Dynamics of dynamics: following the formation of a line attractor

Chen Beer
Faculty of Electrical Engineering
Technion, Israel Institute of Technology
chenco@campus.technion.ac.il

Omri Barak
Faculty of Medicine
Technion, Israel Institute of Technology
omri.barak@gmail.com

Abstract

Trained recurrent neural networks are commonly used both in neuroscience and in machine learning. Recently, several attempts were made to understand the dynamics of such trained networks. Understanding how these dynamics arise through learning, however, is an uncharted territory. Here, we study this process in a simple setting – learning to memorize analog values, leading to the formation of a line attractor. We show that problems associated with sequential learning of different tasks also occur in this setting. Specifically, catastrophic forgetting dramatically hinders a local learning rule from converging. In contrast, we show that the FORCE algorithm avoids this issue, and analyze the underlying causes. Using these insights, we suggest steps towards a more biologically plausible version of FORCE, as well as an approach to design optimal training protocols.

1 Introduction

In recent years, trained recurrent neural networks were used as a model for several experimental tasks in Neuroscience [1, 2, 3, 4, 5]. While some progress was made on understanding the dynamics of networks following training [1, 6], the formation of these dynamics throughout learning remains a puzzle. We study this process in a simple setting - the formation of a line attractor associated with learning to memorize analog values.

Recurrent neural networks were both designed [7, 8, 9, 10] and trained [11, 12, 13] to memorize analog values. In designed networks, memory is implemented by a continuum of fixed points - a line attractor. The stability properties of this line attractor were studied to understand where they arise from, how they can be imrpoved [14], and how they link to experimental observations [15]. For trained networks, if the memory duration is long and variable enough [11], a line attractor also emerges, and can be detected by reverse engineering methods [6, 1, 16].

Learning to memorize analog values entails two interacting dynamical processes. Within a single trial, the system is expected to converge to a stable fixed point, representing the memorized value. On a longer timescale, learning requires integrating sequentially presented trials into a coherent behavior. We show that the interaction between these slow and fast processes can lead to training success or failure, depending on the learning rule used. The slower process - sequential learning from many trials - was mostly studied in the context of learning multiple tasks. In this case, the danger of catastrophic forgetting was stressed, and several suggestions were made on how to alleviate it [17, 18, 19].

Here, we use reverse engineering to follow the formation of a line attractor throughout training. We show that, depending on the learning rule, the sequential nature of learning can cause catastrophic interference within a single task. By studying the cross-talk between the slow and fast processes, we expose the underlying causes for training success and failure.

2 The framework

2.1 Network architecture

We use a rate based recurrent neural network (RNN), defined by the equations (Figure 1A):

$$\tau \dot{x_i} = -x_i + \sum_{j=1}^{N} J_{ij} r_j + B_i u + w_i^{FB} z \tag{1}$$

where x_i is the input to neuron $i = 1, \ldots, N$, J is a random connectivity matrix with elements sampled iid from $\mathcal{N}(0, g^2/N)$, $r_i = \phi(x_i)$ is the firing rate with $\phi(x) = tanh(x)$, and the external inputs u are fed through weights B_i. The current state of the network is defined by $x \in R^N$ and the output of the network $z = w_{out}^T r$ is fed back through weights w^{FB}. We use $N = 1000, g = 1.2, \tau = 100$msec.

We consider the case where only the output weights are modified during training, which facilitates our analysis. Because of the feedback connection, training still modifies network dynamics, and thus allows successful training of the task. We also verified that the main effects are qualitatively similar when training the internal connections as well.

2.2 The task

We present transient stimuli to the network, and train it to output their value for a long time after stimulus removal (Figure 1B). The stimulus amplitudes are uniformly sampled from $[1, 5]$, and are presented for 500 msec, followed by a delay period uniformly sampled from $[0.5, 6]$ sec.

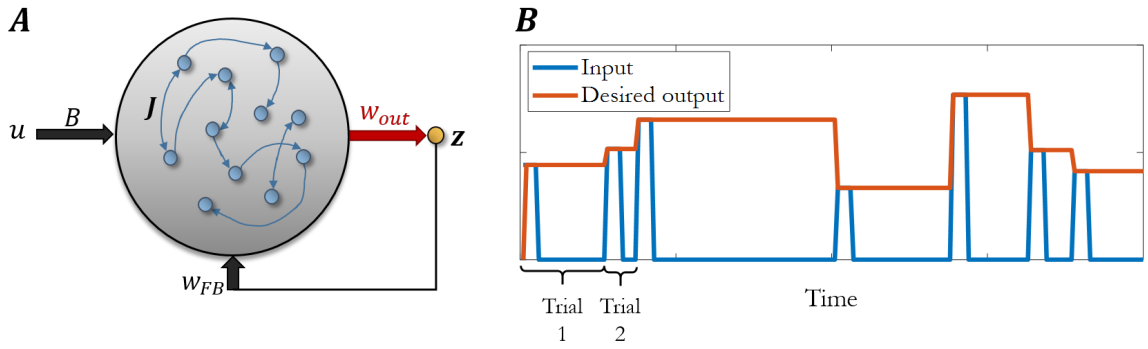

Figure 1: **(A)** Network architecture. Only the output weights (red) are modified during training. **(B)** The task: at each trial a stimulus with amplitude in the range $[1, 5]$ is presented for 500 msec (blue). The delay period between different stimuli is uniformly distributed between 0.5 and 6 seconds. The desired behavior is to output this value for the entire trial duration (orange).

2.3 Learning rules

We inspect the learning process using two different training methods:

FORCE algorithm [20], based on recursive least squares (RLS), with the following update rule:

$$w(t) = w(t - \Delta t) - e^-(t) P(t) r(t) \tag{2}$$

$$e^-(t) = w^T(t - \Delta t) r(t) - f(t) \tag{3}$$

where $P(t)$ is a running estimate of the inverse correlation matrix of the network rates r plus a regularization term, and $f(t)$ is the target output.

Although it is widely used, FORCE is not biologically plausible because the modification of a given synapse depends on information from the entire neural population. These locality considerations led the authors of [20] to suggest a local learning rule:

Least mean squares (LMS), in which the modification rule for the output weights is

$$w(t) = w(t - \Delta t) - \eta(t)e^{-}(t)r(t) \tag{4}$$

where $e^{-}(t)$ is defined as in FORCE, and $\eta(t)$ is a time-varying learning rate: $d\eta/dt = \eta(-\eta + |e|^{\gamma})$. A further step towards biological realism was made by the introduction of a reward based learning rule [21]. This rule was described as leading to convergence on the same set of tasks as FORCE, albeit with a convergence rate that is slower than FORCE and similar to LMS.

3 Results

We trained networks for 300 trials using either FORCE or LMS. Figure 2 shows that while FORCE training resulted in almost perfect performance, this was not the case for LMS. Closer inspection of the LMS output during testing indicates that the network converged to a fixed point corresponding to the last trained stimulus value. Because in [20, 21] LMS was shown to require roughly 10 times more trials to converge than FORCE, we continued training for a further 10^4 trials, but to no avail. The supplementary information shows a variant of this task in which we were able to get slightly better performance after 45000 trials, but still very far from the performance of FORCE after 300 trials.

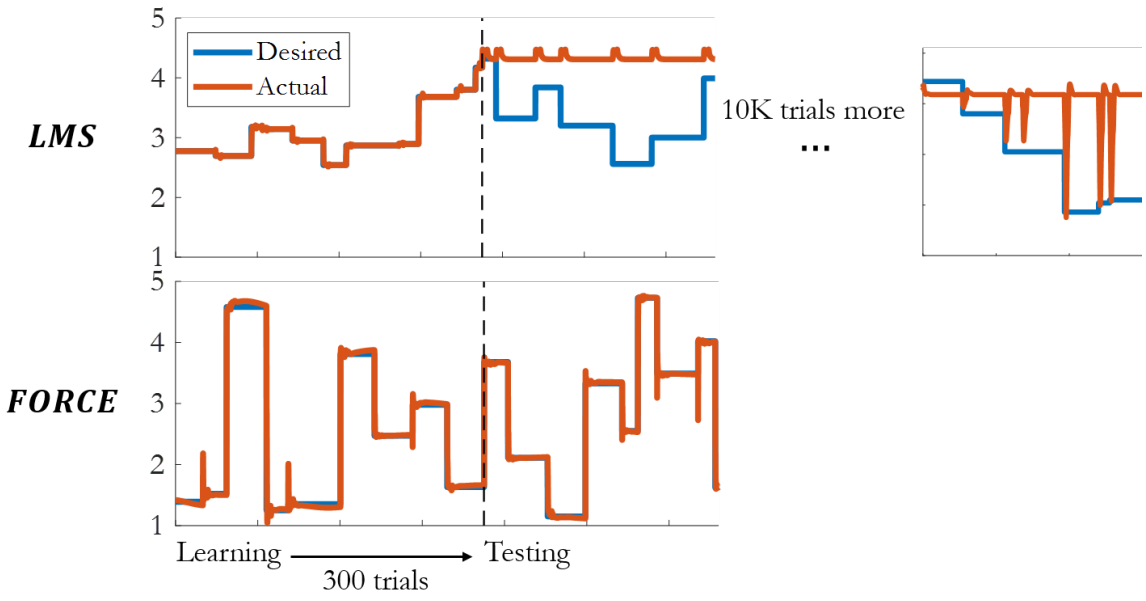

Figure 2: Network output using LMS (top) and FORCE (bottom) training. During training, both algorithms lead to the desired output. Testing, however, reveals that the LMS-trained network converges to the output value of the last training stimulus. The rightmost plot shows a test session following an additional 10^4 training trials. The dashed line denotes the beginning of testing. In FORCE $\alpha = 10$, in LMS $\eta_0 = 10^{-5}$ and $\gamma = 2$.

To gain insight into the underlying reasons for this behavior, we reverse engineer the network to uncover it's dynamics. Following [6], we define a scalar function $q(x) = \frac{1}{2}|\dot{x}|^2$ which is zero for fixed points. In a successfully trained network, we expect to find an approximate line attractor - a one dimensional manifold of very low q values. Each point along this line represents a memorized value. Because we only train the output weights, the location of this line attractor is pre-determined in the following manner. For a given output value $z(t) \equiv A$, the open loop system (in which the target output A is fed back to the network) converges to a unique stable state $\bar{x}$ which is given by: $\bar{x} = J\phi(\bar{x}) + w^{FB}A$. This happens for A that are large enough to suppress chaos [22], which is true for the values used in this work. Figure 3A shows $\bar{x}$ for the entire output range $[1, 5]$ in PCA space.

Training the output weights can determine whether these $\bar{x}$ are indeed fixed points ($w_{out}^{T}\bar{r} = A$), and whether they are stable. Figure 3B shows the value of $q(\bar{x})$ for different values of z. The behavior of the FORCE- and LMS- trained networks is captured by these graphs. The q values of FORCE are very low for the entire z

range, indicating an approximate line attractor. Conversely, LMS leads to a single fixed point around $z = 4.2$, matching the behavior seen in Figure 2.

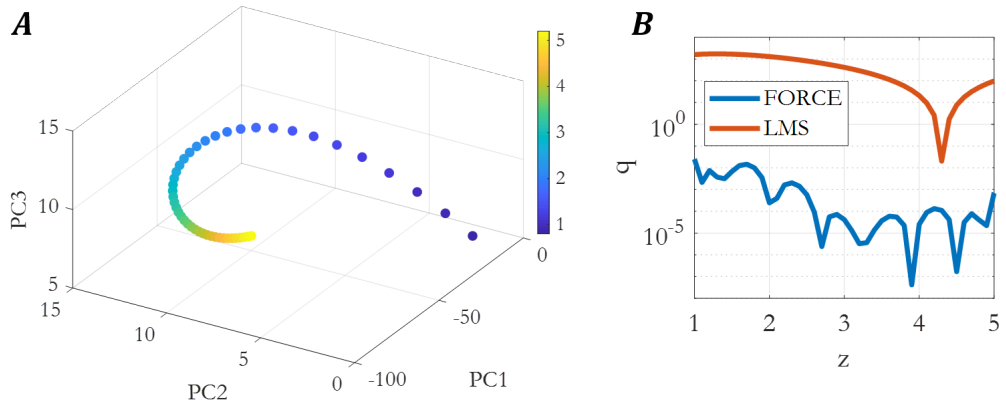

Figure 3: The line attractor. **(A)** The locations $\bar{x}$ for the entire range $z \in [1, 5]$ in PCA space. **(B)** q values along the expected line attractor following 300 trials of training in either FORCE (blue) or LMS (red).

What enables FORCE-trained networks to have many fixed points, while LMS only leads to a single one? Our results suggest that both algorithms will lead to a fixed point after the first trial, but only FORCE will maintain this fixed point after the second trial. We investigate this hypothesis in a slightly reduced setup.

3.1 Reduced setup

In order to disambiguate the pure formation and maintenance of fixed points from input pulse effects and transitions between fixed points, we investigate a reduced setup of the task. Instead of using external inputs to shift between the various output levels, we use the fact that the internal state $\bar{x}$ is precisely defined by output values. We thus use these states as initial conditions for each trial, and eliminate external input.

Training the network on the first trial leads to a fixed point using either algorithm. But the FORCE algorithm, through its use of recursive least squares, maintains a matrix P that implicitly contains information on past trials. P is a running estimate for the inverse of the correlation matrix of the network rates r plus a regularization term:

$$P = \left(\int r(t) r^T(t) dt + \alpha I \right)^{-1} \tag{5}$$

The eigenvectors of P are thus approximately the principal components of the activity, but since P retains the inverse of the activity history, the eigenvalues of these principle components will be very low. The update of synaptic weights in FORCE is proportional to P (Equation 3), and thus protects the directions most visited (e.g. fixed points) from further change.

In our task (even in the reduced setup), each trial begins with a short transient until the network converges into the fixed point corresponding to the current target value. Assuming that the time spent at the fixed point $\bar{r}$ is significantly longer than the transient period, we can write the following approximation for P following the first trial:

$$\tilde{P} = (T \bar{r} \bar{r}^T + \alpha I)^{-1} \tag{6}$$

where T is trial duration. The eigenvalues of P (which we call the nominal case) and $\tilde{P}$ after a single trial are shown in Figure 4. The lowest eigenvalue in both matrices is $(\alpha + T \bar{r}^T \bar{r})^{-1}$, which corresponds to the eigenvector $\bar{r}$. The transient history is reflected in the eigenvalues of the nominal P, and not in $\tilde{P}$.

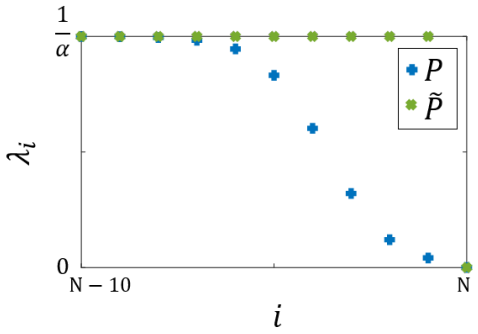

Figure 4: The last 10 eigenvalues of P and $\tilde{P}$ following FORCE training of the first trial. The remaining $N - 10$ equal $1/\alpha$ in both cases. Network and training parameters: $N = 1000$, $g = 1.2$, $\alpha = 1$ and target value $A_1 = 1$.

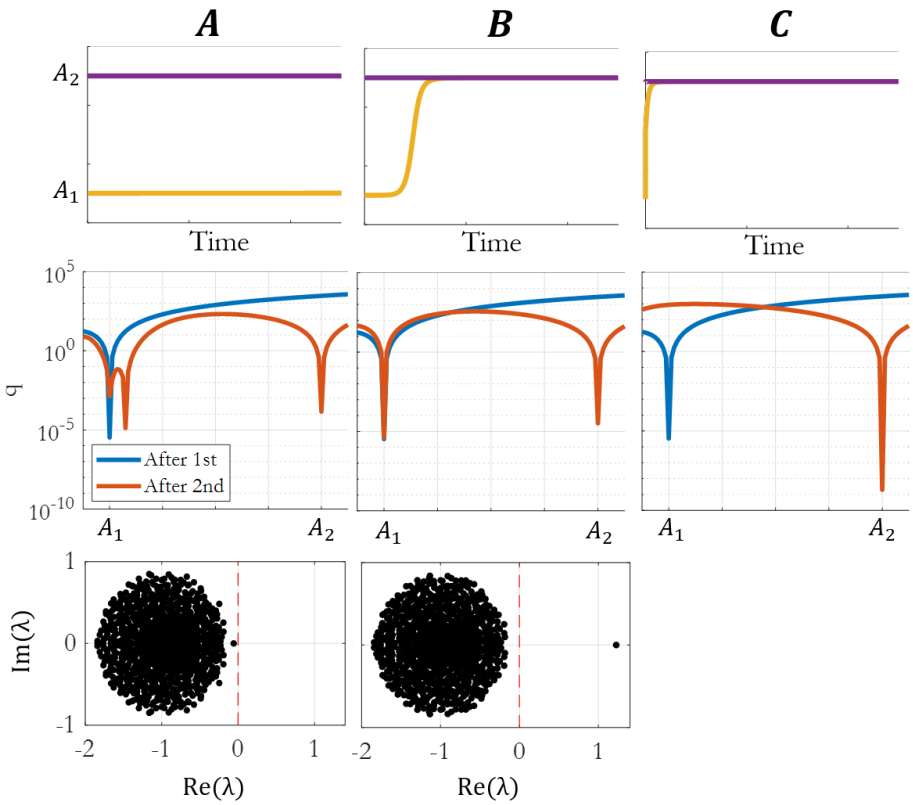

Figure 5: The first two trials. The second trial is learned via **(A)** nominal FORCE, leading to two stable fixed points **(B)** FORCE using $\tilde{P}$, leading to one stable and one unstable fixed point or **(C)** LMS leading to a single fixed point. Top row: Output during testing, after two training trials. Middle row: q values along the expected final line attractor after the first (blue) and second (orange) trials. Bottom row: the spectrum of the network linearized around r_1 after the second trial (not shown in **C** because there is no fixed point). $\alpha = 1$, $A_1 = 1$, $A_2 = 5$.

Now that we understand the information stored in P, we can examine what happens during the second trial (Figure 5). We denote the targets of the two trials A_1 and A_2, with the corresponding putative fixed points r_1 and r_2. We perform the training in both LMS and three variants of FORCE: nominal FORCE with P, FORCE using $\tilde{P}$ and FORCE using a full reset $P = (\alpha I)^{-1}$.

Training with nominal FORCE results in two stable fixed points (Figure 5A), indicating that r_1 was preserved. Both LMS and a full reset of P after the first trial cause the network to forget the first value, and as a result, only one fixed point corresponding to the last value A_2 remains (Figure 5C shows LMS which is nearly identical to the full reset case). Using $\tilde{P}$, which contains information on the fixed point r_1, but not the transients leading to it, results in an intermediate behavior. The resulting network contains two fixed points, but only the one corresponding to the last value is stable (figure 5B).

These results demonstrate how information regarding the existence of fixed points and their stability is stored in P during the learning process. It also illustrates how the transient history is crucial for stability.

Using these insights we now turn to two applications. First, we suggest a step towards a more biologically plausible version of FORCE that still learns the task. Second, we show how choosing the order of stimulus presentation can affect the resulting network dynamics.

3.2 Modified FORCE algorithm

The FORCE algorithm suffers from two plausibility issues. First, the algorithm requires storing N^2 values in order to update N synapses. Second, the update rule is highly non-local. To alleviate these problems, two modifications to FORCE were suggested. The first, LMS, within the original paper [20], and a later suggestion using reinforcement learning [21] which was claimed to be equivalent to LMS. In both cases, it was hypothesized that it is possible to learn the same tasks, albeit with slower convergence. Our results, however, indicate that there might be a qualitative difference that will prevent these local rules from converging to full performance because of the destruction of old fixed points. We also simulated the reward based rule, and observed a behavior similar to LMS.

Here, we show how the first of the above issues can be addressed. Instead of maintaining N^2 values representing the full history $r(t)$, we only keep the last few (three) fixed points, $O(N)$ values:

$$P_\mu = (\sum_{k=\mu-3}^{\mu-1} \bar{r}_k \cdot \bar{r}_k^T + \alpha I)^{-1} \tag{7}$$

where μ indexes the trials. Figure 6 compares the performance of this rule (yellow) to other variants of FORCE. Panel A shows that it reaches low error values, while panel B illustrates the meaning of this error by plotting the drift for various initial conditions.

This result is apparently at odds with Figure 5B, in which erasing the transient information from P destabilized the old fixed point. We believe that the reason this learning rule eventually converges is that there is high correlation between the various $\bar{x}$ values, and thus protecting a few fixed points can have a greater effect.

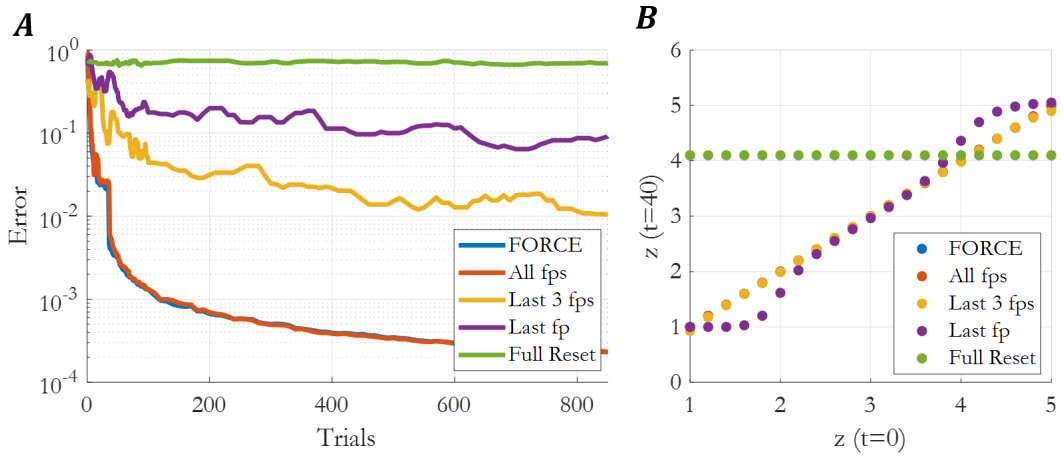

Figure 6: Modified FORCE algorithm **(A)** Average cross validation error as a function of trials. Keeping the last 3 fixed points (yellow) is compared to other variants of FORCE. **(B)** Drift during testing for the different variants. The blue, orange and yellow markers are overlapping. All results are averages over 5 realizations. standard errors were at least an order of magnitude smaller than the mean.

3.3 Choosing the order of trials

In both the training of animals (shaping, [23]) and of artificial networks (curriculum learning, [24]), people have observed that the order of training can have an effect on the final performance. Our results indicate an interplay between the stability of learned fixed points and the process of learning. We hypothesize that this feature can be used to design optimal training sequences.

As a proof of concept, we compare learning when the stimuli are presented continuously from 1 to 5 (forward, Figure 7A blue) or from 5 to 1 (backward, Figure 7A red). We see that during this sweep (first 40 trials), the error decreases in a very different profile for the two directions. Strikingly, after this single sweep we transition to a random presentation of stimuli (dashed lines) and find that this initial sweep has a long lasting effect on performance. The supplemental material shows further tests of this effect, including better extrapolation to untrained stimuli.

Why is the forward sweep better? The different stimuli are associated with different intrinsic stabilities. To see that we compute the output weights solution using ordinary least squares on the entire line of $\bar{x}$. This guarantees a line of fixed points, but does not guarantee stability. Figure 7B shows the two leading eigenvalues around these fixed points. As expected, one of them is a zero eigenvalue, while the other varies across the stimulus range. Because the lower values are unstable, the backward sweep is forced to make large changes to the readout weight towards the end of training, and these hurt the quality of the line attractor in the high values.

How can one identify such stability properties in a real life setting? The insets in Figure 7B show the test results after only two trials in the forward or backward direction. The qualitative differences indicate that it is possible to assess such stability properties, and possibly modify the order of trials accordingly.

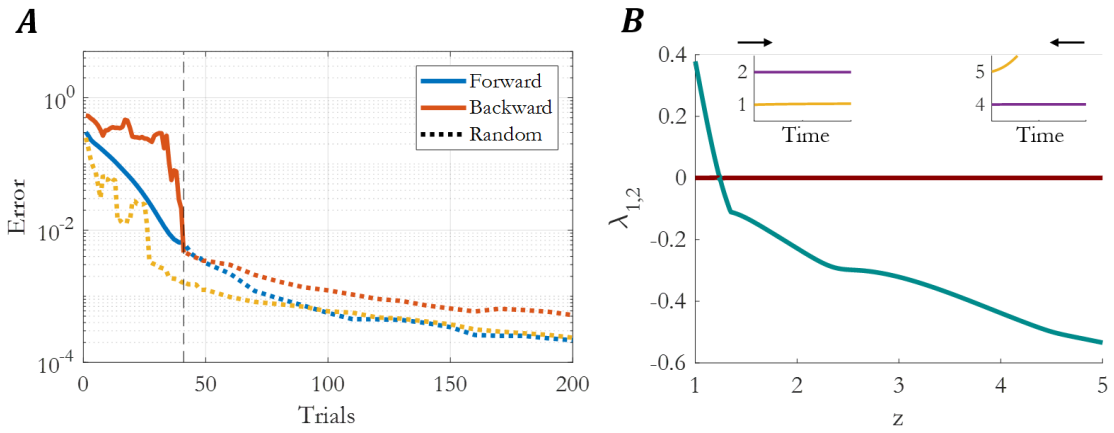

Figure 7: Order of training affects performance. **(A)** Average cross validation error as a function of the learning process. The solid lines indicate an ordered sweep from 1 to 5 (forward, blue), or from 5 to 1 (backward, red). Following this initial sweep, the remaining stimuli were drawn randomly. The yellow line is a fully random sequence. **(B)** Intrinsic stability as shown by leading eigenvalues of the Least Squares solution. Insets show test results after two trials that start in the forward or backward direction.

4 Discussion

In this work we followed the formation of a line attractor through the online training of a recurrent neural network. By doing so we were able to demonstrate that problems usually associated with sequential learning of multiple tasks are also relevant in this setting. We showed that FORCE, an RLS based algorithm, avoids catastrophic forgetting by implicitly avoiding updates to previously visited directions. By dissecting the contributions of the various parts of a trial to this implicit memory, we could understand how the maintenance of fixed points and their stability leads to learning.

We focused on the FORCE algorithm, due to its simplicity and usage within neuroscience inspired tasks [25, 12]. This algorithm, however, is not biologically plausible. Both LMS and a reward based rule were suggested as slower, but effective plausible versions of FORCE. Our results indicate that this might not be the case in more complex tasks such as the one studied here. Furthermore, we suggest a different route towards

plausible learning - by maintaining a small number of recent states in the P matrix we are able to converge with $O(N)$ storage, but still using non-local updates. It might be possible to integrate our suggestion with other recent works [18, 19] to obtain a quickly converging local version of FORCE.

Although we stress the sequential nature of learning, the fact that this is a single task leads to important distinctions from the sequential learning of uncorrelated tasks. In our case, the rate vectors corresponding to different points along the line attractor are highly correlated. This is the reason that it is possible to achieve stability even when removing the transient contributions to P.

Shaping, or choosing an optimal sequence of training examples, is a topic of importance in both animal training and machine learning research [23, 24]. Our results indicate that when dynamical systems, such as recurrent neural networks, are considered, stability can play a major role in this aspect. In the future, it might be possible to probe the system to understand its stability properties, and use this information to guide the ordering of the following stimuli.

Understanding the dynamics of trained recurrent neural networks is an important and difficult challenge [4, 26, 5]. Understanding the formation of these dynamics is an even more formidable task. Here, we take a first step towards this goal and show that it sheds light on the nature of learning rules, and can provide clues on how to improve learning. The next steps towards this goal are expected to provide even more.

Acknowledgments

We thank Daniel Soudry for helpful discussions. This work was supported by the Israeli Science Foundation (grant no. 346/16).

References

[1] Valerio Mante, David Sussillo, Krishna V. Shenoy, and William T. Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, 503(7474):78–84, November 2013. ISSN 0028-0836, 1476-4687. doi: 10.1038/nature12742. URL http://www.nature.com/articles/nature12742.

[2] Abigail A. Russo, Sean R. Bittner, Sean M. Perkins, Jeffrey S. Seely, Brian M. London, Antonio H. Lara, Andrew Miri, Najja J. Marshall, Adam Kohn, Thomas M. Jessell, Laurence F. Abbott, John P. Cunningham, and Mark M. Churchland. Motor Cortex Embeds Muscle-like Commands in an Untangled Population Response. *Neuron*, 97(4):953–966.e8, February 2018. ISSN 0896-6273. doi: 10.1016/j.neuron.2018.01.004. URL http://www.sciencedirect.com/science/article/pii/S0896627318300072.

[3] Jing Wang, Devika Narain, Eghbal A. Hosseini, and Mehrdad Jazayeri. Flexible timing by temporal scaling of cortical responses. *Nature Neuroscience*, 21(1):102–110, January 2018. ISSN 1546-1726. doi: 10.1038/s41593-017-0028-6. URL https://www.nature.com/articles/s41593-017-0028-6.

[4] David Sussillo. Neural circuits as computational dynamical systems. *Current opinion in neurobiology*, 25:156–163, 2014. URL http://www.sciencedirect.com/science/article/pii/S0959438814000166.

[5] Omri Barak. Recurrent neural networks as versatile tools of neuroscience research. *Current Opinion in Neurobiology*, 46:1–6, October 2017. ISSN 0959-4388. doi: 10.1016/j.conb.2017.06.003. URL http://www.sciencedirect.com/science/article/pii/S0959438817300429.

[6] 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. URL http://www.mitpressjournals.org/doi/full/10.1162/NECO_a_00409.

[7] C. K Machens, R. Romo, and C. D Brody. Flexible control of mutual inhibition: a neural model of two-interval discrimination. *Science*, 307(5712):1121–1124, 2005.

[8] P. Miller, C. D. Brody, R. Romo, and X. J. Wang. A recurrent network model of somatosensory parametric working memory in the prefrontal cortex. *Cerebral Cortex*, 13(11):1208–1218, 2003.

[9] R. Ben-Yishai, R. L. Bar-Or, and H. Sompolinsky. Theory of orientation tuning in visual cortex. *Proceedings of the National Academy of Sciences*, 92(9):3844–3848, 1995. URL http://www.pnas.org/content/92/9/3844.short.

[10] R. Singh and C. Eliasmith. Higher-dimensional neurons explain the tuning and dynamics of working memory cells. *Journal of Neuroscience*, 26(14):3667–3678, 2006.

[11] Emin Orhan and Wei Ji Ma. A Diverse Range of Factors Affect the Nature of Neural Representations Underlying Short-Term Memory. *bioRxiv*, page 244707, 2018.

[12] Omri Barak, David Sussillo, Ranulfo Romo, Misha Tsodyks, and L.F. Abbott. From fixed points to chaos: Three models of delayed discrimination. *Progress in Neurobiology*, 103:214–222, April 2013. ISSN 0301-0082. doi: 10.1016/j.pneurobio.2013.02.002. URL http://www.sciencedirect.com/science/article/pii/S0301008213000129.

[13] H. Francis Song, Guangyu R. Yang, and Xiao-Jing Wang. Reward-based training of recurrent neural networks for cognitive and value-based tasks. *eLife*, 6:e21492, 2017. URL https://elifesciences.org/content/6/e21492v2.

[14] Vladimir Itskov, David Hansel, and Misha Tsodyks. Short-term facilitation may stabilize parametric working memory trace. *Frontiers in computational neuroscience*, 5, 2011. URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3199447/.

[15] Klaus Wimmer, Duane Q. Nykamp, Christos Constantinidis, and Albert Compte. Bump attractor dynamics in prefrontal cortex explains behavioral precision in spatial working memory. *Nature Neuroscience*, 17(3):431–439, March 2014. ISSN 1097-6256. doi: 10.1038/nn.3645. URL http://www.nature.com/neuro/journal/v17/n3/full/nn.3645.html.

[16] Garrett E. Katz and James A. Reggia. Using Directional Fibers to Locate Fixed Points of Recurrent Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2017. URL http://ieeexplore.ieee.org/abstract/document/8016349/.

[17] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, and Agnieszka Grabska-Barwinska. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[18] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995, 2017.

[19] Chen Zeno, Itay Golan, Elad Hoffer, and Daniel Soudry. Bayesian Gradient Descent: Online Variational Bayes Learning with Increased Robustness to Catastrophic Forgetting and Weight Pruning. *arXiv:1803.10123 [cs, stat]*, March 2018. URL http://arxiv.org/abs/1803.10123. arXiv: 1803.10123.

[20] David Sussillo and L. F. Abbott. Generating Coherent Patterns of Activity from Chaotic Neural Networks. *Neuron*, 63(4):544–557, August 2009. ISSN 0896-6273. doi: 10.1016/j.neuron.2009.07.018. URL http://www.sciencedirect.com/science/article/pii/S0896627309005479.

[21] Gregor M. Hoerzer, Robert Legenstein, and Wolfgang Maass. Emergence of complex computational structures from chaotic neural networks through reward-modulated Hebbian learning. *Cerebral Cortex*, 2012. URL http://cercor.oxfordjournals.org/content/early/2012/11/09/cercor.bhs348.short.

[22] Kanaka Rajan, L. F. Abbott, and Haim Sompolinsky. Stimulus-dependent suppression of chaos in recurrent neural networks. *Physical Review E*, 82(1):011903, July 2010. doi: 10.1103/PhysRevE.82.011903. URL http://link.aps.org/doi/10.1103/PhysRevE.82.011903.

[23] Burrhus F. Skinner. Reinforcement today. *American Psychologist*, 13(3):94, 1958.

[24] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[25] Rodrigo Laje and Dean V. Buonomano. Robust timing and motor patterns by taming chaos in recurrent neural networks. *Nature Neuroscience*, 16(7):925–933, July 2013. ISSN 1097-6256. doi: 10.1038/nn.3405. URL http://www.nature.com/neuro/journal/v16/n7/full/nn.3405.html.

[26] Peiran Gao and Surya Ganguli. On simplicity and complexity in the brave new world of large-scale neuroscience. *Current Opinion in Neurobiology*, 32:148–155, June 2015. ISSN 0959-4388. doi: 10.1016/j.conb.2015.04.003. URL http://www.sciencedirect.com/science/article/pii/S0959438815000768.