DON'T JUDGE A BOOK BY ITS COVER - ON THE DYNAMICS OF RECURRENT NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

To be effective in sequential data processing, Recurrent Neural Networks (RNNs) are required to perform data processing as well as to keep track of past events by creating *memories*. Consequently RNNs are harder to train than their not recurrent counterparts. In this paper, we investigate the representation of memories formed in trained RNN internal state under various training protocols. It is not clear whether there is a trade-off between learning discriminative task, learning actions and learning to memorize. Having observed the RNN's apparently consistent performance regardless of training protocol, we expected the internal dynamics to be similar as well. Instead we were surprised to discover substantial differences, leading to differences in the ability to generalize for unforeseen tasks or conditions. In an attempt to understand these differences we proposed a method for tracking the formation of memories along the course of training, and indeed results on memory shaping process were obtained.

1 INTRODUCTION

Recurrent Neural Networks (RNN) are the key tool currently used in machine learning when dealing with sequential data (Sutskever et al., 2014), and in many tasks requiring a memory of past events (Oh et al., 2016). This is due to the dependency of the network on its past states, and through them on the entire input history. This ability comes with a cost - RNNs are known to be hard to train (Pascanu et al., 2013a). This difficulty is commonly associated with the vanishing gradient that appears when trying to propagate errors over long times (Hochreiter, 1998), and has been addressed by introducing various architectures such as Long Short-Term Memory (LSTM) cells (Hochreiter & Schmidhuber, 1997), Gated Recurrent Units (Chung et al., 2015).

RNNs, however, are not only required to memorize the input, but are also trained to process it. This processing, which could perhaps be easier for a feed-forward network, utilizes the same connections that are trained for the memory component of the task. These two components could interfere with each other, hampering task performance. We show a simple task in which naively training an RNN fails, prompting us to look for alternative training strategies.

In the spirit of curriculum learning (Bengio et al., 2009; Cirik et al., 2016) we start training the network on one aspect of the task; either first learning to extract features or firstly learn to memorize, and when successful move to learn the second aspect as well. Both approaches work, leading to comparable performance on the full task. This is consistent with previous reports (Jozefowicz et al., 2015) that show curriculum learning can accelerate training, albeit reaching fairly similar results after convergence of training (Cirik et al., 2016).

A more careful analysis, however, reveals that the resulting networks differ in their extrapolation abilities and reflect their training histories. Understanding the underlying cause for these differences requires some form of reverse engineering. This can be done by focusing on individual recurrent units (Karpathy et al., 2015; Oh et al., 2016), or by analyzing global network properties. We opt for the latter, analyzing the RNN hidden states as a discrete-time dynamical system. In this framework, memories are expected to be associated with fixed points of the dynamics (Hopfield, 1982; Sussillo, 2014; Barak, 2017; Amit, 1989; Manjunath & Jaeger, 2013). We extend tools used in continuous-time systems in neuroscience (Sussillo & Barak, 2013) to discrete time gated units used in machine learning. Our analysis reveals that although some memories are represented by mentioned robust, fixed-point attractors, and thus persist for long periods of time, other are encoded in more transient

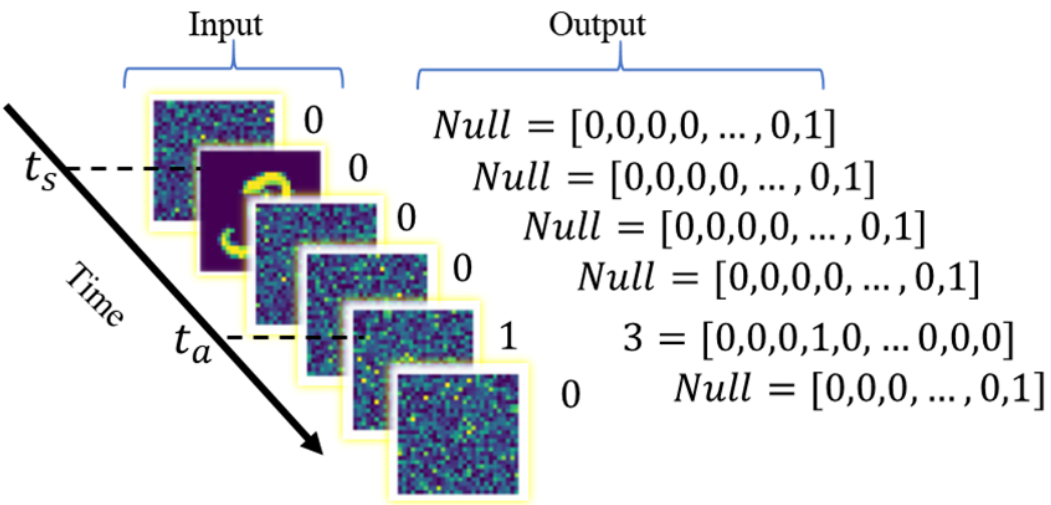

Figure 1: The task. The network is presented an MNIST digit amidst noisy images, and has to report its label at a later time, as requested by a separate input ($0, 1$ to the right of images). Output should be null at all times except the reporting time. The precise times t_a, t_s vary from trial to trial.

dynamics, such has slow points and fixed points with narrow basins-of-attraction, and are effective only for short time periods; with effectiveness of transient dynamics and basin-of-attraction of fixed-points attractors varying with training protocol.

To gain insight on how the different training protocols give rise to different dynamical objects, we followed their formation across training. We discovered that when training by first learning to process and then learning to memorize the RNN the initial internal representation of classes does not feature slow dynamics and it is only during the subsequent training that stable or quasi stable representation emerges. Conversely, second procedure leads to numerous glitches along the course of training, compromising memory sustainability.

2 Task definition

Inspired by real-world applications of Recurrent Neural Networks (Oh et al., 2016), we designed a task where (figure 1) the RNN has to combine stimulus processing and memorization. The network is presented with a series of noisy images, among which appears a single MNIST digit at time t_s. At a later time point (t_a), the network receives a response trigger in a separate input channel, prompting it to output the label of the digit. At all other times, the network should input a null label.

The stimulus and reporting times are chosen randomly each trial from a uniform distribution on $[1, T_{max}]$ subject to the constraint $t_a - t_s > 4$. The total stimulation time $T_{max} = 20$, and the network was requested to distinguish between $|V| = 10$ different classes of MNIST (LeCun et al., 2010).

Each pixel of the noise mask was sampled from a Gaussian distribution with mean and variance matching its counterpart at the image corpus $\epsilon \sim N(\mu_M, \sigma_M^2)$.

As an unforeseen flavor of this task, we required our system to report the classification twice, by triggering it accordingly.

The motivation behind this task is three-fold. First, as explained, this task is comparable to real-world scenarios where RNNs are used for, combining the need for both stimulus memorization and feature extraction. Second, the task lends itself to parametric variations, allowing to compare both different training protocols and generalization abilities.

Third, desiring to understand the dynamical nature of memorization in discrete Gated-RNNs, the delay between stimulus and response trigger allows for evolution of RNN hidden-state (HS), which

can be reliably analyzed and using well known methods from dynamical systems (Sussillo & Barak, 2013), which we modify to our discrete setting.

3 MODEL

The network consists of a single recurrent layer of $N = 200$ Gated Recurrent Units and an output layer of $|V| + 1 = 11$ neurons, $|V| = 10$ neuron for the different classes, and an additional neuron for the null indicator. The input layer has $n + 1$ neurons, where n is the number of pixels in the image and an extra binary input channel for the response trigger $X_r(t)$ defined by:

$$X_r(t) = \begin{cases} 1, & \text{if } t = t_a. \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The network was trained using the 'Adam' optimizer (Kingma & Ba, 2014) with a soft-max cross-entropy loss function with a increased loss on reporting at $t = t_a$ in proportion to T_{max}. No regularization was needed as no over-fitting occurred (Tables 5) presumably thanks to noisy input. Full description of each protocol, including schedules and other hyper-parameters is given in Appendix.

4 TRAINING PROTOCOL: TWO TYPES OF CURRICULA

We found that training failed when using straightforward SG optimization on the full task. The network converged to a state where it consistently reports 'null' without regarding neither the output trigger nor the images it has received as inputs. This suboptimal behavior did not improve upon further training. On the other hand, we observed that simpler versions of the task are learnable. If the maximal delay between stimulus and reporting time was short or when we introduced only a limited number of different digits, the network was able to perform the task. This led us to try two different protocols of curriculum learning in order teach the Neural Net the full required task:

1. Vocabulary curriculum (VoCu) - here we started from two classes $'V' = \{c_1, c_2\}$ and then increased the vocabulary gradually until reaching the full class capacity. This protocol is similar to the original concept of (Bengio et al., 2009) except the fact that in our vocabulary all the classes occur with the same frequency, and the selected order of curricula is in fact arbitrary.

2. Delay curriculum (DeCu) - starting from short delays between stimulus and reporting time ($T_{max} = 6$), we progressively extended it toward the desired values. Implicitly mentioned in (Hochreiter, 1998), this regime is expected to mitigate the vanishing gradient problem, at least during initial phase of training.

5 EXTRAPOLATION ABILITY DEPENDS ON TRAINING PROTOCOL

We found that, in good accordance with existing literature (Bengio et al., 2009; Jozefowicz et al., 2015) results for nominal test-set were fairly indifferent to both architecture and training protocol. Once we evaluated the ability of each setting to extrapolate to more challenging task settings, however, similarity ends and differences emerge.

We tested three different extrapolation settings, that are motivated by an hypothesis on the underlying mechanism of the network. Following previous studies (Sussillo & Barak, 2013; Mante et al., 2013), we expected the dynamics of the hidden state to converge to stable fixed points following the stimulus, where each point corresponds to a different digit. Such dynamics would enable the network to maintain the memory for the varying delay periods it encounters in different trials. At the response trigger, the network is expected to report the stimulus class, and report *Null* posterior to trigger, which we hypothesized it does so by return to the state prior to trigger and therefore expected to be accurate when a second trigger is introduced.

We evaluated these hypotheses by three different extensions to the task: extending the delay, adding noise, and requiring two responses within a single trial. First, we observed how each setting performs when the delay between stimulus and response trigger is extended further beyond $T_{max} = 20$. Expecting $|V| = 10$ robust fixed-point attractors to form, we conjectured that retrieval accuracy should not be affected by the growing delay.

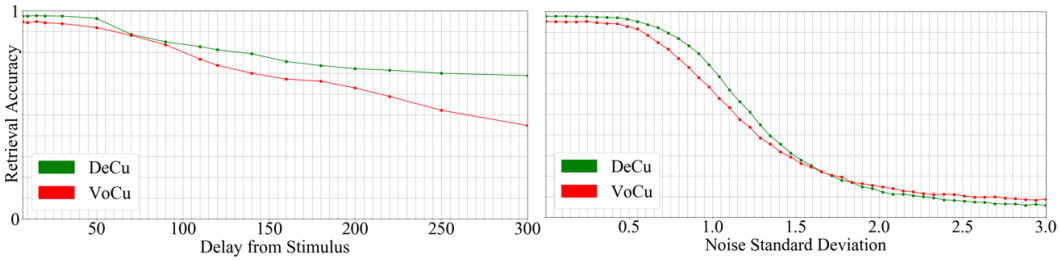

Figure 2: Retrieval accuracy when increasing the delay between stimulus and response trigger beyond $T_{max} = 20$ (*left*) and for increasing noise standard deviation beyond σ_M (*right*). Despite similar performance initially, the ability to generalize for greater delays and greater noises than trained for varies with protocol with VoCu being far less capable compared to DeCu

Table 1: Second Retrieval

	DeCu	VoCu
First trigger at $t = 12$	97.6%	94.75%
Second trigger at $t = 12$	97.22%	79.91%
Difference	0.38 %	**14.81**%

Table 2: Retrieval accuracy when there are two response triggers (with a delay of $t = 6$ and $t = 12$ time-steps from stimulus) instead of a single one. Ability to extrapolate for when there is more than a single response trigger varies greatly with curriculum, with Delayed Curriculum being virtually non-effected by the first trigger and Vocabulary Curriculum with being devastated by it.

Experiments revealed that this was not the case - performance deteriorated with increasing delay, and did so in a setting-dependent manner (figure 2. This deterioration implies that not every memorized digit corresponds to a stable fixed point attractor. The dynamical objects that do represent memories in our networks are analyzed in the following section.

Similarly, we can evaluate reporting accuracy when a the noise mask has a higher variance than used for training. Once more, this serves to evaluate the hypothetical fixed points, and figure 2 shows that not all curricula are equally able to mitigate input-driven noise which indicates to differences in basin-of-attraction of the attractors formed under each setting.

To evaluate the final part of the hypothesis, we introduce a second response trigger, and evaluate the retrieval accuracy then. After a response trigger, the network is to output a null indicator, as it did prior to the trigger. It can do so in two ways, the first is return to the state is was in prior to the trigger, thus respond correctly to the second trigger as well. The second option is to move to a state that has no recollection of the stimulus but has a read-out of *Null*, effective in the single trigger, trained on task, but unable to extrapolate to a two trigger setting.

Table 1 shows that functionality diminishes significantly under a two trigger setting when training with VoCu, and is barely affected when training with DeCu. Further indication of weaker dynamics resulted in training by successive addition

All the extrapolation experiments show that despite comparable results superficially, the governing dynamics and representation of the networks vary with training protocol, resulting in different extrapolation performance.

6 Dynamics of Hidden Representation

The contradiction between the naive fixed point hypothesis and the extrapolation results led us to analyze the dynamical properties of networks under the two different training protocols. Prior work provides a wide spectrum of alternative hypotheses with one end represented by systems without

stable states Manjunath & Jaeger (2013); Maass (2011) and on the other extreme, memory networks Sukhbaatar et al. (2015) that memorize *everything* and later use only the relevant memories while ignoring all the rest.

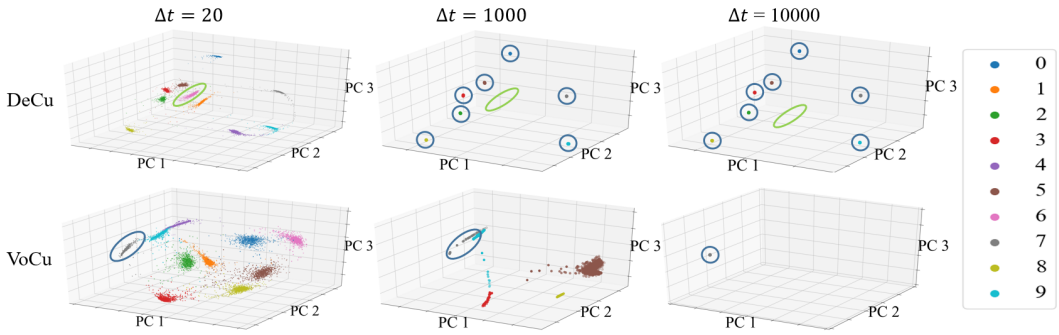

Figure 3: HS projected on leading principal components $\Delta t = 20, 10^3, 10^4$ time-steps after stimulus is introduced and for nomimal input noise strength, color codded by RNN prediction. Top: Decu, Bottom: Vocu. Circled in *blue* are robust fixed points slow points as they had velocities considerably smaller compared to other points (see figure 4) and did not vanish for large delays, in *green* are fixed points with small basins-of-attraction, as they too had small velocities, but were only able to endure for long delays under weak noises (figure 6 in appendix A). Weaker contractive properties around VoCu memory regions are reflected in wider spread of samples all the way to $\Delta t = 10^3$. Spread of samples along lines is in accordance with findings about memory formation in VoCu that is discussed in sequel. Visualization reported here was performed under a quenched noise setting, where the same series of noisy images was used for the entire experiment.

The relevant phase space of this dynamical system is the activity of hidden state neurons. We thus begin by visually inspecting (in the first 3 PCA components) the activity of the network for the maximal training delay, $\Delta t = 20$. The left panels of Figure 3 show that different trials of each digit are well separated into $|V|$ regions with a one to one correspondence to data classes. This is apparently consistent with a fixed point hypothesis, but the extrapolation results indicate otherwise. Indeed, following these trajectories for a longer delay of $\Delta t = 1000$ shows that some regions converge into what appears to be fixed points, while others vanish (right panels). This is true even when a smaller noise amplitude is used (middle panels), although here more regions survive. These figures also clearly show the difference between the two protocols. While both achieve a good separation with the nominal delay (left), it is already apparent that VoCu leads to much larger clusters, possibly indicating a weaker attraction.

To verify the existence or absence of fixed points hinted by the above visualization, we extend an algorithm developed for continuous time vanilla RNN to our setting Sussillo & Barak (2013). Briefly, fixed points (stable or unstable) are global minima of the (scalar) speed of the dynamics. It is therefore possible to use gradient descent to locate such fixed points. In our case, the quantity to minimize is given by:

$$
\begin{aligned}
z &= \sigma(W_z I + U_z h + b_z) \\
r &= \sigma(W_r I + U_r h + b_r) \\
h^* &= (1 - z) \circ h + z \circ \tanh(W_h I + U_h (r \circ h) + b_h) \\
&\implies S(h, I) = ||h^* - h||_2^2
\end{aligned}
$$

The initial conditions for this gradient descent were obtained by running the network with the mean delay value $\Delta t = 12$, and using the class average. The external input I during gradient descent was the average of the noise images, thus effectively making our system Time-Invariant where such points and their stability are well defined. We verified that using different fixed external inputs did not qualitatively alter the results (not shown). Once the gradient descent converged, we measured the distances between all initial points and all final points. We found that the algorithm always converged to a point within the same class. We also verified that the readout from the point of convergence matched the class label.

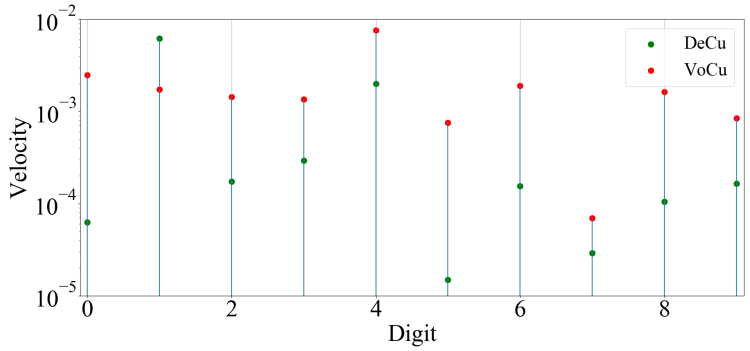

Figure 4: Using discrete Black Box Analysis, we measured the speed of each velocity minima of each class for both VoCu (*red*) and DeCu (*green*). Under VoCu, velocity minima had significantly higher velocities compared to their DeCu counterparts, providing with explanation of results shown in section 5. Further more, although Black Box was performed from a small delay from stimulus ($\Delta = 12$), it can accurately predict which classes will persist and which will vanish 3 for long delays according to velocity of located minima.

We thus saw that every class is associated with a local minima of the speed under all curricula. But, as reported in (Sussillo & Barak, 2013), trained networks utilize both fixed points (zero speed) and slow points (low non-zero speed). Furthermore, even fixed points can be stable or unstable, and even locally stable fixed points can have small or large basins of attraction. We thus analyzed the minima obtained for all classes under both training protocols. For each minimum we noted the value of the speed, the stability obtained by linearizing (relevant also for slow points (Sussillo & Barak, 2013)), and the tolerance to increasing levels of noise (a measure of the basin of attraction).

Using Black Box Analysis, we found that for DeCu, the minima of classes $\{0, 2, 3, 5, 7, 8, 9\}$ were robust fixed-points as they sustained large noise levels over long periods of time, while the minimum of $\{6\}$ was a fixed-point with a small basin-of-attraction, and the other classes $\{1, 4\}$ were slow points and had were measured with speeds larger by more than an order of magnitude compared to the other minima (figure 4). In comparison, under VoCu, only the the minima of $\{7\}$ was a robust fixed-point, evident from its slowness (figure 4) and from PC visualization (3), while all other minima were transient slow points.

The results of this analysis are plotted as circles of different colors in Figure 3, where it can be seen that this procedure (done on $\Delta t = 12$) can predict the long term behavior and is in agreement with section 5 as VoCu has significantly fewer robust fixed-points, and shows faster dynamics compared to DeCu.

We saw that the two training protocols lead to a different representation of the stimulus memory by the network, and to different dynamical objects. How does training give rise to these differences? To answer this question, we followed the local minima of the velocity backwards in training time to learn how they arise and change throughout training. This was done by beginning with the classes at the end of training, and using the local minima of one training step as the initial condition for gradient descent on the network defined by the previous training step (Algorithm 1).

Figure 5 compares the outcome of Algorithm 1 applied on VoCu and DeCu. It can be seen that while in DeCu setting slow point velocity goes down systematically along the course of the training, in the VoCu case introduction of new classes, immediately compromises the slowness of the old ones leading resulting to less contractive dynamics in their vecinity.

While primarily concerned with learning of memory rather than learning to classify, we found it appropriate to evaluate a protocol in which *only* memorizing is being trained at the first phase and it is only later that any classification task introduced. Described in Appendix B results of PTMT training share some properties from VoCu and others from DeCu. Importantly, success of PMTP rules out the possibility that the visual features rather than the memorization capabilities are enabling the VoCu protocol success.

Algorithm 1

procedure BACKTRACK SLOW POINTS FORMATION
  $I \leftarrow$ *DC of Noise*
  $h_0 \leftarrow$ *Slow point at end of training*
  $M \leftarrow$ *Number of Training Steps*
  For $(i = M - 1; i > 0; i - -)$:
    $D \leftarrow$ Neural Network at step i
    $h \leftarrow$ Black Box Analysis on dynamics D starting from h_0 with noise I
    $h_0 \leftarrow h$
  return h_0 as Origin

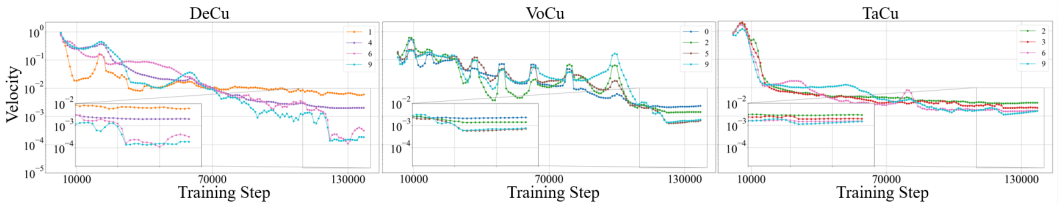

Figure 5: Performing 1 algorithm, and measuring the speed of each minima landed on at each step for the 3 protocols. We observe a monotonically decreasing velocity under DeCu indicating to increasing slowness and attractiveness of slow regions over training. Under VoCu there are spikes in velocity at training steps where new classes are introduced, leading to faster dynamics at the end of training. With PMTP, the velocity of origins is comparable to the other two, randomly initialized curricula, suggesting that post-transfer slow regions were not based of pre-transfer ones, but rather on reusing of generally slow dynamics created under pre-training.

7 DISCUSSION

Training RNN a difficult task, (Pascanu et al., 2013b), and as a consequence has to rely on pre-cooked dynamics. This makes paradigms of transfer learning and curriculum learning particularly relevant for RNN domain (Venugopalan et al., 2016; Ramachandran et al., 2016; Gulcehre et al., 2015), and the specific aspect of extrapolation to unforeseen situations, represented e.g. by second retrieval challenge in our model (table 1) becomes crucial (Schwarting et al., 2018).

Here we showed that while locally optimal solutions obtained by different training protocols perform similarly under nominal conditions, it should not be assumed that they would also perform similarly in unforeseen settings. This issue is a particular case of the more general problem of solution equivalence in gradient based optimization: On the one hand theoretical and numerical evidence does exist for training outcome's indifference to protocol details. Specifically to our case of RNN, it is shown in (Cirik et al., 2016) that, at least for language modeling tasks, performance does not heavily depend on training protocol. On the other hand, such an indifference is far from being fully established. In particular, stochastic gradient optimization suffers from known drawbacks (Dauphin et al., 2014; Martens & Sutskever, 2011) and might prove dependent on initialization (Sutskever et al., 2013) (and in particular pre-training). In this work we were able find considerable differences in dynamical landscapes which in turn translate to idiosyncrasies in functionality between curricula and architectures. By inspecting the slow points in proximity of each class' hidden representation, we establish link between locally slow regions and global memorization.

Ultimately, we followed the formation of the aforementioned fixed-points and other slow dynamics during training process, and were able to qualitatively explain why different training protocols do result in different dynamics. Future applications of this method, summarized in Algorithm 1, include study of success and failure to create attractors, prevent catastrophic forgetting (Beer & Barak, 2018) as well as suitability for domain adaptation and various generalization properties in task performing networks.

REFERENCES

Daniel J. Amit. Modeling brain function: the world of attractor neural networks, 1st edition. 1989.

Omri Barak. Recurrent neural networks as versatile tools of neuroscience research. *Current opinion in neurobiology*, 46:1–6, 2017.

Chen Beer and Omri Barak. Dynamics of dynamics: following the formation of a line attractor. *arXiv preprint arXiv:1805.09603*, 2018.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 41–48, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-516-1. doi: 10.1145/1553374.1553380. URL http://doi.acm.org/10.1145/1553374.1553380.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In *International Conference on Machine Learning*, pp. 2067–2075, 2015.

Volkan Cirik, Eduard H. Hovy, and Louis-Philippe Morency. Visualizing and understanding curriculum learning for long short-term memory networks. *CoRR*, abs/1611.06204, 2016. URL http://arxiv.org/abs/1611.06204.

Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, pp. 2933–2941, 2014.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*, 2015.

Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02): 107–116, 1998.

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

John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 2342–2350. JMLR.org, 2015. URL http://dl.acm.org/citation.cfm?id=3045118.3045367.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

Wolfgang Maass. Liquid state machines: motivation, theory, and applications. In *Computability in context: computation and logic in the real world*, pp. 275–296. World Scientific, 2011.

G. Manjunath and H. Jaeger. Echo state property linked to an input: Exploring a fundamental characteristic of recurrent neural networks. *Neural Computation*, 25(3):671–696, 2013. doi: 10.1162/NECO_a_00411. URL https://doi.org/10.1162/NECO_a_00411. PMID: 23272918.

Valerio Mante, David Sussillo, Krishna V. Shenoy, and William T. Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. In *Nature*, 2013.

James Martens and Ilya Sutskever. Learning recurrent neural networks with hessian-free optimization. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 1033–1040. Citeseer, 2011.

Junhyuk Oh, Valliappa Chockalingam, Satinder P. Singh, and Honglak Lee. Control of memory, active perception, and action in minecraft. *CoRR*, abs/1605.09128, 2016. URL `http://arxiv.org/abs/1605.09128`.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pp. 1310–1318, 2013a.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pp. 1310–1318, 2013b.

Prajit Ramachandran, Peter J Liu, and Quoc V Le. Unsupervised pretraining for sequence to sequence learning. *arXiv preprint arXiv:1611.02683*, 2016.

Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:187–210, 2018.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pp. 2440–2448, 2015.

David Sussillo. Neural circuits as computational dynamical systems. *Current opinion in neurobiology*, 25:156–63, 2014.

David Sussillo and Omri Barak. Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Computation*, 25(3):626–649, 2013. doi: 10.1162/NECO_a_00409. URL `https://doi.org/10.1162/NECO_a_00409`. PMID: 23272922.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147, 2013.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Subhashini Venugopalan, Lisa Anne Hendricks, Raymond Mooney, and Kate Saenko. Improving lstm-based video description with linguistic knowledge mined from text. *arXiv preprint arXiv:1604.01729*, 2016.

A VISUALIZATION FOR SMALL NOISES

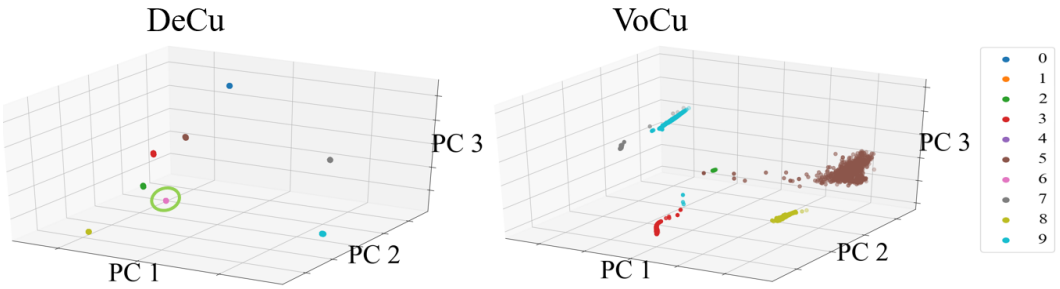

Figure 6: HS projected on leading principal components $\Delta t = 10^3$ time-steps after stimulus is introduced and for input noise strength with $\frac{1}{3}$ of nominal magnitude, color codded by RNN prediction. Left: Decu, Right: Vocu. In *green* are small basin-of-attraction fixed points mentioned in figure 6

B PURE MEMORY TRANSFER PROTOCOL

In Pure Memory Transfer Protocol (PMTP) we initially train the RNN on the same task, but on a simple stimuli; instead of digits one-hot encoded digits with noise mask generated from uniform distribution bounded between $[0, 1]$. After training is successful on said stimuli, we randomly re-initialize read-out parameters and introduce the original stimuli (MNIST images) with nominal noise mask.

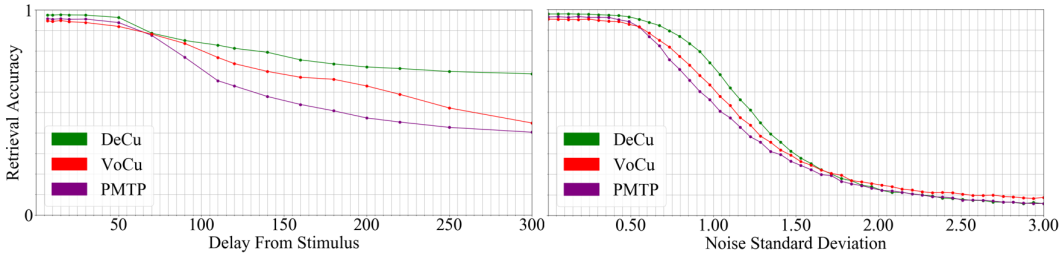

Figure 7: Extrapolation capabilities to vast delays and harsher noise of a RNN trained using PTMT. Reaching similar on-the-surface results, PTMP is less capable of handling harsh input-driven noise compared to the other curricula, and shows a more rapid degradation in retrieval accuracy as delay increases, however, unlike VoCu, PMTP is unaffected by second retrieval.

Table 3: Second Trigger Retrieval

	DeCu	VoCu	TaCu
First trigger at $t = 12$	97.6%	94.75%	95.47%
Second trigger at $t = 12$	97.22%	79.91%	95.17%
Difference	0.38 %	**14.81**%	0.3%

Table 4: Retrieval accuracy when there are two response triggers (with a delay of $t = 6$ and $t = 12$ time-steps from stimulus) instead of a single one. Despite showing similar results to VoCu when extrapolating to long delays and high magnitude noise, PMTP behaves successfully on second trigger.

Training with PMTP yielded comparable results in both test and train accuracies compared to VoCu and DeCu, further illustrating point in (Bengio et al., 2009; Cirik et al., 2016), however, as we saw in

section 5, similar interpolation abilities don't translate to similar extrapolation, and PTMP diverges from VaCu as it is function in a two trigger setting and also diverges from DeCu in ability to endure harsh noise and long delays.

C TRAINING HYPER-PARAMETERS AND RESULTS

When training for MNIST recognition, we used the 'Adam' Kingma & Ba (2014) with learning rate of $\alpha = 10^-5$ for the first $12 \cdot 10^4$ training step, then decreased the learning rate by an order of magnitude to $\alpha = 10^{-6}$ and trained for $2 \cdot 10^4$ more training steps. When pre-training for Pure Memory Transfer Protocol for cyclic one-hot digit recognition, we performed $1 \cdot 10^4$ training steps with a learning rate of $\alpha = 10^{-3}$.

Training under VoCu, new classes were added at steps: $\{8, 16, 26, 36, 48, 62, 78, 98\} \cdot 10^3$. With DeCu, after intially training for $2 \cdot 10^4$ training steps with a maximal delay of $T_{max} = 6$ between stimulus and response trigger, maximal delay was increased by increments of 2 every $2 \cdot 10^4$ training steps.

Table 5: MNIST - Gated Recurrent Unit

		DeCu	VoCu	TaCu	Naive
Training Set	Null	100%	100%	100%	100%
	Digits	97.5%	95.22%	96.24%	0%
Test Set	Null	100%	100%	100%	100%
	Digits	97.44%	94.84%	96.2%	0%

D BLACK BOX AND BACKTRACKING ALGORITHM HYPER-PARAMETERS

For locating velocity minima using discrete Black Box, we run a quenched noise simulation with $\frac{1}{3}$ of nominal magnitude for 500 instances of each class for a delay of $\Delta = 12$ and performed gradient decent to minimize the velocity, and clipping HS values between $[-1, 1]$. We optimized using 'Adam' with learning rate of $\alpha = 10^{-3}$ for $4 \cdot 10^4$ gradient decent steps, with $1 \cdot 10^4$ more steps with $\alpha = 10^{-4}$

With BackTracking algorithm, $6 \cdot 10^3$ gradient decent steps were performed at each Black Box iteration, and by varying only the Hidden-to-Hidden weight and biases during BackTrack as to keep system input independent and dynamical properties well defined.