Training Recurrent Neural Networks Online by Learning Explicit State Variables

Somjit Nath, Vincent Liu, Alan Chan, Xin Li, Adam White, Martha White

ICLR 2020

Partially Observable Online Prediction Problem

- **Online prediction problem**
  - For each time step $t = 1,2,\ldots,n$, the agent observes $\mathbf{o}_t$, makes a prediction $\hat{\mathbf{y}}_t$, and sees the actual outcome $\mathbf{y}_t$
  - Suffer a prediction loss ℓ_t
- **Partially observable**: $\mathbf{o}_t$ is not sufficient to make accurate predictions of $\mathbf{y}_t$, i.e., $p(\mathbf{y}_t \mid \mathbf{o}_t, \mathbf{o}_{t-1}, \ldots, \mathbf{o}_1) \neq p(\mathbf{y}_t \mid \mathbf{o}_t)$

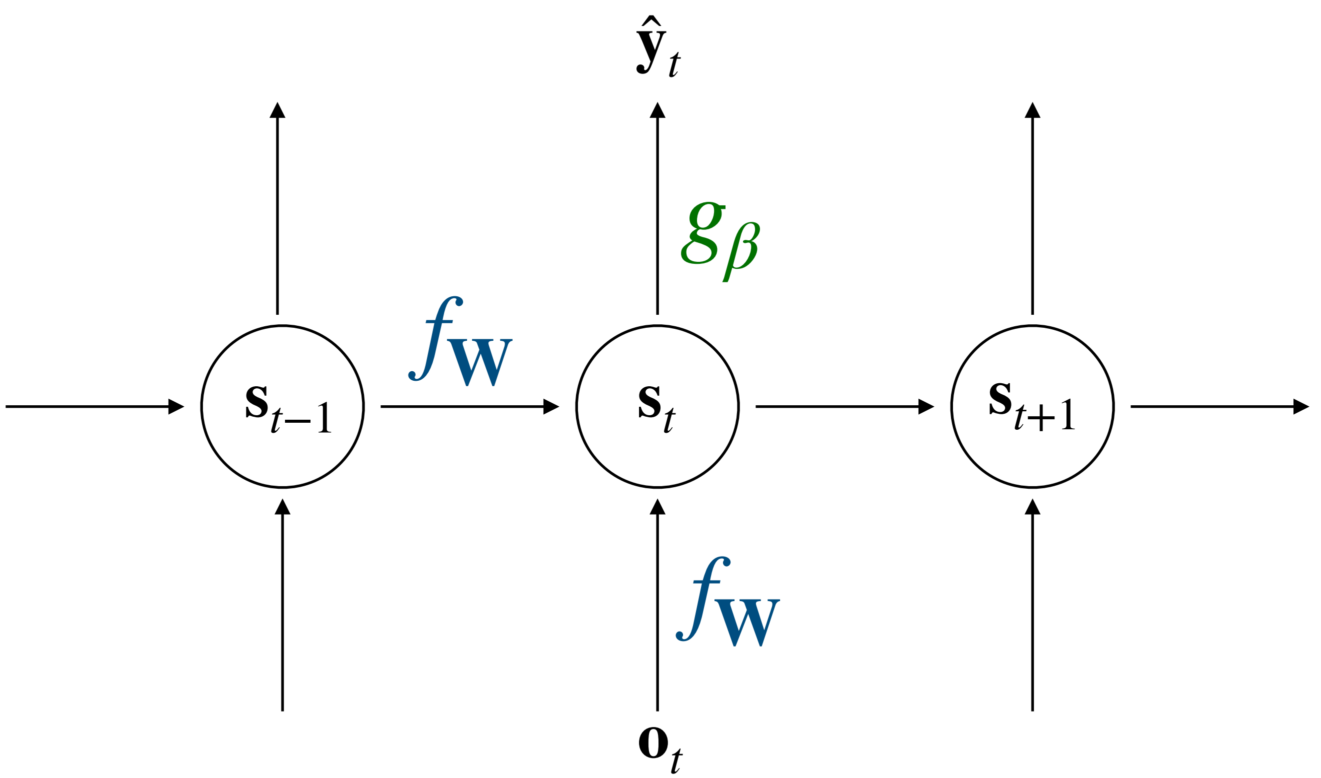
RNN Objective

- Goal: learn a **state-update function** $f_{\mathbf{W}} : \mathbb{R}^k \times \mathbb{R}^d \to \mathbb{R}^k$: $\mathbf{s}_t = f_{\mathbf{W}}(\mathbf{s}_{t-1}, \mathbf{o}_t)$, and a **prediction function** $g_\beta : \mathbb{R}^k \to \mathbb{R}^m$: $\hat{\mathbf{y}}_t = g_\beta(\mathbf{s}_t)$

- RNN minimizes the objective, for some start state s_0:

$$\min_{\mathbf{W},\beta} \sum_{i=1}^{n} \ell_\beta(f_{\mathbf{W}}(\underbrace{\ldots f_{\mathbf{W}}(\overbrace{f_{\mathbf{W}}(\mathbf{s}_0, \mathbf{o}_1)}^{\mathbf{s}_1}, \mathbf{o}_2), \ldots, \mathbf{o}_i)}_{\mathbf{s}_i}; \mathbf{y}_i)$$

- where $\ell_\beta : \mathbb{R}^k \times \mathbb{R}^m \to \mathbb{R}$ is a prediction loss function



Stability and Computational Issues in Training RNNs Online

- RNNs are typically trained using Truncated Backpropagation-through-time (**T-BPTT**) or approximations to Real-Time Recurrent Learning (**RTRL**)
  - T-BPTT is not robust to long-term dependencies
  - RTRL has high computational complexity per step (not practical)
- We investigate an alternative direction for optimizing RNNs, that does not attempt to estimate long gradients back-in-time.

The Fixed Point Objective: Breaking Dependencies with Explicit State Variables

- Define $\mathbf{S} \doteq [\mathbf{s}_0, \ldots, \mathbf{s}_n]$, $\mathbf{O} \doteq [\mathbf{o}_0, \ldots, \mathbf{o}_n]$.

apply state-update operation once

- Define $F_{\mathbf{W}}(\mathbf{S}, \mathbf{O}) \doteq [\mathbf{S}_{:,0}, f_{\mathbf{W}}(\mathbf{S}_{:,0}, \mathbf{O}_{:,1}), \ldots, f_{\mathbf{W}}(\mathbf{S}_{:,n-1}, \mathbf{O}_{:,n})]$

- A constrained minimization with an auxiliary variable $\mathbf{S}$:

$$\min_{\beta,\mathbf{W},\mathbf{S}} \sum_{i=1}^{n} \ell_\beta(f_{\mathbf{W}}(\mathbf{s}_{i-1}, \mathbf{o}_i); \mathbf{y}_i) \quad \text{s.t. } \mathbf{S} = F_{\mathbf{W}}(\mathbf{S}, \mathbf{O})$$

a solution $\mathbf{S}$ is a fixed point of the system defined by $F_{\mathbf{W}}(\,\cdot\,, \mathbf{O})$

- Reformulate into an unconstrained objective:

$$L(\beta, \mathbf{W}, \mathbf{S}) = \frac{1}{n} \sum_{i=1}^{n} \ell_\beta(f_{\mathbf{W}}(\mathbf{s}_{i-1}, \mathbf{o}_i); \mathbf{y}_i) + \frac{\lambda}{2n} \sum_{i=1}^{n} \|\mathbf{s}_i - f_{\mathbf{W}}(\mathbf{s}_{i-1}, \mathbf{o}_i)\|_2^2$$

Lagrange multiplier **quadratic penalty**

T-step Fixed Point Objective

- We can generalize the one step of propagation, simply by generalizing the fixed-point operator. Consider the more general T-step operator:

apply state-update operation T times

$$F_{T,\mathbf{W}}(\mathbf{S},\mathbf{O}) = [\mathbf{S}_{:,0},\mathbf{S}_{:,1},\ldots,\mathbf{S}_{:,T-1},\underbrace{f_{\mathbf{W}}(\ldots f_{\mathbf{W}}(f_{\mathbf{W}}(\mathbf{S}_{:,0},\mathbf{O}_{:,1}),\mathbf{O}_{:,2}),\ldots),\mathbf{O}_{:,T})}_{\hat{\mathbf{s}}_T(\mathbf{s_0},\mathbf{W})},\ldots$$

$$\underbrace{f_{\mathbf{W}}(\ldots f_{\mathbf{W}}(f_{\mathbf{W}}(\mathbf{S}_{:,n-T},\mathbf{O}_{:,n-T+1}),\mathbf{O}_{:,n-T+2}),\ldots),\mathbf{O}_{:,n})}_{\hat{\mathbf{s}}_n(\mathbf{s_{n-T}},\mathbf{W})}]$$

- We minimize the T-step objective:

$$L(\beta,\mathbf{W},\mathbf{S}) = \frac{1}{n}\sum_{i=T}^{n}\ell_\beta(\hat{\mathbf{s}}_i(\mathbf{s}_{i-T},\mathbf{W});\mathbf{y}_i) + \frac{\lambda}{2n}\sum_{i=T}^{n}||\mathbf{s}_i - \hat{\mathbf{s}}_i(\mathbf{s}_{i-T},\mathbf{W})||_2^2$$

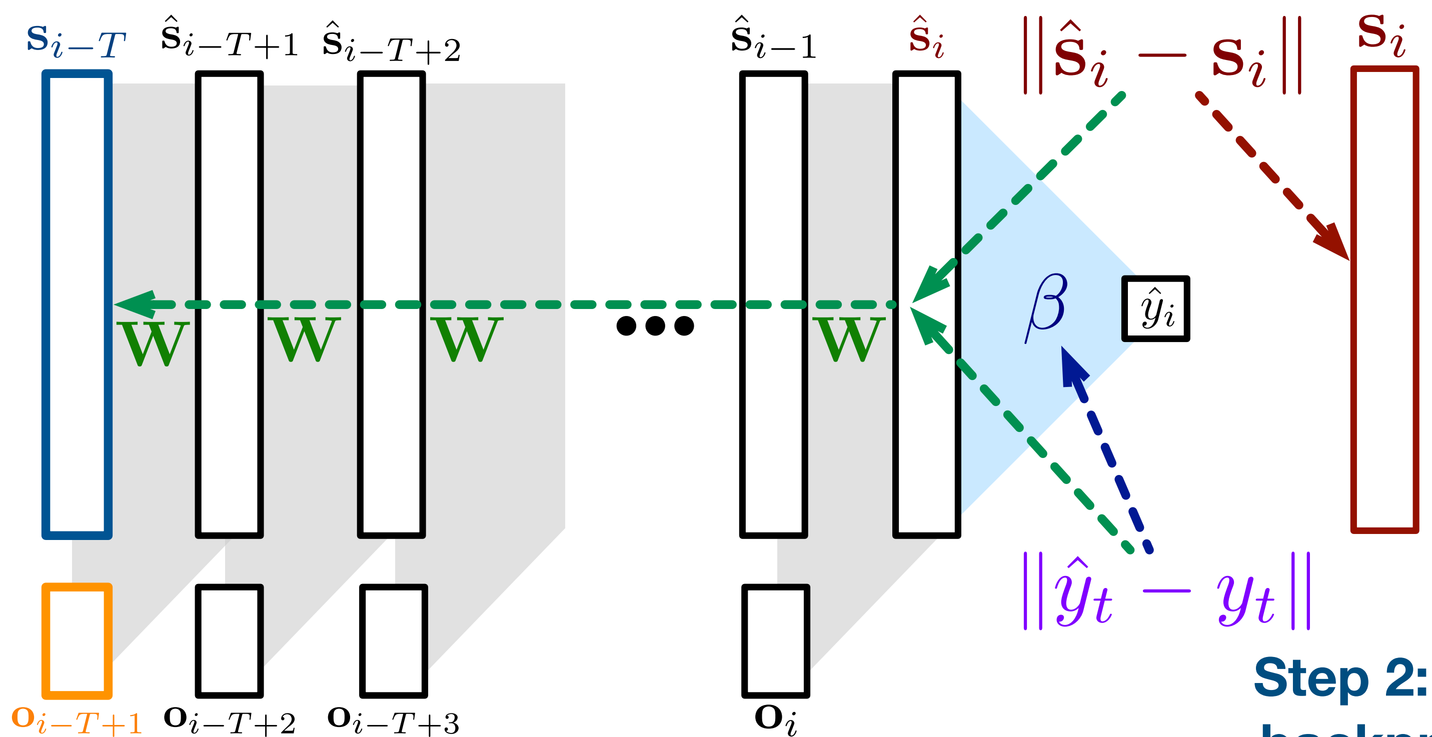
Optimizing the New Objective: Fixed Point Propagation

Use a replay buffer to keep transitions

buffer (before update) $\{\mathbf{s}_{t-n}, \mathbf{o}_{t-n+1}, y_{t-n+1}, \mathbf{s}_{t-n+1}, \mathbf{o}_{t-n+2}, y_{t-n+2}, \ldots, \boxed{\mathbf{s}_i}, \ldots, \mathbf{s}_{t-1}, \mathbf{o}_t, y_t, \mathbf{s}_t\}$

Step 1: Sample a trajectory of length T

sample $\{\boxed{\mathbf{s}_{i-T}}, \boxed{\mathbf{o}_{i-T+1}}, \mathbf{o}_{i-T+2}, \ldots, \mathbf{o}_i, \boxed{y_i}, \boxed{\mathbf{s}_i}\}$



Step 2: Compute the objective, backprop T-steps back in time

buffer (**after** update) $\{\mathbf{s}_{t-n}, \mathbf{o}_{t-n+1}, y_{t-n+1}, \ldots \boxed{\mathbf{s}_{i-T}}, \mathbf{o}_{i-t+1}, y_{i-t+1}, \ldots, \boxed{\mathbf{s}_i}, \mathbf{o}_{i+1}, y_{i+1} \ldots, \mathbf{s}_t\}$

Step 3: Update state variables in the buffer

Algorithm 1 Fixed Point Propagation (FPP)

Input: a truncation parameter T, mini-batch size B, and number of updates per step M
Initialize weights $\mathbf{W}$ and $\boldsymbol{\beta}$ randomly
Initialize state $\mathbf{s}_0 \leftarrow \mathbf{0} \in \mathbb{R}^d$
Initialize an empty buffer B of size N
for $t \leftarrow 1, 2, ...$ **do**
  if B is full **then**
    Remove the oldest transition
  end if
  Observe $\mathbf{o}_t$, $\mathbf{y}_t$, and compute $\mathbf{s}_t = f_{\mathbf{W}}(\mathbf{s}_{t-1}, \mathbf{o}_t)$
  Add $(\mathbf{s}_t, \mathbf{o}_t, \mathbf{y}_t)$ to buffer B
  if $t \geq T$ **then**
    for $j \leftarrow 1, \cdots, M$ **do** ▷ Multiple updates
      Sample a mini-batch of size B, of trajectories of length T, from the buffer:
        $\{(\mathbf{s}_{i_l-T}, \mathbf{o}_{i_l-T}, \ldots, \mathbf{s}_{i_l}, \mathbf{o}_{i_l}, \mathbf{y}_{i_l})\}_{l=1}^{B}$ where i_l is the index of the l-th mini-batch
      Compute the average mini-batch loss and update $\{\mathbf{s}_{i_l-T}, \mathbf{s}_{i_l}\}_{l=1}^{B}$, $\mathbf{W}$ and $\boldsymbol{\beta}$
      Update $\{\mathbf{s}_{i_l-T}, \mathbf{s}_{i_l}\}_{l=1}^{B}$ in the buffer
    end for
  end if
end for

Theoretical Results

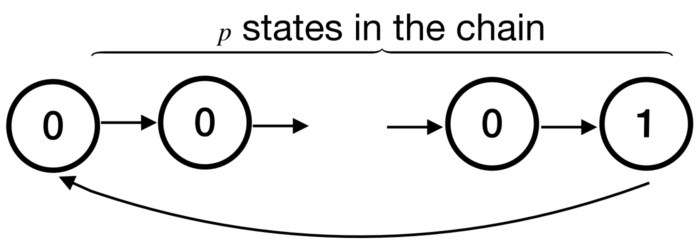
- Key result 1: Convergence of FPP to a stationary point for a fixed buffer (Theorem 1) $\Longrightarrow$ FPP is a sound strategy for using replay buffer to train RNNs

- Key result 2: Recovering RNN solutions when $\lambda \to \infty$ (Theorem 2) $\Longrightarrow$ Our reformulation has not changed the solution.

- *Results rely on results for auxiliary variables in optimization and results using auxiliary variables for neural networks

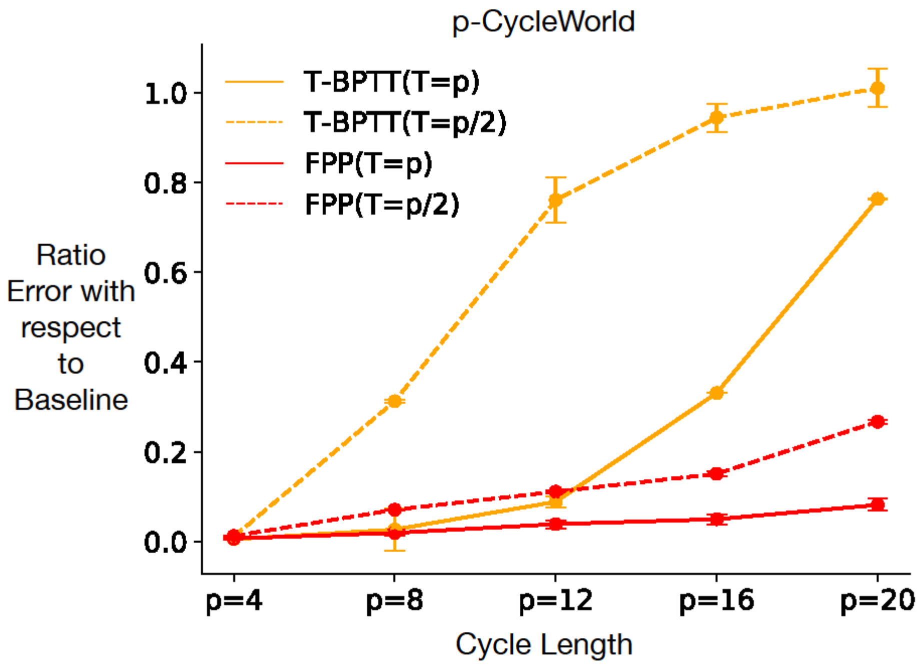
Experimental Results

- Simulation Datasets: CycleWorld, StochasticWorld

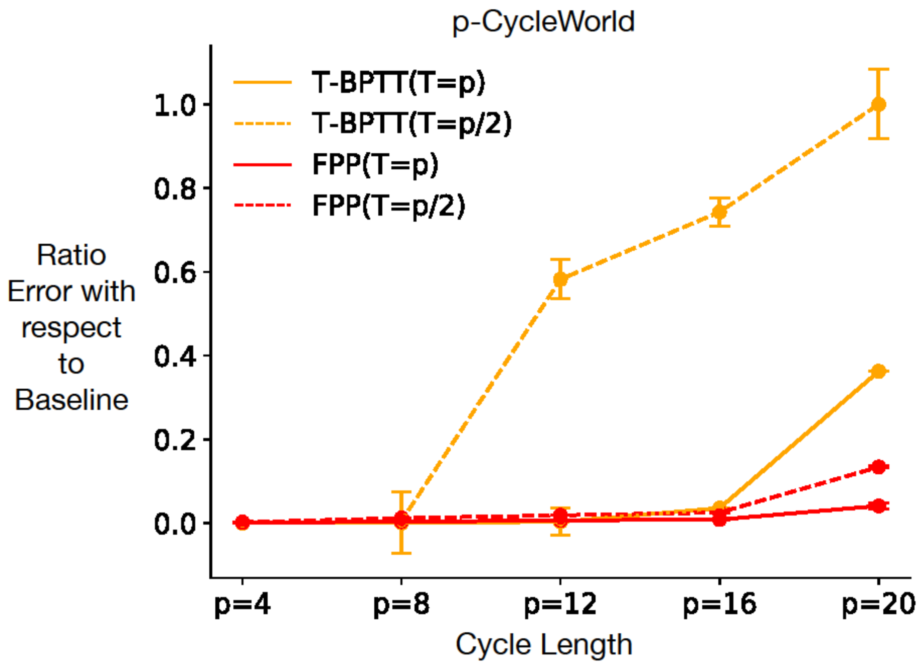


- Real Datasets: Sequential MNIST, PTB, WikiText-2
- Compare FPP, T-BPTT, non-overlap T-BPTT, UORO with varying T (Figure 2 and 3)
  - Key result 1: FPP is robust to T
- Compare FPP and FPP without state updating (Figure 4)
  - Key result 2: FPP can take advantage of multiple updates and mini-batch updates

CycleWorld

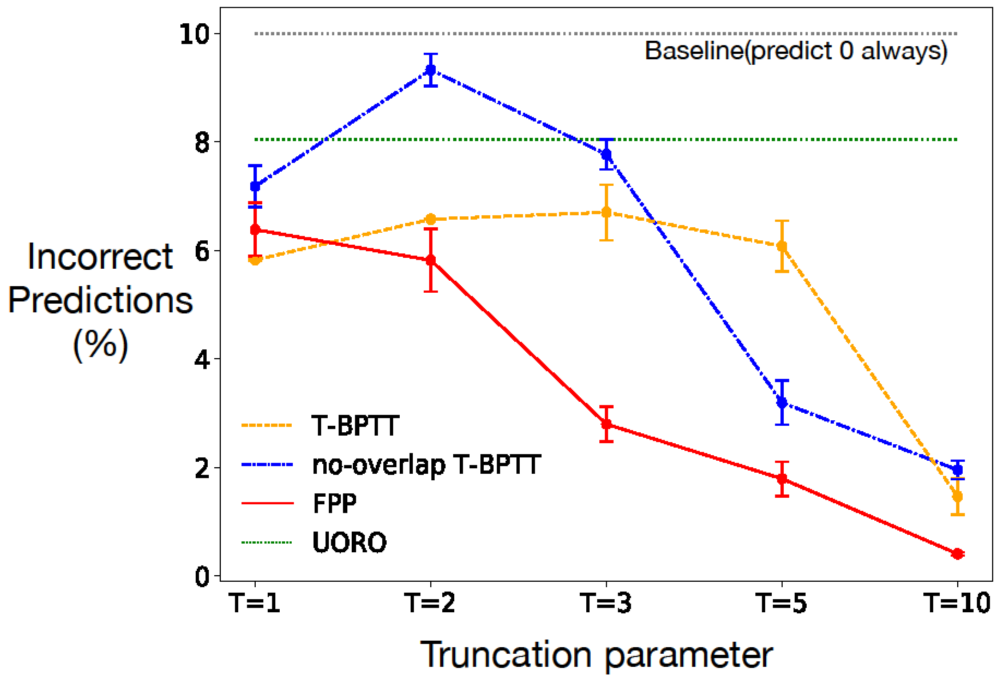


Early Learning
(2.5k steps)

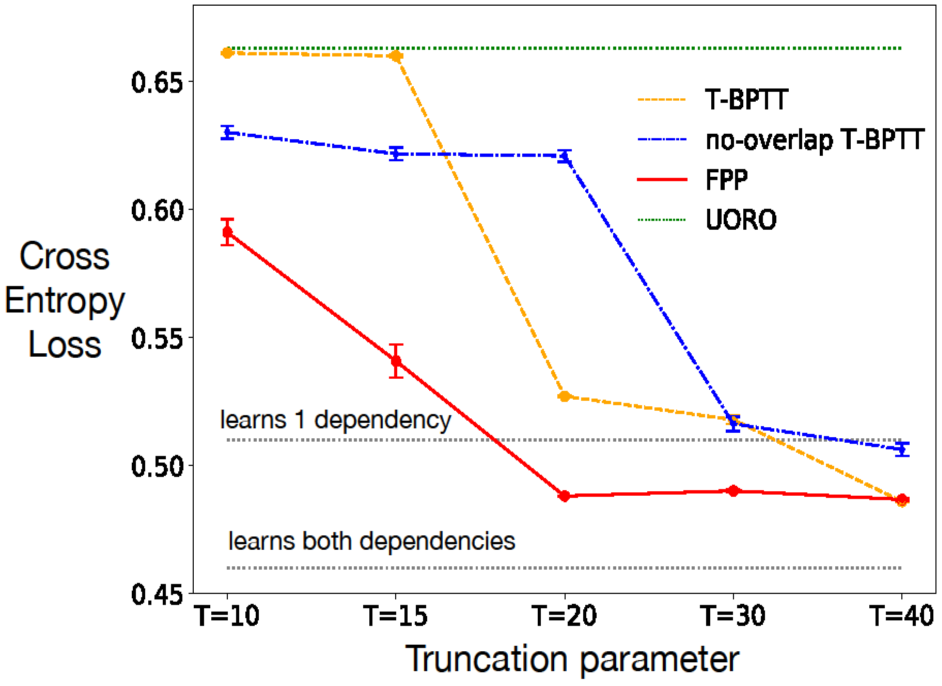


Learning with more data
(15k steps)

Comparison with other Algorithms

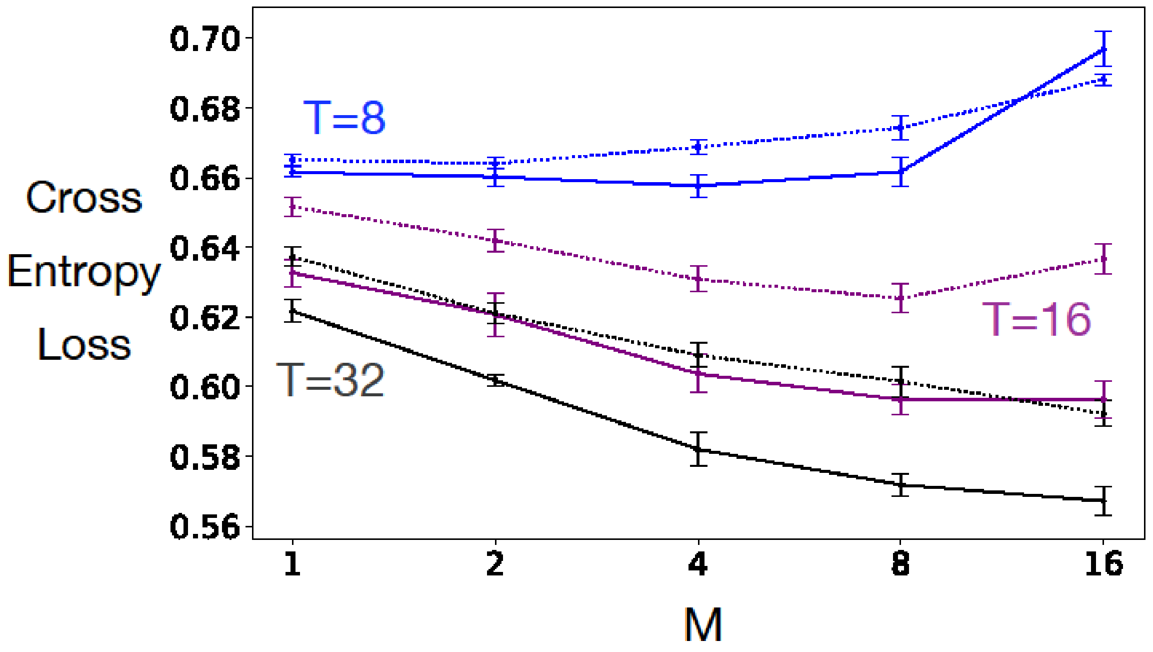


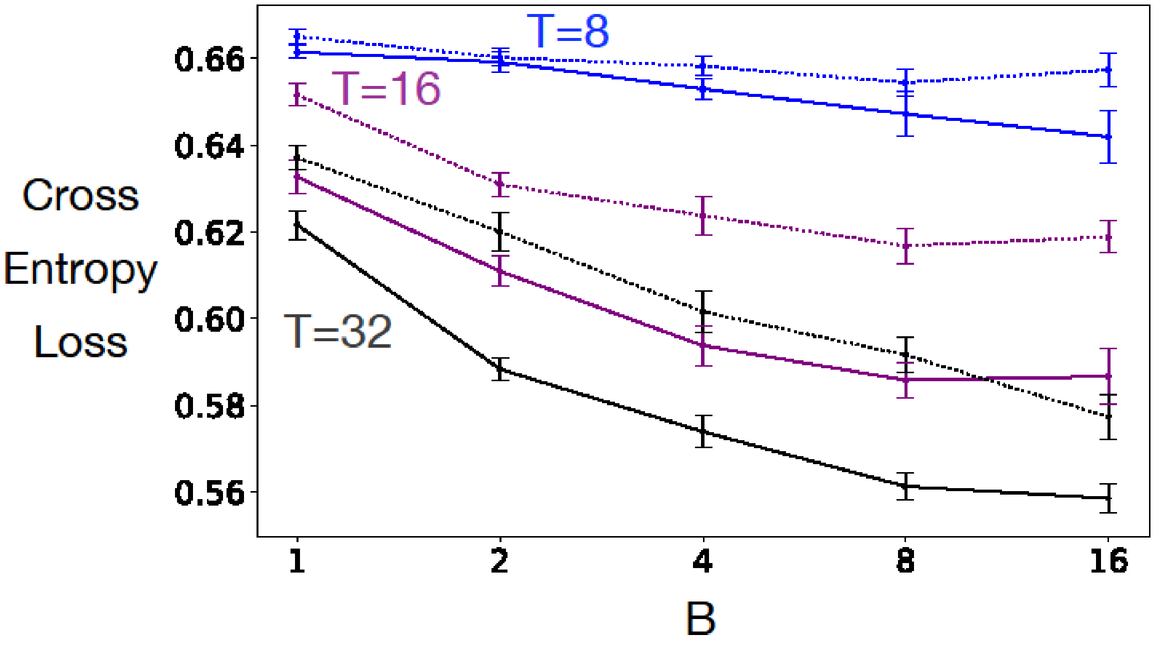
CycleWorld



Stochastic Dataset

Multiple & Batch Updates





Thank you!

- We introduced a new objective to explicitly learn state variables for RNNs, which breaks gradient dependence back in time.

- We theoretically showed the approach is sound and empirically demonstrated improved training robustness.

- FPP could be a promising direction for robustly training RNNs, without the need to compute or approximate long gradients back in time.

RL & AI amii UNIVERSITY OF ALBERTA