Learning rate collapse prevents training recurrent neural networks at scale

Editors: List of editors' names

Abstract

Recurrent neural networks (RNNs) are central to modeling neural computation in systems neuroscience, yet the principles that enable their stable and efficient training at large scales remain poorly understood. Seminal work in machine learning predicts that the effective learning rate should shrink with the size of feedforward networks, and practical heuristics such as learning rate schedulers or width-dependent rescaling have been proposed to stabilize training. Here we demonstrate that an analogous phenomenon, which we term learning rate collapse, poses a concrete barrier for large RNNs in neuroscience-inspired short-term memory tasks. We show that the maximum trainable learning rate decreases as a power law with neuron number, forcing large networks to converge only at critically slow learning rates. While parameter rescaling with the inverse of network size mitigates instability, larger networks still learn substantially slower, indicating that collapse is a structural limitation rather than a trivial parametrization artifact. These optimization limits are further compounded by severe memory demands, which together make training large RNNs both unstable and computationally costly. As a proof of principle, we design a learning process that enforces a low-dimensional geometry in RNN representations, which reduces memory costs and mitigates learning rate collapse. These results situate learning rate collapse within a broader lineage of scaling analyses and establish it as a fundamental obstacle for training large RNNs, with potential solutions likely to come from future work that incorporates careful consideration of symmetry and geometry in neural representations.

Keywords: Recurrent neural networks, computational neuroscience

1. Introduction

In both biological and artificial networks, neurons are often studied under pre-defined in-put-output relationships. Yet, despite such constraints, their collective dynamics can acquire a wide repertoire of complex dynamics and computational capabilities that are not explicitly induced in individual neurons (Gardner et al., 2022; Churchland et al., 2012; Sabatini and Kaufman, 2024; Kurtkaya et al., 2025). Over the past decades, the concept of population coding has become central in formalizing this connection: information is represented not by single units but by the collective dynamics of neural populations (Yuste, 2015; Baeg et al., 2003; Meyers et al., 2008; Chaudhuri et al., 2019; Averbeck et al., 2006). A canonical example is provided by grid cells, where spatial position is encoded through the joint activity of many neurons forming a hexagonal lattice, and this code resides on a low-dimensional toroidal manifold, offering a compelling demonstration of structured population codes (Gardner et al., 2022; Langdon et al., 2023).

To investigate how collective representations support diverse computations and to study neural circuitry, computational neuroscientists often train biologically interpretable recurrent neural networks (RNNs) as virtual twins or on neuroscience-inspired tasks (Rajan et al., 2016; Perich and Rajan, 2020; Dinc et al., 2023; Kurtkaya et al., 2025; Cohen et al., 2020).

For instance, Kurtkaya et al. (2025) trained leaky firing-rate RNNs on the delayed cuediscrimination task to examine short-term memory dynamics. To improve interpretability and uncover low-dimensional dynamics, researchers have also developed methods that draw on concepts from dynamical systems, topology, and geometry (Sussillo and Barak, 2013; Dinc et al., 2025; Mastrogiuseppe and Ostojic, 2018; Valente et al., 2022; Acosta et al., 2023, 2024). A notable example is Sussillo and Barak (2013), who applied dynamical systems theory to analyze RNN phase space, identifying attractors such as fixed points by directly optimizing hidden states. More recently, Dinc et al. (2025) introduced latent processing units (LPUs), extracted via linear combinations of neural activities, which provide a tractable way to characterize population-level dynamics. This ongoing line of work continues to strengthen the case for population coding as a unifying principle across biological and artificial networks (Hu et al., 2024; Cavanagh et al., 2018; Spaak et al., 2017; Meyers et al., 2008; Stroud et al., 2024; Brennan and Proekt, 2023; Pandarinath et al., 2018). On the other hand, experimental and computational studies operate at very different scales. Advances in optical recording have enabled neuroscientists to measure the activity of tens of thousands of neurons simultaneously (Ebrahimi et al., 2022; Bruzzone et al., 2021; Ahrens et al., 2013; Stringer et al., 2019). By contrast, most computational studies of RNNs remain limited to networks with only a few hundred neurons (Masse et al., 2019; Yang et al., 2019; Kurtkaya et al., 2025). This discrepancy raises a fundamental question: What prevents artificial models from scaling to the same dimensions as their biological counterparts?

One well-known limitation is the sheer computational cost and memory demand. For an RNN with N neurons, the number of synaptic weights grows as N^2 , making it infeasible to simulate networks approaching the tens of thousands of units observed in biology. Yet computational burden alone does not fully explain the scaling barrier. A more fundamental issue arises from the optimization dynamics: analyses of gradient propagation show that the variance of weight updates grows with network size, requiring the learning rate to scale inversely with neuron number, *i.e.*, proportional to 1/N (Saxe et al., 2013; Gilboa et al., 2019; Yang et al., 2022). Without such adjustments of the learning rate, large networks are known to diverge rapidly during training.

In this work, we demonstrate an analog of this phenomenon, which we term learning rate collapse, in RNNs. To establish this result, we systematically trained RNNs on three wellestablished neuroscience tasks: the Flip-Flop Task (FF), the Delayed Cue-Discrimination Task (DCDT), and the Delayed Matching Task (DMT) (see Fig. 1 and Sec. 2.2). We then quantified scaling laws by identifying the largest learning rates capable of training more than half of the model instances, revealing a clear power-law dependence on network size (see Fig. 2, 3 and Sec. 3.1, 3.2). Beyond learning-rate adjustments, we also investigated parameter rescaling and showed that scaling the recurrent weight matrices mitigates the learning rate collapse, but larger networks still take longer to train at the same learning rates (see Fig. 4 and Sec. 3.3). Importantly, while both strategies prevent divergence, they do so at the cost of significantly slowing learning, with larger networks requiring many more epochs to converge (see Fig. 1, 4). To overcome these limitations, we introduce a proof of principle training paradigm, in which the latent dynamical systems within low-rank RNNs are optimized (see Fig. 5 and Sec. 3.4). Conceptually, this corresponds to training the parameters at the level of population codes rather than the connections between individual neurons, though the former does describe the latter. While this approach is one possible

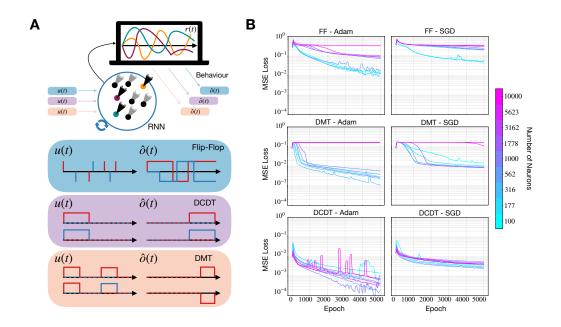


Figure 1: Scaling the learning rate slows down training in all settings. (A) Schematic of the three benchmark tasks: the Flip-Flop task (FF), the Delayed Cue Discrimination Task (DCDT), and the Delayed Matching Task (DMT) (see Sec. 2.2). (B) In total, we trained 3,900 RNNs across tasks, learning rates, optimizers, and seeds. For network sizes greater than 100, we show the best-performing loss curves averaged across seeds. Larger networks consistently converge more slowly, since lower learning rates are required for stability. The learning rates (LR) are reported in Table 1.

method for mitigating the learning-rate collapse while simultaneously reducing memory and computational demands, further work incorporating geometry and symmetry in learned representations may be able to further decrease the number of parameters that need to be trained, thereby enabling training of large-scale RNNs.

2. Methods

2.1. Recurrent neural networks

We employed leaky firing-rate recurrent neural networks (RNNs), which are equivalent to vanilla RNNs, to model population dynamics during short-term memory tasks. The recurrent dynamics were defined as

$$\tau \dot{r}(t) = -r(t) + \phi \left(W r(t) + W^{\text{in}} u(t) \right), \tag{1}$$

where $r(t) \in \mathbb{R}^N$ denotes the activity of N recurrent units (firing rates) at time $t, W \in \mathbb{R}^{N \times N}$ is the recurrent weight matrix, $W^{\text{in}} \in \mathbb{R}^{N \times d_{\text{in}}}$ is the input projection from d_{in} -dimensional external inputs u(t). The nonlinear activation function $\phi(\cdot)$ was chosen as the hyperbolic tangent, $\phi(x) = \tanh(x)$, consistent with prior work on dynamical systems analyses of RNNs

Sussillo and Barak (2013); Mastrogiuseppe and Ostojic (2018). The time constant τ was fixed to unity without loss of generality. The network output was given by a linear readout of the recurrent state:

$$\hat{o}(t) = W^{\text{out}}r(t),\tag{2}$$

where $W^{\text{out}} \in \mathbb{R}^{d_{\text{out}} \times N}$ maps the recurrent activity into a d_{out} -dimensional output space.

Training was performed by minimizing the mean squared error (MSE) between predicted and target outputs:

$$\mathcal{L} = \frac{1}{T} \sum_{t=1}^{T} \|\hat{o}(t) - o^*(t)\|^2,$$
(3)

where $o^*(t)$ denotes the task-specific target output and T is the trial length. Optimization was carried out with the Adam or SGD optimizer, depending on the experiment, across a range of learning rates and random seeds. Further details can be found in the Appendix.

2.2. Task details

In this work, we focus on three commonly studied short-term memory tasks from systems neuroscience literature Masse et al. (2019); Yang et al. (2019); Sussillo and Barak (2013).

The Delayed Cue Discrimination Task (DCDT) is a widely used paradigm for probing short-term memory in both biological organisms and artificial neural networks. Each trial consists of three phases: an input phase $T_{\rm in}$, during which a brief cue is presented; a delay phase $T_{\rm delay}$, where no external input is provided and the RNN must internally maintain the cue; and a response phase $T_{\rm resp}$, in which the RNN is required to produce an output based on the earlier cue. This structure makes DCDT a canonical framework for testing the ability to store and retrieve information across a silent interval. Formally, DCDT can be expressed as:

$$u(t) = \begin{cases} (0,1) \text{ or } (1,0), & \text{if } t \in T_{\text{in}}, \\ (0,0), & \text{otherwise,} \end{cases} \qquad o^*(t) = \begin{cases} u(T_{\text{in}}), & \text{if } t \in T_{\text{resp}}, \\ (0,0), & \text{otherwise.} \end{cases}$$
(4)

where u(t) denotes the input at time t and $o^*(t)$ denotes the ground-truth output at time t. The Delayed Matching Task (DMT) presents two cues T_{in_1} and T_{in_2} separated by a delay interval T_{delay} , and the RNN must determine whether they are identical or different in response T_{resp} . Unlike simple DCDT, DMT requires both the maintenance of the first cue during the delay and its integration with the second cue, making it a more demanding test of memory mechanisms. Formally, DMT can be expressed as:

$$u(t) = \begin{cases} (0,1) \text{ or } (1,0), & \text{if } t \in \{T_{\text{in}_1}, T_{\text{in}_2}\}, \\ (0,0), & \text{otherwise,} \end{cases}$$

$$o^*(t) = \begin{cases} +1, & \text{if } t \in T_{\text{resp}} \text{ and } u(T_{\text{in}_1}) = u(T_{\text{in}_2}), \\ -1, & \text{if } t \in T_{\text{resp}} \text{ and } u(T_{\text{in}_1}) \neq u(T_{\text{in}_2}), \\ (0,0), & \text{otherwise.} \end{cases}$$
(5)

where u(t) denotes the input at time t, and $o^*(t)$ denotes the ground-truth output at time t (+1 for match, -1 for non-match, and 0 otherwise).

Proceedings Track

Fraction of seeds passing accuracy threshold — Adam only (FF / DMT / DCDT)

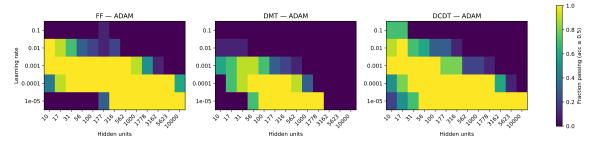


Figure 2: Decreasing the learning rate is necessary for training larger networks. We trained RNNs with 13 different sizes (10^1 to 10^4 , evenly sampled on a log scale), 5 learning rates (10^{-1} to 10^{-5} , log-spaced), and 10 random seeds on three neuroscience tasks. After training, models were deemed successful if they achieved an accuracy greater than 0.5. We then plot the fraction of successful models across all tasks. The results reveal a clear negative scaling law: larger networks require proportionally smaller learning rates (consistent with 1/N behavior) to train successfully. For details see Table 2

The Flip-Flop (FF) task has no explicit delay component. At each timestep, a cue may appear with some probability, either repeating the previous value or switching to the opposite (+1 or -1). The RNN must preserve its current state when the cue is unchanged and flip its output when the cue switches. This makes FF a test of continual state maintenance under unpredictable inputs, rather than storage across fixed delay intervals.

3. Results

3.1. Learning rate collapse prevents training large RNNs to perform short-term memory tasks

We first evaluated whether larger recurrent networks could be trained to perform standard benchmarks of short-term memory: the Flip-Flop task (FF), the Delayed Cue Discrimination Task (DCDT), and the Delayed Matching Task (DMT) (Fig. 1A, Sec. 2.2). These tasks probe different forms of short-term memory, ranging from continual state maintenance (FF) to delayed-response and cue-matching paradigms (DCDT, DMT). Together, they provide a diverse testbed for assessing the stability of training dynamics in recurrent networks.

We trained RNNs using Adam and SGD across varying sizes and a wide range of learning rates, number of neurons, and seeds, resulting in 3,900 distinct networks (see Table 2). Moreover, Table 1 reports the best-performing learning rates for each setting, while Fig. 1B shows representative training curves averaged over seeds. The results reveal a consistent pattern: larger networks converged more slowly, and in many cases, learning rates that were effective for smaller models caused divergence or unstable dynamics in larger ones. Only by decreasing the learning rate could we restore stable optimization.

Across all three tasks and both optimizers, this dependency on network size was robust: the number of recurrent units strongly determined the range of learning rates under which training remained stable. These findings demonstrate that **learning rate collapse is a**

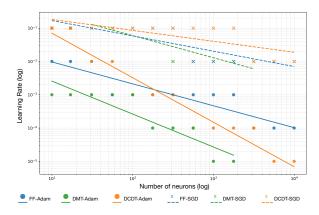


Figure 3: **Learning rate scales as** 1/N. We fit a line to the largest learning rates at each network size that successfully trained at least half of the models. The resulting fits reveal a negative power-law scaling of the form N^{β} with $\beta < 0$. Estimated exponents: FF-Adam, $\beta = 0.66$; DMT-Adam, $\beta = 0.99$; DCDT-Adam, $\beta = 1.34$; FF-SGD, $\beta = 0.46$; DMT-SGD, $\beta = 0.66$; DCDT-SGD, $\beta = 0.33$.

general scaling property of RNN training, and that convergence of large recurrent networks requires systematically smaller learning rates and more training epochs.

3.2. Quantifying the learning rate collapse

To quantify how learning rates scale with network size, we deemed models successful if they achieved an accuracy greater than 0.5. After classifying models, we computed the fraction of successful runs across seeds (Figure 2). From these results, we identified the largest learning rate that trained at least 50% of runs successfully for each network size. We then fit a power-law function to these critical learning rates:

$$\min_{a,\beta} \sum_{i} \left(\log_{10} LR_i - \left(a + \beta \log_{10} N_i \right) \right)^2 \quad \Rightarrow \quad LR = 10^a N^{\beta}, \tag{6}$$

where β is the slope and a is the intercept in log space. Across all tasks and optimizers, the fitted curves exhibit a consistent decaying trend, demonstrating a negative power-law relationship between network size and the maximum stable learning rate (Figure 3). This result confirms the existence of a scaling law: as N increases, the learning rate must diminish approximately as 1/N to ensure stable training. However, this raises an important question: as demonstrated in Figure 1, decreasing the learning rate slows down the training process; can rescaling the network parameters instead of rescaling the learning rate mitigate this slowdown?

3.3. Rescaling the parameters with network size mitigates learning rate collapse, but larger networks still require more epochs to train.

To test whether learning rate collapse could be alleviated by re-parameter, we defined a variant of the RNN in which both recurrent and output weights were rescaled with network size, e.g., $W \to W/N$. This ensured that the effective strength of recurrent interactions

Loss curves (mean across seeds) • Moving average=50 • log-log scale

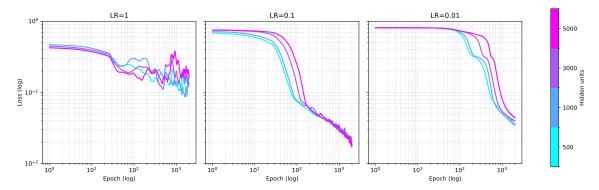


Figure 4: Rescaling the parameters mitigates the learning rate collapse, but larger networks still take longer to train. We defined the recurrent and output weights following a rescaling of the parameters, i.e., $W \to W/N$, and retrained RNNs with varying sizes on the 3-bit flip-flop tasks. All RNNs failed to solve the task at the same learning rate, LR = 1, whereas for smaller learning rates, smaller networks learned the task faster. Each solid line is a mean over 5 runs, RNNs learn to perform the 3-bit flip flop tasks with a high accuracy (> 90%) at around $MSE \approx 0.06$. For details see Table 3

remained invariant as the number of units increased. This formulation maintains the overall structure of the standard RNN update equation while is expected to mitigate the theoretical N^{-1} scaling in the learning rate.

We then trained rescaled RNNs of varying hidden dimensions on the 3-bit flip-flop task, using multiple learning rates and random seeds. Figure 4 shows the resulting loss trajectories across epochs. At high learning rates (LR=1), all models failed to converge regardless of size, consistent with the instability induced by overly large parameter updates. At smaller learning rates, rescaling prevented divergence across network sizes, but convergence dynamics still differed systematically: smaller networks reached low loss values in relatively few epochs, whereas larger networks required substantially more iterations to achieve comparable accuracy. These results show that parameter rescaling stabilizes training but does not remove the scaling disadvantage of large recurrent networks, indicating that learning rate collapse reflects a structural limitation of optimization rather than a trivial consequence of unnormalized weights.

3.4. Learning low-rank representations alleviated the learning difficulties in large RNNs

While re-parameterization stabilized training, it did not remove the slowdown in large RNNs. A more effective strategy is to reformulate the dynamics in a low-rank structure that naturally supports scale-free learning. Specifically, we parametrized low-rank RNNs as

$$\dot{r}(t) = -r(t) + \tanh\left(\frac{1}{N} \sum_{k=1}^{K} m^{(k)} n^{(k)T} r(t) + W^{\text{in}} u(t)\right),\tag{7}$$

Mean over seeds per (hidden size, learning rate) — with Moving Average

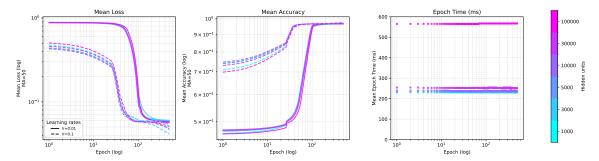


Figure 5: Training low-dimensional latent representations (LPUs) mitigated the learning difficulties in large RNNs. To train low-rank RNNs, we propagated their dimensionally reduced form with a much smaller parameter count (O(N) as opposed to $O(N^2)$). In this case, regardless of the network size, the learning progression curves looked nearly identical for all network sizes (left and middle). More importantly, most networks took about 250ms per epoch (except for the one with 100,000 neurons, which likely required substantial memory), and about O(100) epochs, training in about a minute. For details see Table 4

where $m^{(k)}, n^{(k)} \in \mathbb{R}^N$ are the embedding and encoding vectors for the k-th rank component, and K is the rank of the recurrent connectivity. Prior work has employed this factorization (sometimes referred to as "LINT" (Valente et al., 2022)), but because the full recurrent weight matrix must still be constructed, it retains $O(N^2)$ complexity and remains inefficient for large networks.

To overcome this limitation, we directly reduced the dynamics into the latent system (Dinc et al., 2025). Defining the latent variables as

$$\kappa_k(t) = \frac{1}{N} n^{(k)T} r(t), \tag{8}$$

the dynamics can be written entirely in terms of $\kappa_k(t)$:

$$\tau \dot{\kappa}_k(t) = -\kappa_k(t) + \frac{1}{N} n^{(k)T} \tanh\left(\sum_{\ell=1}^K m^{(\ell)} \kappa_\ell(t) + W^{\text{in}} u(t)\right). \tag{9}$$

This latent formulation evolves only K variables rather than N, reducing the parameter count from $O(N^2)$ to O(NK).

In practice, as shown in Fig. 5, this dimensional reduction enabled efficient training of very large networks while preserving dynamic expressivity and maintaining accuracy. Strikingly, the training curves across network sizes collapsed onto nearly identical trajectories for a given learning rate, and even networks with 100,000 neurons converged within about a minute. Thus, low-rank training not only mitigates learning rate collapse but also achieves genuine scale-free performance in large RNNs, which is crucial for simulating or training virtual twins based on large-scale recordings.

Proceedings Track

4. Discussion and outlook

Our results identify learning rate collapse as a fundamental obstacle to training large recurrent neural networks. Across a range of short-term memory tasks, optimizers, and hyperparameters, we consistently found that the maximum trainable learning rate decreases with network size according to a power law. This scaling law implies that stable training of larger RNNs requires progressively smaller learning rates, which in turn slows convergence and increases computational costs. Importantly, this phenomenon is not a trivial artifact of parametrization but reflects a structural limitation of gradient-based optimization in recurrent systems.

Parameter rescaling partially stabilized training by normalizing the effective recurrent strength across network sizes. However, while this prevented divergence, it did not remove the slowdown: larger networks continued to require substantially more iterations to converge. This observation highlights a key tension: simple rescaling schemes can regularize updates, but they cannot eliminate the inherent inefficiency of training large RNNs at vanishingly small learning rates.

By contrast, our low-rank latent formulation offered a more transformative solution. Projecting the dynamics into a reduced latent space decreased the parameter count from $O(N^2)$ to O(NK), while preserving the expressive capacity of the network. In this setting, learning curves across network sizes collapsed onto one another, and even networks with 100,000 neurons trained efficiently within minutes. These findings show that low-dimensional structure is not merely an interpretability tool but a practical mechanism for scaling RNN training in a size-independent manner.

Taken together, our study situates learning rate collapse within the broader landscape of scaling laws in machine learning. Prior work has established width-dependent rescaling in feedforward networks and mean-field analyses of dynamical isometry in gated architectures; here we demonstrate that recurrent architectures face analogous constraints that manifest as a collapse of the effective learning rate. At the same time, our results resonate with recent neuroscience findings: large biological populations appear to implement computations through low-dimensional manifolds, suggesting that dimensionality reduction is not only a computational convenience but a biological principle.

Looking forward, several directions emerge. First, extending our analysis to gated RNNs and transformer-based recurrent models will clarify whether collapse is universal across sequence-processing architectures. Second, incorporating symmetry and geometry into parameterizations may yield additional scale-free formulations, complementing the low-rank approach. Finally, connecting these optimization limits to experimental data can shed light on how biological networks circumvent collapse, for example by exploiting structured connectivity, local learning rules, or population-level redundancy.

References

- Francisco Acosta, Sophia Sanborn, Khanh Dao Duc, Manu Madhav, and Nina Miolane. Quantifying extrinsic curvature in neural manifolds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 610–619, 2023.
- Francisco Acosta, Fatih Dinc, William Redman, Manu Madhav, David Klindt, and Nina Miolane. Global distortions from local rewards: Neural coding strategies in path-integrating neural systems. *Advances in Neural Information Processing Systems*, 37:100241–100269, 2024.
- Misha B Ahrens, Michael B Orger, Drew N Robson, Jennifer M Li, and Philipp J Keller. Whole-brain functional imaging at cellular resolution using light-sheet microscopy. *Nature methods*, 10(5):413–420, 2013.
- Bruno B Averbeck, Peter E Latham, and Alexandre Pouget. Neural correlations, population coding and computation. *Nature reviews neuroscience*, 7(5):358–366, 2006.
- EH Baeg, YB Kim, K Huh, I Mook-Jung, HT Kim, and MW Jung. Dynamics of population code for working memory in the prefrontal cortex. *Neuron*, 40(1):177–188, 2003.
- Connor Brennan and Alex Proekt. Attractor dynamics with activity-dependent plasticity capture human working memory across time scales. *Communications psychology*, 1(1): 28, 2023.
- Matteo Bruzzone, Enrico Chiarello, Marco Albanesi, Maria Elena Miletto Petrazzini, Aram Megighian, Claudia Lodovichi, and Marco Dal Maschio. Whole brain functional recordings at cellular resolution in zebrafish larvae with 3d scanning multiphoton microscopy. Scientific reports, 11(1):11048, 2021.
- Sean E Cavanagh, John P Towers, Joni D Wallis, Laurence T Hunt, and Steven W Kennerley. Reconciling persistent and dynamic hypotheses of working memory coding in prefrontal cortex. *Nature communications*, 9(1):3498, 2018.
- Rishidev Chaudhuri, Berk Gerçek, Biraj Pandey, Adrien Peyrache, and Ila Fiete. The intrinsic attractor manifold and population dynamics of a canonical cognitive circuit across waking and sleep. *Nature neuroscience*, 22(9):1512–1520, 2019.
- Mark M Churchland, John P Cunningham, Matthew T Kaufman, Justin D Foster, Paul Nuyujukian, Stephen I Ryu, and Krishna V Shenoy. Neural population dynamics during reaching. *Nature*, 487(7405):51–56, 2012.
- Zach Cohen, Brian DePasquale, Mikio C Aoi, and Jonathan W Pillow. Recurrent dynamics of prefrontal cortex during context-dependent decision-making. bioRxiv, pages 2020–11, 2020.
- Fatih Dinc, Adam Shai, Mark Schnitzer, and Hidenori Tanaka. CORNN: Convex optimization of recurrent neural networks for rapid inference of neural dynamics. In Thirty-seventh Conference on Neural Information Processing Systems, 2023. URL https://openreview.net/forum?id=GGIA1p9fDT.

Proceedings Track

- Fatih Dinc, Marta Blanco-Pozo, David Klindt, Francisco Acosta, Yiqi Jiang, Sadegh Ebrahimi, Adam Shai, Hidenori Tanaka, Peng Yuan, Mark J Schnitzer, et al. Latent computing by biological neural networks: A dynamical systems framework. arXiv preprint arXiv:2502.14337, 2025.
- Sadegh Ebrahimi, Jérôme Lecoq, Oleg Rumyantsev, Tugce Tasci, Yanping Zhang, Cristina Irimia, Jane Li, Surya Ganguli, and Mark J Schnitzer. Emergent reliability in sensory cortical coding and inter-area communication. *Nature*, 605(7911):713–721, 2022.
- Richard J Gardner, Erik Hermansen, Marius Pachitariu, Yoram Burak, Nils A Baas, Benjamin A Dunn, May-Britt Moser, and Edvard I Moser. Toroidal topology of population activity in grid cells. *Nature*, 602(7895):123–128, 2022.
- Dar Gilboa, Bo Chang, Minmin Chen, Greg Yang, Samuel S Schoenholz, Ed H Chi, and Jeffrey Pennington. Dynamical isometry and a mean field theory of lstms and grus. arXiv preprint arXiv:1901.08987, 2019.
- Amber Hu, David Zoltowski, Aditya Nair, David Anderson, Lea Duncker, and Scott Linderman. Modeling latent neural dynamics with gaussian process switching linear dynamical systems. *Advances in Neural Information Processing Systems*, 37:33805–33835, 2024.
- Bariscan Kurtkaya, Fatih Dinc, Mert Yuksekgonul, Marta Blanco-Pozo, Ege Cirakman, Mark Schnitzer, Yucel Yemez, Hidenori Tanaka, Peng Yuan, and Nina Miolane. Dynamical phases of short-term memory mechanisms in RNNs. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=ybBuwg0P0d.
- Christopher Langdon, Mikhail Genkin, and Tatiana A Engel. A unifying perspective on neural manifolds and circuits for cognition. *Nature Reviews Neuroscience*, pages 1–15, 2023.
- Nicolas Y Masse, Guangyu R Yang, H Francis Song, Xiao-Jing Wang, and David J Freedman. Circuit mechanisms for the maintenance and manipulation of information in working memory. *Nature neuroscience*, 22(7):1159–1167, 2019.
- Francesca Mastrogiuseppe and Srdjan Ostojic. Linking connectivity, dynamics, and computations in low-rank recurrent neural networks. *Neuron*, 99(3):609–623, 2018.
- Ethan M Meyers, David J Freedman, Gabriel Kreiman, Earl K Miller, and Tomaso Poggio. Dynamic population coding of category information in inferior temporal and prefrontal cortex. *Journal of neurophysiology*, 100(3):1407–1419, 2008.
- Chethan Pandarinath, Daniel J O'Shea, Jasmine Collins, Rafal Jozefowicz, Sergey D Stavisky, Jonathan C Kao, Eric M Trautmann, Matthew T Kaufman, Stephen I Ryu, Leigh R Hochberg, et al. Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature methods*, 15(10):805–815, 2018.
- Matthew G Perich and Kanaka Rajan. Rethinking brain-wide interactions through multiregion 'network of networks' models. Current opinion in neurobiology, 65:146–151, 2020.

- Kanaka Rajan, Christopher D Harvey, and David W Tank. Recurrent network models of sequence generation and memory. *Neuron*, 90(1):128–142, 2016.
- David A Sabatini and Matthew T Kaufman. Reach-dependent reorientation of rotational dynamics in motor cortex. *Nature Communications*, 15(1):7007, 2024.
- Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. arXiv preprint arXiv:1312.6120, 2013. URL https://arxiv.org/abs/1312.6120.
- Eelke Spaak, Kei Watanabe, Shintaro Funahashi, and Mark G Stokes. Stable and dynamic coding for working memory in primate prefrontal cortex. *Journal of neuroscience*, 37(27): 6503–6516, 2017.
- Carsen Stringer, Marius Pachitariu, Nicholas Steinmetz, Charu Bai Reddy, Matteo Carandini, and Kenneth D Harris. Spontaneous behaviors drive multidimensional, brainwide activity. *Science*, 364(6437):eaav7893, 2019.
- Jake P Stroud, John Duncan, and Máté Lengyel. The computational foundations of dynamic coding in working memory. *Trends in Cognitive Sciences*, 2024.
- 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.
- Adrian Valente, Jonathan W Pillow, and Srdjan Ostojic. Extracting computational mechanisms from neural data using low-rank rnns. Advances in Neural Information Processing Systems, 35:24072–24086, 2022.
- Greg Yang, Edward J Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. arXiv preprint arXiv:2203.03466, 2022.
- Guangyu Robert Yang, Madhura R Joglekar, H Francis Song, William T Newsome, and Xiao-Jing Wang. Task representations in neural networks trained to perform many cognitive tasks. *Nature neuroscience*, 22(2):297–306, 2019.
- Rafael Yuste. From the neuron doctrine to neural networks. *Nature reviews neuroscience*, 16(8):487–497, 2015.

Proceedings Track

Table 1: Best learning rates (α) for each task, optimizer, and network size N demonstrated in Figure 1.

\overline{N}	FF-Adam	FF-SGD	DMT-Adam	DMT-SGD	DCDT-Adam	DCDT-SGD
100	10^{-3}	10^{-1}	10^{-4}	10^{-2}	10^{-3}	10^{-1}
177	10^{-3}	10^{-1}	10^{-4}	10^{-2}	10^{-4}	10^{-1}
316	10^{-3}	10^{-1}	10^{-4}	10^{-2}	10^{-4}	10^{-1}
562	10^{-3}	10^{-2}	10^{-5}	10^{-2}	10^{-4}	10^{-2}
1000	10^{-4}	10^{-2}	10^{-5}	10^{-2}	10^{-5}	10^{-2}
1778	10^{-4}	10^{-2}	10^{-5}	10^{-2}	10^{-5}	10^{-2}
3162	10^{-4}	10^{-2}	10^{-1}	10^{-2}	10^{-5}	10^{-2}
5623	10^{-4}	10^{-2}	10^{-2}	10^{-2}	10^{-5}	10^{-2}
10000	10^{-5}	10^{-2}	10^{-3}	10^{-3}	10^{-5}	10^{-2}

Appendix A. Methods

Low-rank RNNs approximation:

To reduce the dimensionality of recurrent dynamics while preserving essential computations, we employed a low-rank approximation of the recurrent connectivity matrix. Specifically, instead of maintaining a full recurrent weight matrix $W \in \mathbb{R}^{N \times N}$, where N denotes the number of recurrent units, we factorized it into a dot product of two low-rank matrices:

$$W \approx \sum_{i=1}^{K} m_i n_i^T, \tag{10}$$

where $m \in \mathbb{R}^{N \times K}$ and $n \in \mathbb{R}^{K \times N}$, with $K \ll N$. This factorization enforces a rank-K constraint on the recurrent connectivity, ensuring that the dynamics are confined to a K-dimensional latent subspace. To illustrate this low-rank approximation, we can define the following:

$$\kappa(t) = n^T r(t), \quad \kappa(t) \in \mathbb{R}^K,$$
(11)

where $r(t) \in \mathbb{R}^N$ denotes the high-dimensional recurrent state of the network and $\kappa \in \mathcal{R}^K$ denotes the latent representations. Here, n is the *encoding weights* projecting population activity into the low-dimensional latent subspace (constrained by low-rank approximation).

Training with low-rank RNNs:

We trained a rank-one RNN, parametrized through low rank factorization of the reconnectivity matrix, and propagated in low-dimensional space (*i.e.*, latent space) rather than evolving it in high-dimensional full neural state vectors (*i.e.*, RNN's update equation). More specifically, we employed following time evolution method to train our large networks:

$$\tau \dot{\kappa}(t) = -\kappa(t) + \frac{1}{N} n^T \tanh(m\kappa(t) + W^{\text{in}} u(t)). \tag{12}$$

Here, N denotes the total number of neurons in the recurrent population, m represents the embedding weights, and n the encoding weights. The task input at time t is denoted by u(t), with W^{in} specifying the input weight matrix.

The networks were trained using the mean squared error (MSE) as the loss function, and model parameters were optimized with the Adam optimizer. Mean accuracy (Fig. 5B) was assessed by computing the correlation coefficient between predicted and target trajectories and averaged out across various trials using PyTorch's built-in *correlation-coefficient* method. In addition, we recorded the computational cost of training per epoch (Fig. 5C) by measuring the elapsed time during execution using Python's *time* module.

Training Setups:

We trained recurrent neural networks (RNNs), with dynamics governed by a leaky integration update and nonlinearity tanh. Training hyperparameters were as follows for all experiments:

Table 2: Training setups for learning rate collapse RNN experiments.

Parameter	Specification
Random seeds	10 independent runs per configuration
Learning rates	$\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$
Batch size	50 trials during training; 200 trials for evaluation/analysis
Epochs	5,000 epochs (all runs completed; no adaptive early stop-
	ping)
Optimizers	Adam and SGD (default torch parameters), with mean squared error (MSE) loss
N-4l:	
Network sizes	13 Hidden dimensions sampled logarithmically between 10^1 and 10^4
Computational	Parallelized cluster jobs (up to 100 processes). Training ex-
resources	ecuted primarily on CUDA GPUs (H100 NVL/RTX 3090);
	small-scale runs on CPU. Epoch runtime measured with
	Python time module.

Proceedings Track

Table 3: Training setups for our weight rescaling RNN experiments.

Parameter	Specification
Random seeds	5 independent runs per configuration
Learning rates	$\{10^{-2}, 10^{-1}, 10^0\}$
Batch size	50 trials during training; 50 trials for evaluation/analysis
Epochs	2,000 epochs (all runs completed; no adaptive early stop-
	ping)
Optimizers	Adam (default torch parameters), with mean squared error
	(MSE) loss
Network sizes	$\{500, 1000, 3000, 5000\}$
Computational	Parallelized cluster jobs. Training executed primarily on
resources	CUDA GPUs (H100 NVL/RTX 3090); small-scale runs on
	CPU. Epoch runtime measured with Python time module.

Table 4: Training setups for our LPU RNN experiments.

Parameter	Specification		
Random seeds	5 independent runs per configuration		
Learning rates	$\{10^{-2}, 10^{-1}\}$		
Batch size	50 trials during training; 50 trials for evaluation/analysis		
Epochs	500 epochs (all runs completed; no adaptive early stopping)		
Optimizers	Adam (default torch parameters), with mean squared error		
	(MSE) loss		
Network sizes	$\{10^3, 3*10^3, 5*10^3, 10^4, 3*10^4, 10^5\}$		
Computational	Parallelized cluster jobs. Training executed primarily on		
resources	CUDA GPUs (RTX 3090). Epoch runtime measured with		
	Python time module.		

Rescaling Procedure

A key challenge in scaling recurrent neural networks lies in the growth of activity magnitudes with network size. The contribution of the recurrent weights to the hidden state can be written as a dot product

$$(W_{\text{rec}}h_t)_i = \sum_{j=1}^N W_{\text{rec}}^{(i,j)} h_t^{(j)},$$
(13)

where each hidden unit i accumulates input from N presynaptic units. Similarly, the output readout is given by

$$(W_{\text{out}}h_t)_k = \sum_{j=1}^N W_{\text{out}}^{(k,j)} h_t^{(j)}.$$
 (14)

As N increases, these summations scale with the number of terms, leading to larger effective magnitudes. Without correction, this growth destabilizes training and forces the learning rate to shrink approximately as 1/N to maintain stability.

To counteract this effect, we applied a rescaling procedure in which both recurrent and output weights were normalized by network size:

$$W_{\rm rec} \rightarrow \frac{W_{\rm rec}}{N}, \qquad W_{\rm out} \rightarrow \frac{W_{\rm out}}{N}.$$
 (15)

This normalization ensures that the total contribution of recurrent interactions and output projections remains approximately constant as N increases, rather than diverging with the number of units. By contrast, the input projection

$$(W_{\rm in}u_t)_i = \sum_{m=1}^{d_{\rm in}} W_{\rm in}^{(i,m)} u_t^{(m)}, \tag{16}$$

only sums over the input dimensionality $d_{\rm in}$, which is fixed and independent of the hidden size N. Therefore, its magnitude does not grow with N, and there is no need to normalize $W_{\rm in}$. Similarly, biases are additive constants and do not scale with network size, so they were also left unmodified.

Statistical treatment: To assess training outcomes, we evaluated performance across multiple random seeds for each configuration. A run was classified as *successful* if it achieved accuracy above 0.5 (chance level for binary discrimination tasks). We then computed the *success density*, defined as the fraction of seeds that met this threshold. This measure provided a robust estimate of training reliability for each setting. Critical learning rates were identified as the largest learning rates at which at least 50% of runs succeeded, enabling power-law fits across network sizes.

Reproducibility: Upon publication, we plan to release the full codebase along with all trained RNN models to facilitate transparency and reproducibility.