2018) to setup this experiment. This dataset consists of $929K$ training words, $73K$ validation words, and $82K$ test words with $10K$ vocabulary. Although this is a small scale dataset for studying language modelling, it has been extensively used by previous works (Merity et al., 2018; Yang et al., 2018; Bai et al., 2019). We follow (Merity et al., 2018) to setup our experiments for BPTT and FPTT. We use three layer LSTM model for this task with embedding dimensions 280. As recommended our hidden states for three layers are 1150. All the other hyper-parameter settings (learning rate, batch size, dropout, other regularizers, dynamic evaluation parameters, etc.) have been borrowed from (Yang et al., 2018) and they are ideal for the BPTT LSTM experiment since this has been tuned by previous works. We report the results with dynamic evaluation(Krause et al., 2018) on the trained model. We use the same architecture and training setup to train LSTMs with both BPTT and FPTT . We use $K = 10$ in our experiments and tune the α values in the grid $\{0.5, 0.1, 0.05, 0.01, 0.005, 0.001\}$ using the validation dataset.

PTB-c is the character level modelling task that uses 150 sequence length. It contains 5M characters for training, $396K$ for validation, and $446K$ for testing, with an alphabet size of 50. The evaluation metric used for this dataset is bits per character (bpc). Similar to perplexity, the lower value of bpc corresponds to better performance. We utilize (Merity et al., 2018) to setup the character level task. We use 3-layer LSTM models as recommended with hidden size 1000 and embedding dimension 200. Remaining hyper-parameters have been kept as it is and they are well tuned for BPTT training as per previous work. We train this model with both BPTT and FPTT with the same setting. We perform similar parameter tuning as in the PTB-w experiment to find α values and set $K = 10$ for this dataset.

6.2. Training Time Comparison.

In this section, we demonstrate that training recurrent neural networks with the proposed method is not computationally expensive than back-propagation through time. Table 7 shows the training time in minutes as measured by the `time` utility in the python language. Note that this refers to the wall-clock time and we ensure that the system is running only one experiment during this process in order to perform fair comparison. As evident from the table, the proposed method fares well in training time when compared to BPTT.

Table 7. **Training time comparison (reported in hours).**

Dataset	BPTT	FPTT
PTB-300	1.01	1.12
Pixel-MNIST	3.29	2.68
Permute-MNIST	3.41	2.97
Pixel-CIFAR-10	3.57	3.25
Add-Task	0.31	0.18

6.3. Impact of α hyper-parameter.

We explore the sensitivity to the α hyper-parameter in the Algorithm 2. We use the PTB-300 language modelling dataset. In this experiment we train FPTT on the following α values: $\{1.0, 0.8, 0.5, 0.1, 0.05, 0.01, 0.005, 0.001\}$. Table 8 lists the validation accuracy for these α values. It shows that FPTT is insensitive to $\alpha \leq 0.5$. Note that very small value of α, i.e. $\alpha \to 0$ would lead FPTT to ignore the regularizer and would only optimize the instantaneous loss at every step resulting in diverging iterates. While very high value of α would lead FPTT to only optimize the regularizer and hence very poor generalization performance.

6.4. Comparison with Online Gradient Descent.

Our parameter update equations perform gradient descent with dynamic regularizer in order to constraint the iterates to be nearby. In this experiment we show that such an additional regularizer is required to achieve better generalization error. We train LSTMs on the PTB-300 dataset with following training algorithms : (a) BPTT : the standard back-propagation through time method, (b) FPTT (OGD) : the proposed algorithm without incorporating the dynamic regularizer, (c) FPTT : the proposed algorithm with dynamic regularizer. Table 9 shows results for these three settings along with the known baselines for this dataset. It can be inferred that incorporating the dynamic regularizer improves the performance of the proposed

Table 8. PTB-300 language modelling (validation perplexity) : various α values.

α	Validation Perplexity
1.0	265
0.8	115
0.5	110
0.1	112
0.05	116
0.01	112
0.005	113
0.001	114

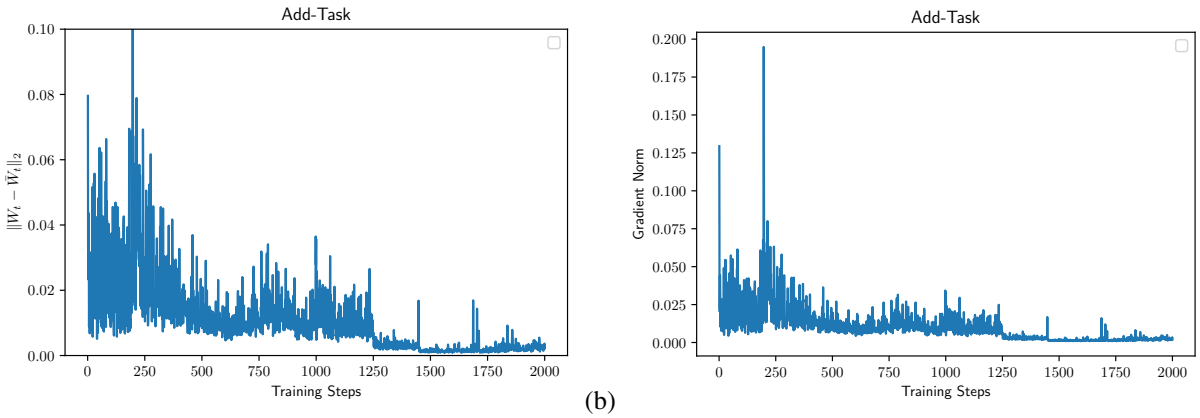
algorithm.

Table 9. Ablative results for PTB word level language modelling : Sequence length (300), 1-Layer LSTM. Comparing the FPTT scheme with and without the dynamic regularizer.

Dataset	**PTB-w**	
	Perplexity	**#Params**
FastGRNN (Kusupati et al., 2018)	116.11	53K
IncrementalRNN (Kag et al., 2020)	**115.71**	30K
SpectralRNN (Zhang et al., 2018)	130.20	31K
LSTM (Zhang et al., 2018)	130.21	64K
LSTM (Kusupati et al., 2018)	**117.41**	210K
LSTM	117.09	210K
FPTT (OGD) LSTM	113.74	210K
FPTT LSTM	**106.27**	210K

6.5. Convergence $W_t - \bar{W}_t$.

In order to show that the proposed algorithm converges to a single average iterate towards the end of the training phase, we show the convergence plot for $\|W_t - \bar{W}_t\|_2$ and the gradient norm for the training steps. Figure 4 shows these plots for the Add-Task with sequence length $T = 200$. As evident from these plots, the gap between W_t and $\bar{W}_t$ closes as the training progresses. During the initial training steps, one can observe that there's a significant difference between W_t and $\bar{W}_t$ and as the training progresses this difference goes to 0. Similarly, the gradient norm reaches 0 by the end of the training phase.



(a) (b)

Figure 4. Add Task ($T = 200$): Convergence plot for $\|W_t - \bar{W}_t\|_2$ and the gradient norm plot. As expected the parameter iterates W_t start to converge to the average iterate $\bar{W}_t$. Similarly the gradient norms start decays to near 0 as the training progresses.

6.6. Copy Task Experiments.

In this section, we perform experiments on the Copy-Task dataset (Arjovsky et al., 2016; Hochreiter & Schmidhuber, 1997). For a fixed length T, an example input sequence consists of values (x, y). Both x and y are sequences of length $T + 20$. The entries of the sequence are drawn from the alphabet $\{a_i\}_{i=0}^{9}$. The first 10 entries of the sequence x are drawn uniformly, independently and with replacement from $\{a_i\}_{i=0}^{7}$. These first 10 entries are the sequence which need to be remembered and referred to as the magic sequence. The next $T - 1$ entries are set to a_8 representing the blank sequence and needs to be ignored or treated as noise in the sequence. The next entry is a single character a_9 which implies a delimiter in the input sequence and remaining entries are set as a_8. The target label y consists of the character a_8 till the length $T + 10$ and followed by the magic 10 character sequence in the remaining 10 locations. The main motivation behind this task is to remember the magic sequence throughout the length T and as T increases this task becomes harder for the recurrent neural networks as they have to remember the magic sequence through a very large sequence length. We measure the error in this task using categorical cross-entropy between the label y and the predicted label $\hat{y}$ output by the RNN after going through the input sequence.

Note that there is a memoryless baseline (Arjovsky et al., 2016) for the Copy-Task. It would be to predict a_8 for the sequence length $T + 10$ and then predict each of the final 10 characters from the set $\{a + i\}_{i=0}^{7}$ independently and uniformly at random. This results in the cross-entropy value of $\frac{10 \log 8}{T+20}$.

In our experiment we use $1-$layer LSTM with hidden state size 128 and batch size 100. We use Adam as the optimizer and $2e-4$ as the learning rate. We provide an independent training set at each step and evaluate the performance on an independent test set of the same batch size. Figures 5 shows the convergence plot for the training LSTMs with BPTT and FPTT on sequence length $T = 30$ and $T = 200$. It can be seen that LSTMs trained with BPTT stay close to the baseline solution for quite a while and then start to decay. While FPTT-LSTMs reach the memoryless baseline quickly and reach the convergence much faster than BPTT.

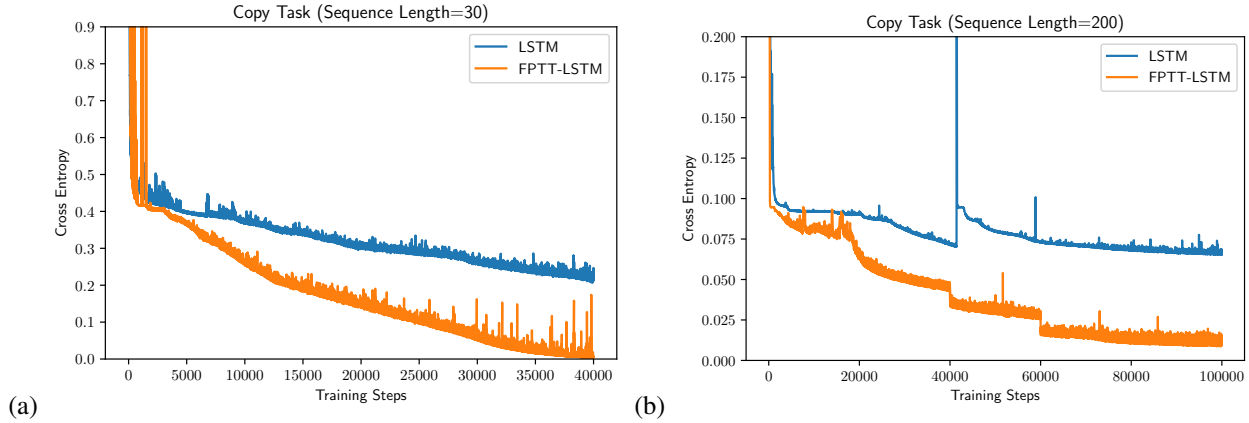


Figure 5. Convergence plot Copy Task for Sequence Lengths (a) $T = 30$: naive strategy results in 0.39 as the solution, and (b) $T = 200$: naive strategy results in 0.09 as the solution.

6.7. Proofs

Proposition 2. *In the Algorithm 2, suppose, the sequence W_t is bounded and converges to a limit point W_∞. Further assume the loss function ℓ_t is $\beta-$Smooth and $\gamma-$Lipschitz. Let the cumulative loss be $F = \frac{1}{T}\sum_{t=1}^{T} \nabla \ell_t(W_\infty)$ after T iterations [5]. It follows that W_∞ is a stationary point of Eq. 3, i.e., $\lim_{L\to\infty} \frac{\partial F}{\partial W}(W_\infty) = 0$.*

Rewriting the first equation in Eq. 6 as:

$$W_{t+1} = \bar{W}_t + \frac{1}{\alpha}(\nabla \ell_{t-1}(W_t) - \nabla \ell_t(W_{t+1})) \tag{7}$$

[5]For simplicity in exposition, we concatenate all the losses ℓ_t into a single online stream and get rid of the index N, that gets repeated to provide T iterations of the gradient updates

We assumed that the sequence W_t is bounded and converges to a limit point W_∞. By Cesaro mean[6] argument

$$W_{t+1} \to W_\infty \implies \frac{1}{T}\sum_{t=1}^{T} W_{t+1} \overset{T\to\infty}{\longrightarrow} W_\infty \tag{8}$$

Summing Eq. 7 over t

$$\begin{aligned}
&\frac{1}{T}\sum_{t=1}^{T} W_{t+1} = \frac{1}{T}\sum_{t=1}^{T} \bar{W}_t + \frac{1}{T}\sum_{t=1}^{T} \frac{1}{\alpha}(\nabla \ell_{t-1}(W_t) - \nabla \ell_t(W_{t+1})) \\
\implies &\frac{1}{T}\sum_{t=1}^{T} W_{t+1} = \frac{1}{T}\sum_{t=1}^{T} \bar{W}_t - \frac{1}{\alpha T}\nabla \ell_T(W_{T+1}) \qquad \text{Telescoping Sum}
\end{aligned}$$

Since loss functions are $\gamma-$Lipschitz, we get that $\nabla \ell_T(W_{T+1})$ term is bounded, i.e. $\|\nabla \ell_T(W_{T+1})\| \leq \gamma$. Hence, $\frac{1}{T}\nabla \ell_T(W_{T+1}) \overset{T\to\infty}{\longrightarrow} 0$.

$$\implies \frac{1}{T}\sum_{t=1}^{T} \bar{W}_t \overset{T\to\infty}{\longrightarrow} W_\infty \tag{9}$$

Summing second equation in Eq. 6 over t, we get

$$\begin{aligned}
\frac{1}{T}\sum_{t=1}^{T} \bar{W}_{t+1} &= \frac{1}{2T}\sum_{t=1}^{T} (\bar{W}_t + W_{t+1}) - \frac{1}{2\alpha T}\sum_{t=1}^{T} \nabla \ell_t(W_{t+1}) \\
\implies \frac{1}{2\alpha T}\sum_{t=1}^{T} \nabla \ell_t(W_{t+1}) &= \frac{1}{2T}\sum_{t=1}^{T} (\bar{W}_t + W_{t+1}) - \frac{1}{T}\sum_{t=1}^{T} \bar{W}_{t+1} \\
&= \frac{1}{2T}\sum_{t=1}^{T} W_{t+1} + \frac{1}{2T}\sum_{t=1}^{T} \bar{W}_t - \frac{1}{T}\sum_{t=1}^{T} \bar{W}_{t+1} \\
&= \frac{1}{2T}\sum_{t=1}^{T} W_{t+1} - \frac{1}{2T}\sum_{t=1}^{T} \bar{W}_t + \frac{1}{T}(\bar{W}_1 - \bar{W}_{T+1}) \\
\implies \lim_{T\to\infty} \frac{1}{2\alpha T}\sum_{t=1}^{T} \nabla \ell_t(W_{t+1}) &= \lim_{T\to\infty} \Big(\frac{1}{2T}\sum_{t=1}^{T} W_{t+1} - \frac{1}{2T}\sum_{t=1}^{T} \bar{W}_t + \frac{1}{T}(\bar{W}_1 - \bar{W}_{T+1}) \Big)
\end{aligned}$$

Note that the quantity $(\bar{W}_1 - \bar{W}_{T+1})$ is finite. Hence, $\lim_{T\to\infty} \frac{1}{T}((\bar{W}_1 - \bar{W}_{T+1})) = 0$. From the Eq. 8 and Eq. 9, the other two terms have limits that exists. Plugging in these values in the above expression we get,

$$\implies \lim_{T\to\infty} \frac{1}{2\alpha T}\sum_{t=1}^{T} \nabla \ell_t(W_{t+1}) = \frac{1}{2}W_\infty - \frac{1}{2}W_\infty + 0 = 0 \tag{10}$$

From $\beta-$smoothness assumption on the loss function, we have that

$$\begin{aligned}
&\implies \|\nabla \ell_t(W_{t+1}) - \nabla \ell_t(W_\infty)\| \leq \beta \|W_{t+1} - W_\infty\| \\
\implies \|\frac{1}{2\alpha T}\sum_{t=1}^{T} \nabla \ell_t(W_{t+1}) &- \frac{1}{2\alpha T}\sum_{t=1}^{T} \nabla \ell_t(W_\infty)\| \leq \frac{\beta}{2\alpha T}\sum_{t=1}^{T} \|W_{t+1} - W_\infty\|
\end{aligned}$$

[6] https://www.ee.columbia.edu/~vittorio/CesaroMeans.pdf

Note $W_{t+1} \to W_\infty \implies \forall \delta > 0, \exists t_\delta \text{ s.t. } \forall t > t_\delta, \|W_{t+1} - W_\infty\| < \delta$

$$\implies \|\frac{1}{2\alpha T}\sum_{t=1}^{T}\nabla \ell_t(W_{t+1}) - \frac{1}{2\alpha T}\sum_{t=1}^{T}\nabla \ell_t(W_\infty)\| \leq \frac{\beta}{2\alpha T}(T - t_\delta)\delta + \frac{\beta}{2\alpha T}\sum_{t=1}^{t_\delta}\|W_{t+1} - W_\infty\|$$

As $T \to \infty$, both terms on the right hand becomes arbitrarily close to 0, i.e. we have sequence of vectors $\{\frac{1}{2\alpha T}\sum_{t=1}^{T}\nabla \ell_t(W_{t+1}) - \frac{1}{2\alpha T}\sum_{t=1}^{T}\nabla \ell_t(W_\infty)\}_{T=1}^{\infty}$ that converge to the 0 vector.

$$\implies \lim_{T\to\infty}\|\frac{1}{2\alpha T}\sum_{t=1}^{T}\nabla \ell_t(W_{t+1}) - \frac{1}{2\alpha T}\sum_{t=1}^{T}\nabla \ell_t(W_\infty)\| = 0$$

From Eq. 10, we know that $\frac{1}{2\alpha T}\sum_{t=1}^{T}\nabla \ell_t(W_{t+1}) \overset{T\to\infty}{\longrightarrow} 0$

$$\implies \frac{1}{2\alpha T}\sum_{t=1}^{T}\nabla \ell_t(W_\infty) \overset{T\to\infty}{\longrightarrow} 0$$

This is the proposed stationarity condition, and our claim follows.