Continual Learning of Recurrent Neural Networks by Locally Aligning Distributed Representations Supplementary Material

Alexander Ororbia*, Ankur Mali, C. Lee Giles, *Fellow, IEEE,* and Daniel Kifer

TABLE I
COMPLEXITY ANALYSIS OF VARIOUS RNN LEARNING ALGORITHMS (FOR M LAYERS OF n UNITS) FOR BOTH OFFLINE AND ONLINE SCENARIOS.

	Offline		**Online (per time step)**	
Algorithm	Time	Space	Time	Space
BPTT	$O(n^2LM)$	$O(n^2M)$	$O(n^2TM)$	$O(n^2M)$
TBPTT	$O(n^2hM)$	$O(n^2M)$	$O(n^2hM)$	$O(n^2M)$
RTRL	$O(n^4M)$	$O(n^3M)$	$O(n^4M)$	$O(n^3M)$
UORO	$O(n^2LM)$	$O(n^2M)$	$O(n^2M)$	$O(n^2M)$
P-TNCN (LRA)	$O(n^2LM)$	$O(n^2M)$	$O(n^2M)$	$O(n^2M)$

APPENDIX

In this appendix, we present a complexity analysis of the P-TNCN trained via LRA as well as several of the key algorithms used to train RNNs. In addition, we present a qualitative exploration of the P-TNCN's ability to process out-of-domain inputs, an extra experiment in zero-shot adaptation, and a small stream experiment to demonstrate how the model works with discrete-valued nonlinearities.

On Temporal and Spatial Complexity: To appreciate the value of the P-TNCN and its learning procedure, LRA, one should consider the temporal and spatial complexities in both the offline and online stream-driven learning scenarios. We examine the key algorithms one could use to train an M-layered RNN (in this case, the analysis is further restricted to simple Elman-style RNNs), i.e., BPTT, TBPTT, RTRL, and UORO. Each layer has n units. In the *offline* setting, the RNN is to process and adapt sequences of length L. In the *online* setting, the RNN is process and adapt to an infinitely-long sequence (i.e. a data stream), where T marks the length of the sequence seen up until the current point of time t. In Table I we present the results of this analysis for the settings described above (with some complexity results from [1], which we extend). Note that (for any time-step) the upper bound calculation for all RNN-based learning approaches is dominated by the computation of the recurrent weight matrix update (an $n \times n$ weight matrix, as was the case in [1]).

TBPTT reduces its time complexity, compared to BPTT, in both settings by only unrolling over a history buffer of length h. In the *offline* setting, we note that the P-TNCN trained via LRA operates with the same complexity as BPTT. However, note that our approach to training circumvents many of the issues associated with backprop's unrolling, i.e., vanishing gradients and restriction to differentiable activations. P-TNCN has the same time complexity as UORO (but better accuracy, as seen in the main paper) and better time/space complexity than RTRL. When a stream is being processed step by step, BPTT requires the RNN being trained to be unrolled back until the start of the sequence *at each time step*, yielding a time complexity of $O(n^2TM)$. TBPTT reduces this a bit by only maintaining a fixed buffer of length h, resulting in a complexity of $O(n^2hM)$. The complexities of P-TNCN (LRA), UORO, and RTRL do not change in the online setting (again, the advantage of P-TNCN here is in accuracy).

Alexander Ororbia, Computer Science, Rochester Institute of Technology, Rochester, NY 14623 USA e-mail: ago@cs.rit.edu (see https://www.cs.rit.edu/∼ago/).

Ankur Mali, Daniel Kifer, and C. Lee Giles, Pennsylvania State University.

TABLE II
ONE-SHOT TRAINING PERFORMANCE OF BOUNCING MNIST MODELS ON BOUNCING NOTMNIST FOLLOWED BY ZERO-SHOT ADAPTIVE PERFORMANCE TO BOUNCING FASHION MNIST (FMNIST).

	MNIST $\rightarrow$ NotMNIST		**NotMNIST $\rightarrow$ FMNIST**	
Model,1-0-shot	**CE**	**SE**	**CE**	**SE**
LSTM, BPTT	882.71	269.65	689.96	101.892
LSTM, SAB	883.61	272.58	691.28	103.778
LSTM, RTRL	863.12	239.62	640.09	97.018
P-TNCN	**842.13**	**215.01**	**603.34**	**82.841**

Out-of-Domain Samples: To explore the P-TNCN's ability to process out-of-domain inputs, we generated new sequences from the Bouncing MNIST and NotMNIST processes as was done in [2] where each sequence contained either only one object (digit or character) or three objects that bounced around. Since our P-TNCNs were only trained on sequences with two objects bouncing around, they have never been exposed to sequences with one or three objects. In Figure 1, we show the predictions generated by a trained P-TNCN and see that it does a reasonably good job at predicting a single object bouncing around, which stands in contrast to what was discovered in [2] (which only investigated the case of Bouncing MNIST), where the LSTM trained with BPTT was found to "hallucinate" a second digit over top the first/original one. With respect to the three-object sequence sample, we also observe that the P-TNCN is able to maintain its ability to roughly track multiple objects in space (even if not as easily as it could with two digits), and does *not* seem to merge the digits into blobs as the LSTM of [2] does on MNIST. However, even thought the P-TNCN appears to do a much better job of predicting the near future, due to its dynamic error units, incorporating an iterative attention mechanism could serve to further improve the P-TNCN's generative abilities.

Additional Zero-Shot Adaptation Results: This extra experiment extends the zero-shot adaptation experiment pre-

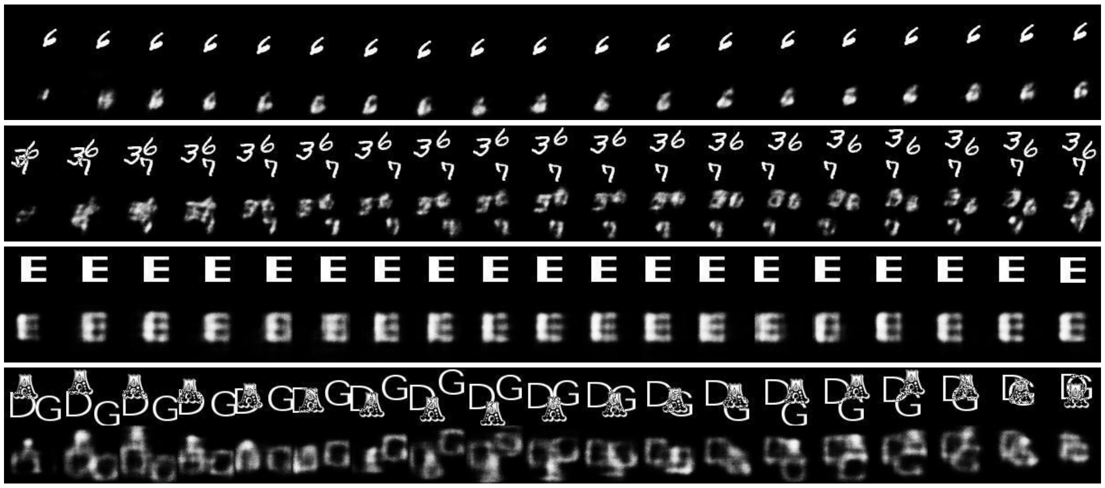

Fig. 1. Out-of-domain test runs. Frame by frame predictions (bottom row) of the P-TNCN compared to ground truth (top row) frames on test sequences of one and three moving digits. Recall that the P-TNCN was only trained on sequences of two moving digits.

sented in the main paper by adding in an intermediate phases of one-shot adaptation/learning before evaluating zero-shot capabilities. This setting could be considered a variation of the zero-shot setting where some aspect of continual generative modeling (of test-sets) has been mixed in.

Specifically, we first take a model already trained on Bouncing MNIST, after many epochs (dispensing with the extreme, one-shot-only constraint of the continual learning experiment) and dynamically adapt it to the NotMNIST test-set (*one-shot learning*). After this one-shot adaptation phase, we evaluate each model's *zero-shot* adaptivity on Fashion MNIST test-set.

During one-shot adaptation to NotMNIST, we update parameters using stochastic gradient descent with a fixed step size of 0.01 and report its test-then-train generalization in Table II. Finally, after one-shot adaptation to NotMNIST, we report the the zero-shot performance, measured in terms of squared error, on Fashion MNIST. The results of this experiment, shown in in Table II, further demonstrate that the P-TNCN's zero-shot adaptive abilities do not degrade when further learning is permitted. Specifically, even when processing the completely orthogonal Fashion MNIST test-set after one-shot adapting to NotMNIST, the P-TNCN outperforms the LSTM models trained with BPTT, SAB, and RTRL.

Using Discrete-Valued Activation Functions: One rather interesting property of the P-TNCN is that it does not require knowledge of the first derivative of its pointwise activation functions. This means that one could employ a much wider variety of nonlinearities than one could not use in a standard, differentiable RNN that would require BPTT/SAB/RTRL/UORO-based approaches. As a result, the P-TNCN is more general and thus uniquely suited for a wider variety of applications. We demonstrate that one can indeed train a P-TNCN with nondifferentiable activity, we construct a simple toy problem as our experiment. We fit a 2-layer (20 units in each) model composed of the non-differentiable signum activations to a stream of data generated by the "noisy cosine function", or $\mathbf{x}_t = cos(t) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, 0.02)$ (we simulate $100K$ discrete steps, where when $k = k + 1$, $t = t + \Delta t$, $\Delta t = 0.05$). We obtain a prequential squared error, or pSE (a test-then-train metric inspired by the online learning literature [3], [4]), of $pSE = 0.0163$ whereas the same exact P-TNCN (but one with differentiable tanh activations) yields a $pSE = 0.0169$. The gold standard model (the cosine function) obtains a $pSE = 0.0006$ and a random/non-adapted baseline model yields $pSE = 1.1057$ (the closer to the gold standard, the better).