Learning Long-term Dependencies Using Cognitive Inductive Biases in Self-attention RNNs

Giancarlo Kerg [1 2 *] **Bhargav Kanuparthi** [1 2 *] **Anirudh Goyal** [1 2] **Kyle Goyette** [1 2 3] **Yoshua Bengio** [1 2 4] **Guillaume Lajoie** [1 2 5]

Abstract

Attention and self-attention mechanisms, inspired by cognitive processes, are now central to state-of-the-art deep learning on sequential tasks. However, most recent progress hinges on heuristic approaches that rely on considerable memory and computational resources that scale poorly. In this work, we propose a relevancy screening mechanism, inspired by the cognitive process of memory consolidation, that allows for a scalable use of sparse self-attention with recurrence. We use simple numerical experiments to demonstrate that this mechanism helps enable recurrent systems on generalization and transfer learning tasks. Based on our results, we propose a concrete direction of research to improve scalability and generalization of attentive recurrent networks.

1. Introduction

Historically, recurrent neural networks (RNNs) have been the deep network architecture of choice to process sequential inputs with dependencies over various timescales. Just like neural circuits in the brain, they enable *dynamics* that can be shaped to interact with input streams. However, RNNs (including gated RNNs (Schmidhuber & Hochreiter, 1997; Cho et al., 2014)) still struggle with large timescales as their iterative nature leads to unstable information propagation (Bengio et al., 1994; Pascanu et al., 2013; Schmidhuber & Hochreiter, 1997; Hochreiter, 1991). Indeed, Bengio et al. (1994) showed that without making additional assumptions, storing information in a fixed-size state vector in a stable way necessarily leads to vanishing gradients when back-propagating through time (see also (Hochreiter, 1991)).

Several attempts have been made to augment RNN dynamics with external memory to mitigate these issues (Sukhbaatar et al., 2015; Graves et al., 2014; Santoro et al.; Graves et al., 2016), but it is only recently that access to externally stored information has become effective with the introduction of *attention*, and more particularly *soft attention* mechanisms (Bahdanau et al., 2014). There is substantial empirical evidence that attention, especially *self-attention* (Vaswani et al. (2017); Ke et al. (2018)), is very helpful to improve learning and computations over long-term dependencies, and that these types of inductive biases offer promising advantages for generalization and transfer learning (Ke et al., 2018; Goyal et al., 2019). However, this requires to hold inputs and/or past states in memory in order to be considered for attentive computations, a process that typically scales quadratically with sequence length. As for recurrent networks, a promising solution to this has been the use of sparse attention (see e.g. SAB (Ke et al., 2018)), leading to models with varying degrees of reliance on both recurrence and self-attention. However most of these models only delay computational issues by a constant factor and are thus still scaling quadratically with sequence length.

In this paper, we are pushing the idea of sparse attention in RNNs further. We showcase a simple *relevancy screening mechanism* that aims to efficiently consolidate relevant memory, leading to an inductive bias that reduces the size of the computational graph from quadratic to linear in sequence length. The detailed treatment of this mechanism, including supporting theorems for gradient propagation, will be presented in a forthcoming paper. In this present manuscript, we compare various recurrent and attention models with our proposed relevancy screening mechanism on a series of simple numerical experiments highlighting how the underlying cognitive inductive bias helps with generalization. Finally, we discuss how our methods can serve as a building block to derive more sophisticated inductive biases which might further improve memory retrieval and generalization.

[*]Equal contribution [1]Mila, Quebec AI Institute, Canada [2]Université de Montréal, Département d'Informatique et Recherche Opérationelle, Montreal, Canada [3]Université de Montréal, CIRRELT, Montreal, Canada [4]CIFAR senior fellow [5]Université de Montréal, Département de Mathématiques et Statistiques, Montreal, Canada. Correspondence to: Giancarlo Kerg <giancarlo.kerg@gmail.com>.

Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 108, 2020. Copyright 2020 by the author(s).

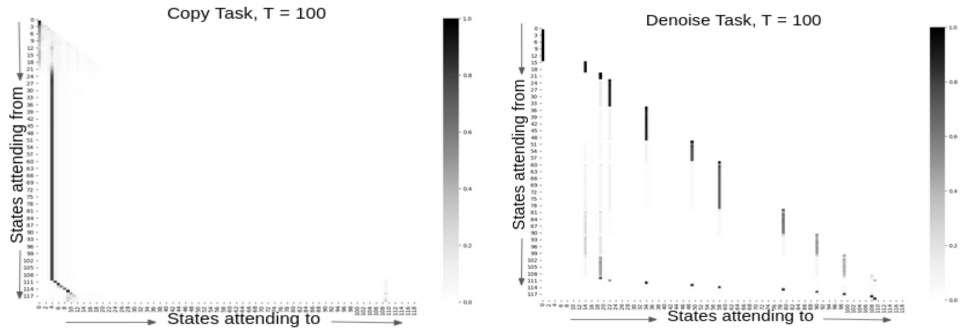

Figure 1. Magnitude of attention weights between states in a trained, fully recurrent and fully attentive model (Bahdanau et al. (2014)). Each pixel in the lower triangle corresponds to the attention weight of the skip connection departing from the the state marked on the y-axis to the state marked on the x-axis. Left shows Copy task, right shows Denoise task. Task details in Section 3

2. Relevancy screening mechanism

Let $x_t \in \mathbb{R}^m$ be the input and $h_t \in \mathbb{R}^n$ be the hidden state at time step t, satisfying the RNN update equation for all $t \geq 1$,

$$h_{t+1} = \phi(Vs_t + Ux_{t+1} + b) \quad (1)$$

$$s_t = f(h_t, c_t) \quad (2)$$

where ϕ is a non-linearity, $f : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$, $V \in \mathbb{R}^{n \times n}$, $U \in \mathbb{R}^{n \times m}$, $b \in \mathbb{R}^n$ and $c_t = \alpha_{1,t}h_1 + \alpha_{2,t}h_2 + \ldots + \alpha_{t,t}h_t$ with $\alpha_{i,t} := \frac{\exp(e_{i,t})}{\sum_{j=1}^{t} \exp(e_{j,t})}$ and $e_{i,t} := a(s_{t-1}, h_i)$, where $a : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ is the attention *alignment function*. Throughout, we assume training is done via gradient descent of a cost function L using back-propagation.

The most notable advances in the use of attention is in purely attention-based systems such as the Transformer (Vaswani et al., 2017), which completely foregoes recurrence and inspired some of the work listed above. While the performance of these systems is impressive, their memory and computation requirements grows quadratically with the total sequence length. To address this issue, many variants that aim to "sparsify" the attention matrix have been proposed. Notably, Ke et al. (2018) developed the Sparse Attentive Backtracking model (SAB), a self-attentive Long Short-Term Memory network (LSTM) (Schmidhuber & Hochreiter, 1997) that leverages sparsity by selecting only the top-k states in memory based on an attention score, propagating gradients only to those chosen hidden states. Building on the SAB model (Ke et al., 2018), we remark that although sparse attention attends to the top-k states at any point in time, attention scores must be computed on all events stored in memory to extract the k best ones. Thus, the resource bottleneck is not controlled by k, but rather by the number of stored events in memory. In SAB, there is a naive attempt to control this number by only recording network states at each 10 time steps. However, this reduces the size of the computational graph only by a constant factor, but retains $O(T^2)$ complexity. Meanwhile, we know that the only important hidden states to conserve for good gradient propagation are the "relevant" ones (i.e. the ones where the associated incoming attention scores are sufficiently bounded away from zero). Thus, we propose to reduce complexity while maintaining good gradient propagation by selectively storing states that are predicted to be relevant in the future, using a *relevancy screening mechanism*.

Algorithm 1 Relevancy Screening

```
 1: procedure: RelRNN($\mathbf{s}_{t-1}, \mathbf{x}_t$)
    Require: Previous macro-state - $\mathbf{s}_{t-1}$
    Require: Input - $\mathbf{x}_t$, $\nu > 0$, $\rho > 0$
    Require: Short-term buffer $s^{(i)}_{t-1} \in S_{t-1}$
    Require: Relevant set $r^{(i)}_{t-1} \in R_{t-1}$
 2: $h_t \leftarrow \phi(V\mathbf{s}_{t-1} + U\mathbf{x}_t + b)$
 3: $S_t = S_{t-1}$.add($h_t$)
 4: if $t - \nu > 0$ then
 5:     $S_t = S_t$.remove($h_{t-\nu}$)
 6: if $t - \rho > 0$ and $C(t-\rho) = True$ then
 7:     $R_t = R_{t-1}$.replaceWith($h_{t-\rho}$)
 8: $M_t = [S_t, R_t]$
 9: for all $m^{(i)} \in M_t$ do
10:     $\tilde{z}^{(i)} \leftarrow v_a^T \cdot \tanh(W_a\mathbf{s}_{t-1} + U_a m^{(i)})$
11: $z \leftarrow$ softmax($\tilde{z}$)
12: $\mathbf{s}_t = h_t + \sum_i z^{(i)} m^{(i)}$
13: return $\mathbf{s}_t$
```

The idea is simple: devise a screening function $C(i)$ which estimates the future relevance of h_i, and store selected states in a *relevant set* $R_t = \{h_i | i < t \wedge C(i) = True\}$ for future attention. In principle, one can explicitly control how R_t grows with t, thus mitigating the complexity scaling outlined above. Here, $C(i)$ could take many forms, the best of which depends on task structure. In what follows, we present an example screening mechanism.

We take inspiration from memory consolidation principles in human cognition (Alberini et al., 2013), which defines the transfer of events from short-term to long-term memory. We remark that for some tasks such as those depicted in Figure 1, relevance varies very little across time. To implement relevancy screening for such tasks, at every time step t we attend to two subsets of the past hidden states. We call the first subset a *short-term buffer* $S_t = \{h_{t-\nu}, h_{t-\nu+1}, .., h_{t-1}\}$ which consists of the hidden states of the last ν time steps, while the second subset is the relevant set R_t. We compute the *relevance score* at time step i, $\beta(i) = \sum_{j=i}^{i+\nu-1} \alpha_{i,j}$, measuring the integrated attention scores over our short-term buffer S_t. More precisely, $C(i)$ is satisfied if $\beta(i)$ is part of the top ρ relevance scores when compared to all previously observed hidden states, where ρ is a fixed hyper-parameter satisfying $\rho \geq |R_t|$ for all t. The pseudo-code in Algorithm 1 describes the screening mechanisms and the interaction between the short-term buffer S_t and a finite size relevant set R_t. '.replaceWith()' is a function replacing the hidden state with the lowest relevance score by the hidden state in the argument.

We note that the sets S_t and R_t give rise to a sparse attention mechanism with sparsity coefficient κ satisfying $\kappa = \nu + \rho \geq |S_t| + |R_t|$. Hence, memory complexity is constant while the $O(T^2)$ bottleneck of computational complexity is replaced by $O((\rho + \nu) \cdot T) = O(T)$.

3. Experiments

Before describing experiments, we make a few remarks. First, we stress that Relevancy Screening can be applied to any semi-parametric attentive model but we refer to the version presented below, which uses an RNN/LSTM base, as RelRNN/RelLSTM ("*Relevance RNN /LSTM*"). Second, our objective is not to find state-of-the-art performance but to highlight a more scalabale usage and the generalization advantages of state relevancy and selective sparsity. In what follows, we denote MemRNN/MemLSTM for vanilla self-attention RNN/LSTM as defined in (Bahdanau et al., 2014). We consider two task categories to highlight distinct impacts of selective attention and recurrence. The first category specifies tasks with sparse dependency chains, highlighting generalization and transfer properties, and the second includes a variety of reinforcement learning (RL) tasks, suggesting that selective relevancy is a key ingredient for generalization in RL. Further implementation details and hyper-parameters can be found in the Appendix A.

3.1. Tasks with sparse dependency chains

A good stereotypical task type that captures sparse sequences of important states are memorization tasks. Here, the network has to memorize a sequence of relevant characters presented among several non-relevant ones, store it for some given time delay and output them in the same order as they were read towards the end of the sequence.

Transfer Copy task: An important advantage of sparse attentive recurrent models such as RelRNN is that of generalization. This is illustrated by the Transfer Copy scores (Hochreiter & Schmidhuber, 1997) where models are trained on Copy task for $T = 100$ and evaluated for $T > 100$. Table 1 shows results for the models listed above, in addition to h-detach (Arpit et al., 2018), an LSTM-based model with improved gradient propagation. Importantly, where purely recurrent networks performed well on the original task, all fail to transfer, with discrepancy growing with T. As expected, MemRNN and SAB keep good performance but RelRNN outperforms them, with almost perfect performance for all T. While both SAB and RelRNN use sparse memory storage and retrieval, the distinguishing factor is RelRNN's use of relevancy screening, indicating it's importance for transfer. The performance of RelLSTM on Transfer Copy is exactly the same as RelRNN.

Table 1. Results for Transfer Copy task.

T	100	200	400	2000	5000
orth-RNN	99%	4%	16%	10%	0%
expRNN	100%	86%	73%	58%	50%
MemRNN	99%	**99%**	**99%**	92%	OOM
RelRNN	**100%**	**99%**	**99%**	**99%**	**99%**
LSTM	99%	64%	48%	19%	14%
h-detach	100%	91%	77%	51%	42%
SAB	99%	95%	95%	95%	95%
RelLSTM	**100%**	**99%**	**99%**	**99%**	**99%**

Table 2. Results for Denoise task.

T	100	300	500	1000	2000
orth-RNN	90%	71%	61%	29%	3%
expRNN	34%	25%	20%	16%	11%
MemRNN	99%	**99%**	**99%**	**99%**	OOM
RelRNN	**100%**	**99%**	**99%**	**99%**	**99%**
LSTM	82%	41%	33%	21%	15%
GORU	92%	93%	91%	93%	73%
SAB	99%	**99%**	**99%**	**99%**	**99%**
RelLSTM	**100%**	**99%**	**99%**	**99%**	**99%**

Denoise task Jing et al. (2019): This generalizes the Copy task as the symbols that need to be copied are now randomly distributed among the T time steps, requiring the model to selectively pick the inputs that need to be copied. We test our method against all the previously mentioned models in addition to GORU (Jing et al., 2019) for various values of T (Table 2). RelLSTM performs exactly as RelRNN and again, we see RelRNN maintain complete performance across all T values, outperforming all purely recurrent models. MemRNN performs as RelRNN/RelLSTM but fails to train due to memory overflow beyong $T = 500$.

3.2. MiniGrid reinforcement learning tasks

We next consider a few tasks from MiniGrid (Chevalier-Boisvert et al., 2018) in the OpenAI gym (Brockman et al., 2016) in which an agent must get to certain goal states. We use a partially observed formulation of the task, where the agent only sees a small number of squares ahead of it. These tasks are difficult to solve with standard RL algorithms, due to (1) the partial observability of the environment and (2) the sparsity of the reward, given that the agent receives a reward only after reaching the goal. We use Proximal Policy Optimization (PPO, (Schulman et al., 2017)) along with LSTM, MemLSTM, and RelLSTM as the recurrent modules. All models were each trained for 5000000 steps on each environment. The hyperparameters used for RelLSTM are $\nu = 5$ and $\rho = 5$. Our goal is to compare generalization of the solutions learned by each model by training on smaller version of an environment and testing it on a larger version. On the *MiniGrid-DoorKey-5x5-v0* environment the average reward for LSTM is 0.94, MemLSTM is 0.94 and RelLSTM is 0.93. On transferring the learned solution to the *16x16* version of that environment the average reward for LSTM is 0.09, MemLSTM is 0.31 and RelLSTM is $\mathbf{0.44}$. In summary, we find that transfer scores for RelLSTM are much higher than other tested models for several environments.

Environment	LSTM	MemLSTM	RelLSTM
		Train	
RedBlueDoors-6x6	**0.97**	**0.97**	**0.97**
GoToObject-6x6	0.81	**0.85**	0.84
MemoryS7	**0.96**	0.4	0.94
GoToDoor-5x5	**0.28**	0.17	0.25
Fetch-5x5	0.48	0.42	**0.5**
DoorKey-5x5	**0.94**	**0.94**	0.93
		Test	
RedBlueDoors-8x8	**0.95**	**0.95**	**0.95**
GoToObject-8x8	0.66	0.66	**0.74**
MemoryS13	0.25	0.24	**0.30**
GoToDoor-8x8	0.13	0.11	**0.15**
Fetch-8x8	0.38	0.44	**0.45**
DoorKey-16x16	0.09	0.31	**0.44**

Table 3. Average Train and Test Rewards for MiniGrid Reinforcement Learning task. The models were trained on the smaller version of the environment and tested on the larger version to test to generalization of the solution learned.

4. Conclusion & Discussion

We identify event relevancy as a key concept to efficiently scale attentive systems to very long sequential computations. This is illustrated via a *Relevancy Screening Mechanism*, inspired by the cognitive process of memory consolidation in the brain, that efficiently selects network states, called relevant events, to be committed to long-term memory based on a screening heuristic operating on a fixed-size short-term memory buffer. We showcase the benefits of this mechanism in an attentive RNN and LSTM which we call RelRNN and RelLSTM respectively, using simple but illustrative numerical experiments, and demonstrate the optimal trade-off between memory usage and good gradient propagation it achieves.

While our proposed relevancy screening mechanism exploits "local" attention scores (measured while events are in short-term memory buffer), we acknowledge other types of relevancy could be formulated with heuristics better suited to distinct environments enhancing the effects of the used cognitive inductive bias on generalization. For instance, promising directions include leveraging predictive coding principles to select "surprising events", or neural networks could be used to learn the screening function $C(i)$ in an end-to-end fashion. Another promising direction is to improve the representation of the hidden states in such a way that one wouldn't need to attend over a large subset of past states, even if the number of past relevant states becomes large. Inspired by pattern replays in the hippocampus (Foster & Wilson, 2006), one could attempt to define a mechanism which reorganizes the non-parametric memory offline, in order to improve the representation of the hidden states in such a way that retrieval becomes more efficient.

Finally, we argue that memory selectivity is an important concept for generalization and computational efficiency in self-attentive recurrent neural networks, while serving as a building block to construct more sophisticated cognitively inspired inductive biases. Further work is necessary to identify the key mechanisms present in our approach that enable this learning flexibility. As a step toward building this understanding, we note that a relevancy screening process enables a balance between recurrent dynamics and self-attention which leads to improved computational complexity and memory requirements. A detailed analytical treatment of this can be found in a forthcoming paper.

References

Alberini, C., Johnson, S., and Ye, X. *Memory Reconsolidation*, pp. 81–117. 12 2013. ISBN 9780123868923. doi: 10.1016/B978-0-12-386892-3.00005-6.

Arpit, D., Kanuparthi, B., Kerg, G., Ke, N. R., Mitliagkas, I., and Bengio, Y. h-detach: Modifying the lstm gradient towards better optimization. *arXiv preprint arXiv:1810.03023*, 2018.

Bahdanau, D., Cho, K., and Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv e-prints*, art. arXiv:1409.0473, Sep 2014.

Bengio, Y., Simard, P., and Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. 2016. URL `http://arxiv.org/abs/1606.01540`.

Chevalier-Boisvert, M., Willems, L., and Pal, S. Minimalistic gridworld environment for openai gym. `https://github.com/maximecb/gym-minigrid`, 2018.

Cho, K., van Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, 2014.

Foster, D. J. and Wilson, M. A. Reverse replay of behavioural sequences in hippocampal place cells during the awake state. *Nature*, 440(7084):680–683, 2006.

Goyal, A., Lamb, A., Hoffmann, J., Sodhani, S., Levine, S., Bengio, Y., and Schölkopf, B. Recurrent independent mechanisms. *arXiv e-prints*, art. arXiv:1909.10893, Sep 2019.

Graves, A., Wayne, G., and Danihelka, I. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S. G., Grefenstette, E., Ramalho, T., Agapiou, J., et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471, 2016.

Hochreiter, S. Untersuchungen zu dynamischen neuronalen netzen [in german] diploma thesis. *TU Münich*, 1991.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Jing, L., Gulcehre, C., Peurifoy, J., Shen, Y., Tegmark, M., Soljacic, M., and Bengio, Y. Gated orthogonal recurrent units: On learning to forget. *Neural computation*, 31(4):765–783, 2019.

Ke, N. R., Goyal, A., Bilaniuk, O., Binas, J., Mozer, M. C., Pal, C., and Bengio, Y. Sparse attentive backtracking: Temporal credit assignment through reminding. In *Advances in Neural Information Processing Systems*, pp. 7640–7651, 2018.

Pascanu, R., Mikolov, T., and Bengio, Y. On the difficulty of training recurrent neural networks. *ICML*, 2013.

Santoro, A., Faulkner, R., Raposo, D., Rae, J., Chrzanowski, M., Weber, T., Wierstra, D., Vinyals, O., Pascanu, R., and Lillicrap, T. Relational recurrent neural networks. In *Advances in Neural Information Processing Systems 31*.

Schmidhuber, J. and Hochreiter, S. Long short-term memory. *Neural Computation*, 1997.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. 2017. URL `http://arxiv.org/abs/1707.06347`.

Sukhbaatar, S., szlam, a., Weston, J., and Fergus, R. End-to-end memory networks. In *Advances in Neural Information Processing Systems 28*, pp. 2440–2448. 2015. URL `http://papers.nips.cc/paper/5846-end-to-end-memory-networks.pdf`.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *NeurIPS*, 2017.

Appendix

A. Additional Results

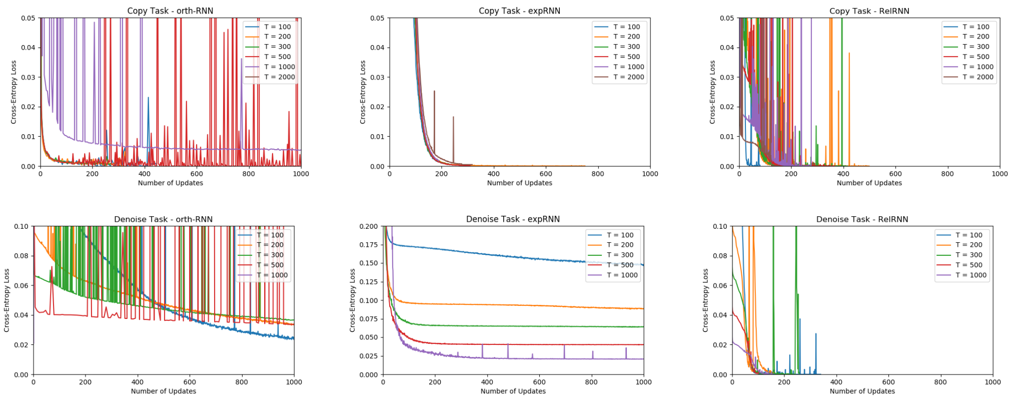

Figure 2. Cross-entropy vs training updates for Copy (top) and Denoise (bottom) tasks for $T = \{100, 200, 300, 500, 1000, 2000\}$. 1 unit of the x-axis is equal to 100 iterations of training with the exception of expRNN where 1 unit on the x-axis is 10 iterations of training.

Table 4. Results for Copy Task

T	LSTM	orth-RNN	expRNN	MemRNN	SAB	RelRNN	RelLSTM
100	100%	100%	100%	100%	100%	100%	100%
200	100%	100%	100%	100%	100%	100%	100%
300	100%	100%	100%	100%	100%	100%	100%
500	12%	100%	100%	100%	100%	100%	100%
1000	12%	80%	100%	100%	100%	100%	100%
2000	12%	11%	100%	OOM	100%	100%	100%

Table 5. Hyperparameters used for Copy task

Model	lr	optimizer	non-linearity	ν	ρ
orthRNN	0.0002	RMSprop	modrelu	-	-
expRNN	0.0002	RMSprop	modrelu	-	-
LSTM	0.0002	Adam	-	-	-
RelRNN	0.0002	Adam	tanh	10	10

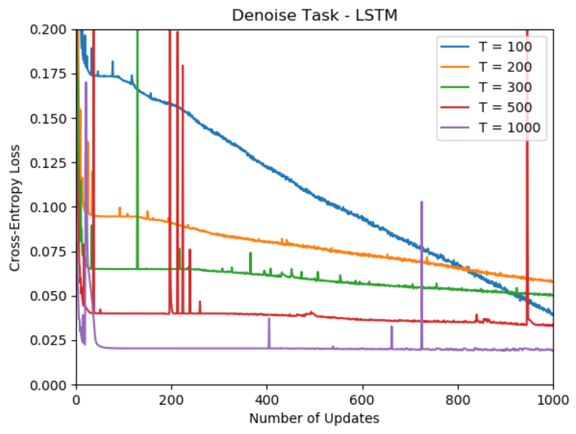

Figure 3. Training curves for LSTM on Denoise task

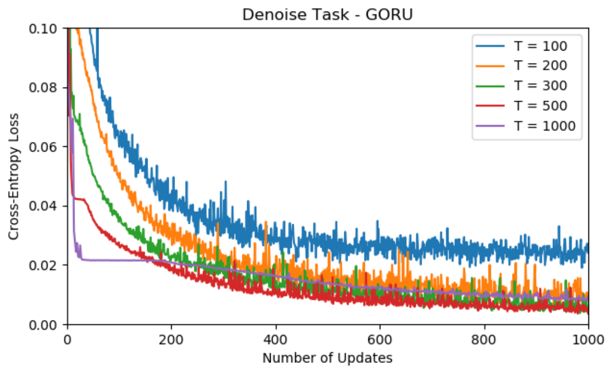

Figure 4. Training curves for GORU on Denoise task

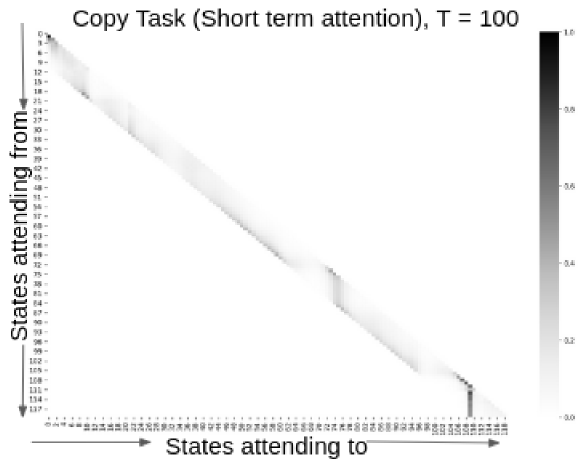

Figure 5. Heatmap of attention scores on Copy task when only doing attention over the Short Term Buffer.

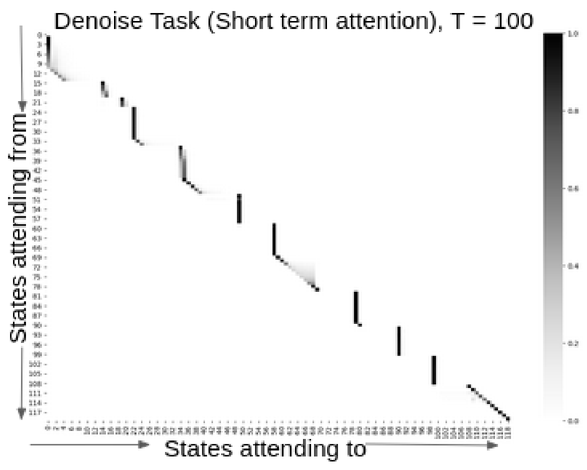

Figure 6. Heatmap of attention scores on Denoise task when only doing attention over the Short Term Buffer.

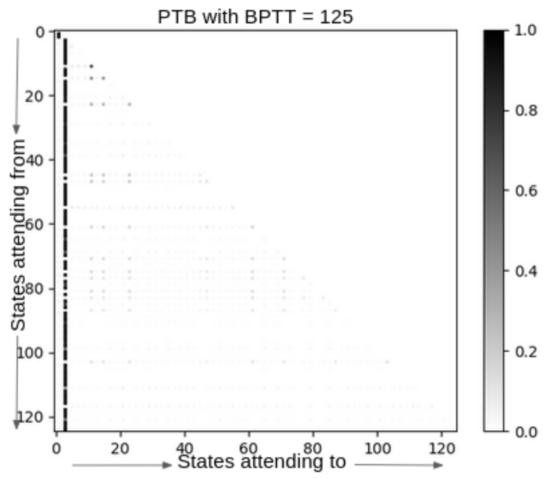

Figure 7. Heatmap of attention scores on PTB task training with full attention and BPTT of 125

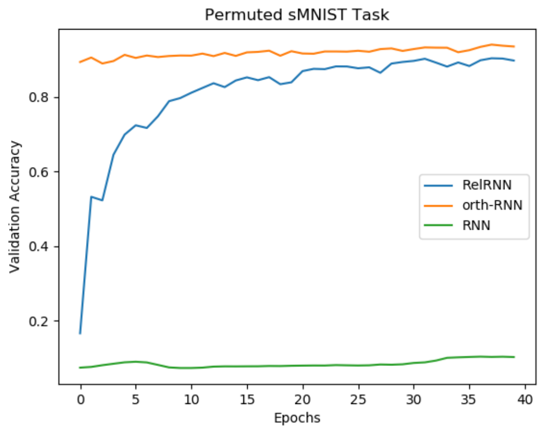

Figure 8. Validation accuracy curves for pMNIST

Table 6. Hyperparameters used for Denoise task

Model	lr	optimizer	non-linearity	ν	ρ
orthRNN	0.0002	RMSprop	modrelu	-	-
expRNN	0.0002	RMSprop	modrelu	-	-
LSTM	0.0002	Adam	-	-	-
GORU	0.001	RMSprop	-	-	-
RelRNN	0.0002	RMSprop	modrelu	10	10

Table 7. PTB and pMNIST results.

	PTB Task		**pMNIST**
Model	BPC	Accuracy	Accuracy
RNN	1.56	66%	90.4%
orth-RNN	1.53	66%	93.4%
expRNN	1.49	68%	**96.6**%
RelRNN	**1.43**	**69**%	92.8%
LSTM	**1.36**	**73**%	91.1%
h-detach	-	-	92.3%
SAB	1.37	-	94.2%
RelLSTM	**1.36**	**73**%	**94.3**%

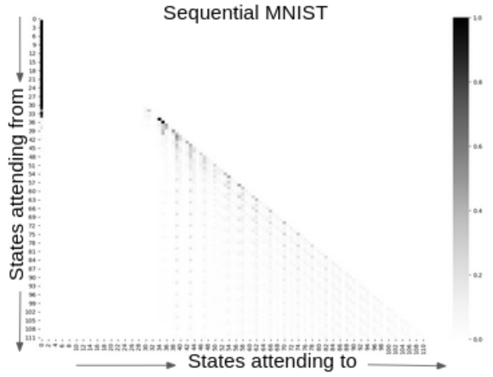

Figure 9. Heatmap of attention scores on MNIST digit classification. 7 pixels were grouped at each time step to make visualization of heatmap easier.

Table 8. Hyperparameters used for sequential MNIST

Model	lr (lr orth)	optimizer	non-linearity	ν	ρ
orthRNN	0.0001	Adam		modrelu	-
-					
expRNN	0.0001(0.00001)	Adam	modrelu	-	-
LSTM	0.0002		-	-	-
GORU			-	-	
RelRNN	0.0003	Adam	modrelu	10	10

Table 9. Hyperparameters used for PTB

Model	lr (lr orth)	optimizer	non-linearity	ν	ρ
orthRNN	0.001	Adam	tanh	-	-
expRNN	0.003(0.0003)	Adam	tanh	-	-
LSTM	0.0002		-	-	-
GORU			-	-	
RelRNN	0.0003	Adam	tanh	10	5