RNN Training along Locally Optimal Trajectories via Frank-Wolfe Algorithm

Yun Yue *, Ming Li *, Venkatesh Saligrama †, Ziming Zhang*
{yyue, mli12, zzhang15}@wpi.edu, srv@bu.edu

January 15, 2021

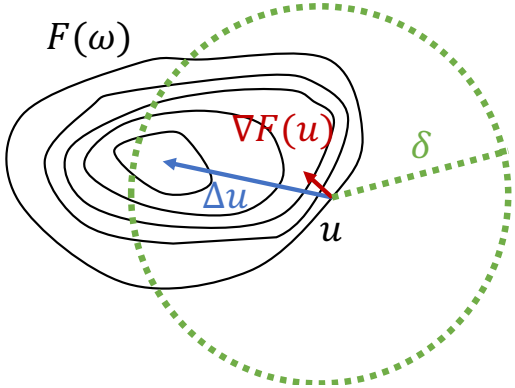
Instability of training RNNs

$$\min_{\omega} \left[F(\omega) \stackrel{def}{=} \mathbb{E}_{\mathcal{B}\sim\mathcal{X}\times\mathcal{Y}} f(\mathcal{B};\omega) \stackrel{def}{=} \sum_{(x,y)\sim\mathcal{B}} \ell(y, \mathrm{z}_M; \omega_\ell) \right]$$
$$\text{s.t. } \mathrm{z}_m = h(x_m, \mathrm{z}_{m-1}; \omega_h), \ \forall \ m \in [M], \tag{1}$$

- The number of time steps, M, is large where long-term dependencies exist among the data;
- The state transmission function, h, involves multiple hidden states such as in deep RNNs;
- The data samples, $\mathcal{X}$, are very noisy or the true signal is weak.

Instability of training RNNs (cont.)

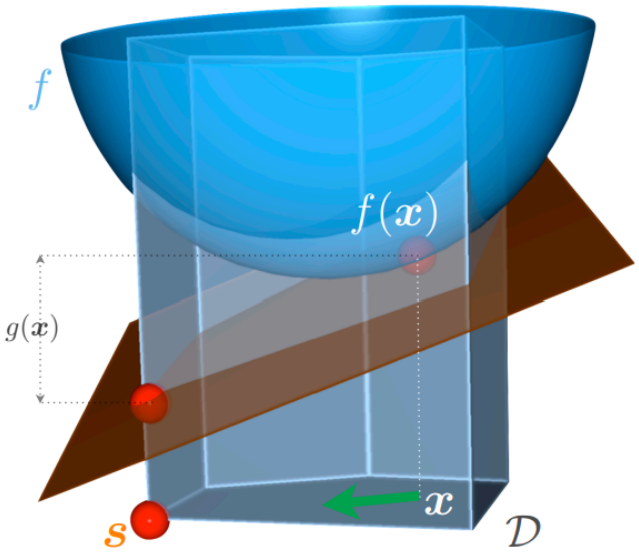
- TBPTT, Gradient norm clipping, Weight matrix identity initialization, different activation functions *etc.* ;
- Gated RNNs, Unitary & Orthogonal RNNs, Lipschitz RNN, FastRNN, Incremental RNNs (iRNNs), Independently RNN (IndRNN) *etc.* ;



Proposed method.

Trust-Region *vs.* Projected Gradient *vs.* Frank-Wolfe

- Trust-Region:
  $f(x) \approx \tilde{f}(x) = f(c) + \nabla f(c)^T (x - c) + \frac{1}{2!}(x - c)^T H(c)(x - c);$
- Projected Gradient: May lead to slow convergence due to the vanishing/exploding gradients;
- Frank-Wolfe;



(From Jaggi 2011)

Frank-Wolfe RNN Optimizer

Input : objective f, norm p, local radius $\delta_t, \forall t$, max numbers of iterations K, T

Output: RNN weights ω

Randomly initialize ω_0;

for $t = 1, \cdots, T$ **do**

  $\Delta\omega_{t,0} \leftarrow 0$;

  for $k = 1, \cdots, K$ **do**

    $s_{t,k} \leftarrow \arg\min_{s \in \mathcal{C}(p,\delta_t)} \langle s, \nabla_{\Delta\omega} F(\omega_{t-1} + \Delta\omega_{t,k-1}) \rangle$;

    $\Delta\omega_{t,k} \leftarrow (1 - \frac{1}{k})\Delta\omega_{t,k-1} + \frac{1}{k} s_{t,k}$;

  end

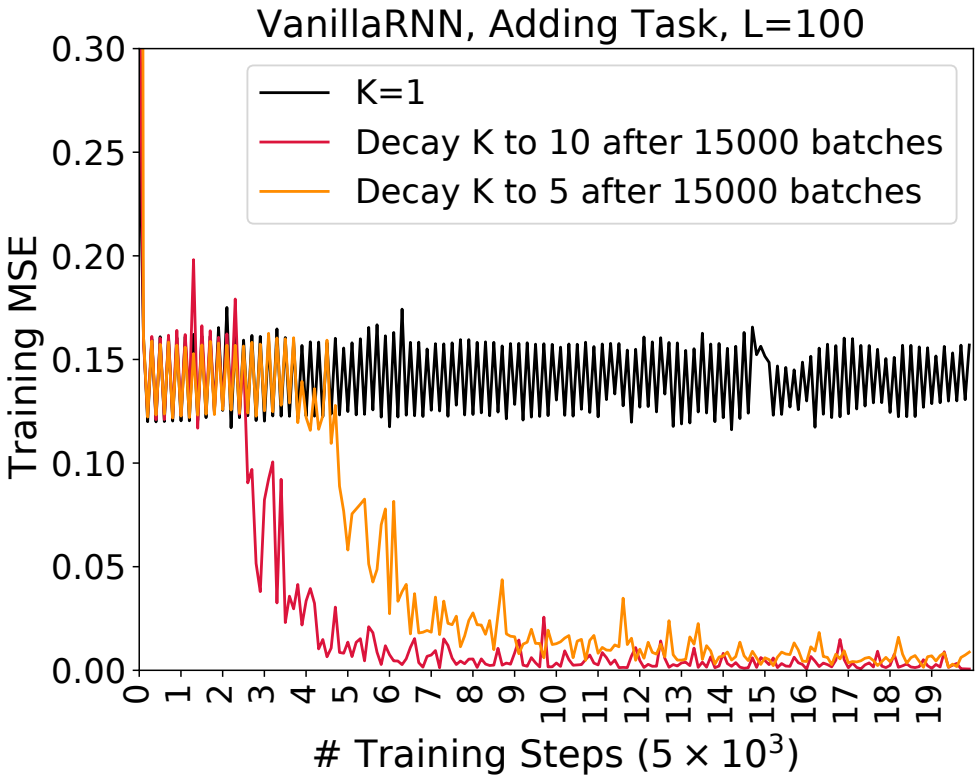
  $\omega_t \leftarrow \omega_{t-1} + \eta \Delta\omega_{t,K}$;

end

return ω_T;

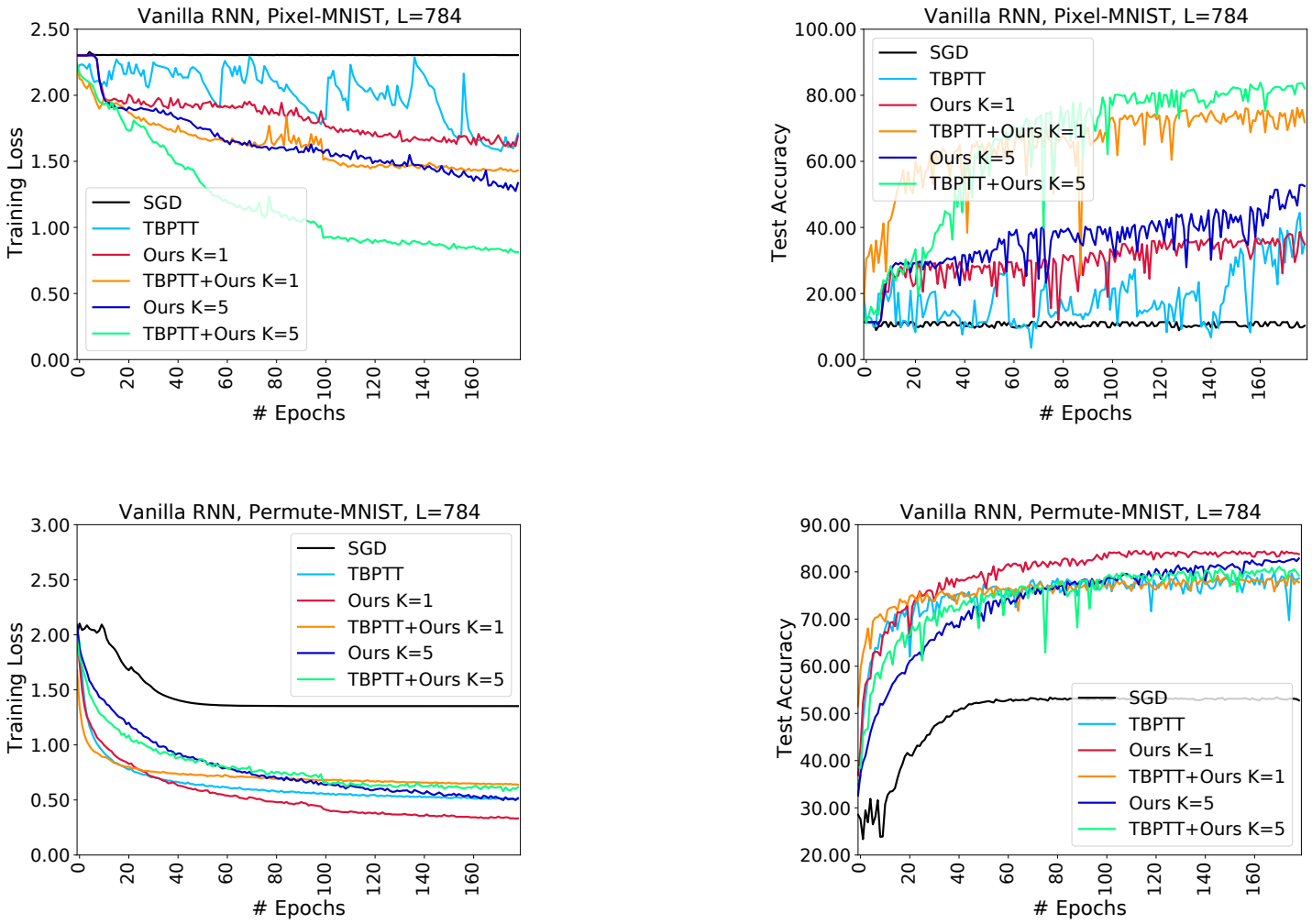
Sublinear convergence rate of $O(1/\epsilon)$ for ϵ error (proof in paper).

Experiments: Adding Task



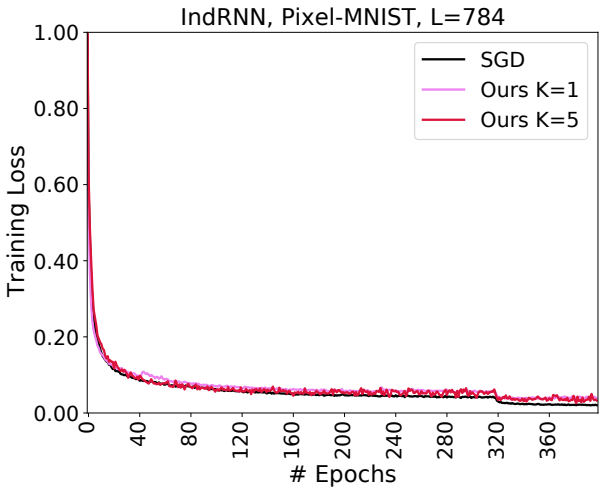
Training loss of Adding Task

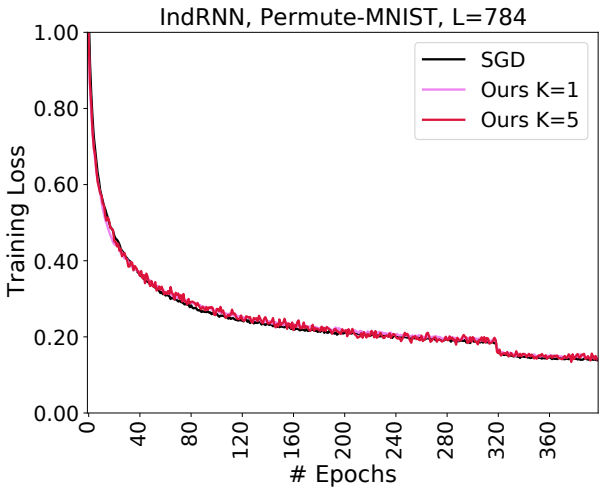
Experiments: Pixel-MNIST & Permute-MNIST



Training loss and test accuracy on Pixel-MNIST and Permute-MNIST

Experiments: Pixel-MNIST & Permute-MNIST





Training loss on Pixel-MNIST and Permute-MNIST with indRNN

Table 1: Test accuracy (%) (training hours) of IndRNN

Dataset	**Acc. (Time)**		
	Baseline	Ours K=1	Ours K=5
Pixel-MNIST	98.88 (4.84)	98.73 (3.46)	98.82 (2.55)
Permute-MNIST	93.00 (4.92)	92.87 (3.68)	92.59 (2.41)

Experiments: HAR-2 & Noisy HAR-2

Table 2: Test accuracy (%) and training time (hr) of RNN

Method	**HAR-2**	**Time**	**Noisy-HAR-2**	**Time**
SGD	87.66	0.17	74.38	0.17
SGD+Clipping	93.36	0.13	74.38	0.13
TBPTT	93.62	0.38	86.20	0.56
LSTM+Adam	94.40	0.14	92.12	0.17
Ours K=1	93.52	0.15	86.04	0.14
Ours K=5	94.11	0.14	89.36	0.14
Ours K=10	93.65	0.37	89.52	0.35
Ours+BN	94.37	0.36	89.38	0.41
TBPTT+Ours	94.01	0.35	89.28	0.84
LSTM+Ours	94.95	0.19	**92.41**	0.42
IndRNN	95.73	0 .46	91.20	0.45
IndRNN+Ours	**96.55**	0.13	92.15	0.17

RNN Training along Locally Optimal Trajectories via Frank-Wolfe Algorithm

Yun Yue *, Ming Li *, Venkatesh Saligrama †, Ziming Zhang*
{yyue, mli12, zzhang15}@wpi.edu, srv@bu.edu

January 15, 2021