The Role of Synaptic Strength in Learning and Structural Robustness in RNNs

Fatih Dinc*
KITP, Geometric Intelligence Lab
UC Santa Barbara

Edouard dePonnat*
Geometric Intelligence Lab
UC Santa Barbara

Henrik Weyer
KITP
UC Santa Barbara

Yanin Guerra
KITP
UC Santa Barbara

Nina Miolane
Geometric Intelligence Lab
UC Santa Barbara

Abstract

Recurrent neural networks (RNNs) trained on neuroscience tasks tend to learn low-rank changes to their connectivity, but the role of synaptic strength in this setting remains unclear. Theory of low-rank RNNs has mostly assumed weak synapses with structured connectivity, while classical random network theory focused on strong synapses at extensive ranks. Here, we study RNNs initialized with rank-deficient, fully trainable random connectivity and find that strong and weak synapses offer different advantages. Strong random synapses support diverse nonlinear dynamics at initialization, including numerical signatures of chaos. Weak synapses instead suppress perturbations to their dynamics under random neuron loss as the population grows. Trained networks show features of both regimes. Strongly initialized networks can learn rapidly even at low rank, across several tasks and in both vanilla and gated RNNs. Their final weights retain substantial initial strong connectivity, while learned changes decrease more rapidly with network size, approaching weak-synapse scaling. Training with recurrent dropout further shifts these changes toward weak scaling. Synaptic strength, alongside rank, thus influences both learning and structural robustness in RNNs.

1 Introduction

Many real-world networks have connectivity dominated by a few effective dimensions despite having a vast number of components [1]. Brains, too, seem to follow this broader pattern [2, 3, 4]. Although they harbor vast numbers of synapses [5] and exhibit high-dimensional neural population activity [6, 7, 8], population-level computations often unfold in low-dimensional subspaces [9, 10, 11, 12]. To study how these computations arise, computational neuroscientists have long studied recurrent neural networks (RNNs) trained on cognitive and motor tasks that mimic those performed by animals in laboratories [13, 14, 15, 16, 17, 18, 19]. These models allow us to examine how learning shapes connectivity and how that connectivity gives rise to dynamics and computation [19].

Low-rank structure provides a particularly tractable connection between these levels of description. Theoretical work has shown that low-rank synaptic connectivity naturally defines low-dimensional coding variables in RNNs [19, 20, 21]. Empirical studies further show that task-trained RNNs often develop low-rank updates or effectively low-rank computational structure [16, 17, 18, 19], and optimization choices can favor such structure during learning [22]. Low-rank structure also supports adaptation, compression, and multitask computation in broad network architectures [23, 24, 25]. Yet connectivity that supports a learned task need not provide a useful starting point for acquiring it. Biological circuits must also maintain their computations as their neurons and synapses change, while

*These authors contributed equally to this work.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

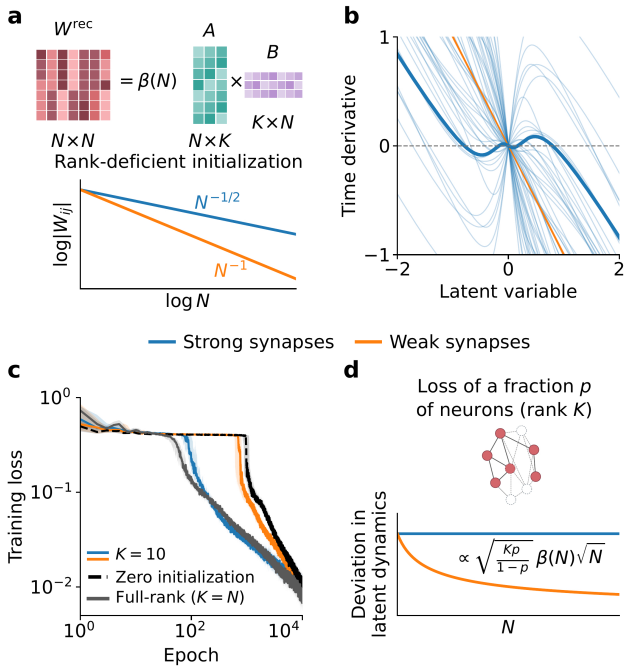

Figure 1: **Two synaptic scaling regimes in rank-deficient initialization.** **a** Rank-K recurrent connectivity with weak ($\beta(N) = N^{-1}$) or strong ($\beta(N) = N^{-1/2}$) synapse initialization. **b** Rank-one phase portraits with weak (orange) and strong (blue) random synapses ($N = 20{,}000$). Weak-synapse realizations nearly coincide whereas strong synapses yield diverse dynamics, including tristability (thick curve; Section 2). **c** Learning dynamics of RNNs initialized with weak (orange) or strong (blue) synapses on a behavioral task (Section 3). Curves: median and IQR across 20 seeds. **d** Deviation in latent dynamics under independent neuron loss. At fixed K and removal probability p, weak but not strong synapses suppress the deviation as N increases (Section 4).

continuing to learn throughout life [26]. Reconciling these diverse observations raise two questions: does starting with low-rank connectivity come at a cost for learning, and what makes the resulting computation robust to perturbations?

Existing theory approaches these questions from two different limits; each focuses on a different regime typically and fixes a hidden control variable, *i.e.*, *the strength of synaptic connections* $\beta(N)$, where N is the number of neurons in the network (Fig. 1**a**). Classical random-network theory studies strong random connections ($\beta(N) = N^{-0.5}$) that sustain fluctuations and nonlinear dynamics, including chaos, in high-dimensional networks [27, 28, 29]. Related work links connectivity spectra to the propagation of learning signals, motivating strong initialization near criticality to facilitate learning [30]. Low-rank RNN theory instead typically assumes weak connections ($\beta(N) = N^{-1}$), whose structured contributions distribute computation across the population and can reduce its dependence on individual neurons or connections [20, 21, 31, 32]. These models provide a tractable framework for building and analyzing task solutions [16, 19, 33], which often requires that learning establishes correlations between the low-rank components that make up the *weak* synaptic connections.

Joining these two lines of work, we show that they predict complementary advantages of synaptic strength, dynamical richness and robustness, before and after learning. Specifically, to examine whether learning can bring these advantages into the same circuit, we introduce *rank-deficient initialization*, varying the rank and strength of random recurrent connectivity independently while allowing all recurrent weights to change during training (Fig. 1). Our contributions are:

1. We show how synaptic strength determines whether random rank-deficient connectivity supports rich or asymptotically quiescent dynamics at initialization (Proposition 1).
2. We provide numerical evidence that strong random synaptic connectivity can support chaotic dynamics at a *finite* rank (Fig. 3), which extends findings of chaos in full-rank RNNs [27, 28] and rank-deficient RNNs with extensive ranks [29].
3. We demonstrate that random low-rank initialization can support learning speeds comparable to its full-rank counterpart under strong synaptic scaling (Fig. 4).
4. We establish a robustness advantage of weak synaptic scaling under random neuron loss and identify a tendency toward weak scaling in learned updates (Proposition 2, Figs. 5-S15).

Together, our results show that low-rank connectivity can support both rapid learning and robustness to neuron loss, and identify synaptic strength, which has often been coupled with rank in existing models, as a key factor influencing these advantages.

2 Rank-deficient initialization in RNNs

2.1 Rank and synaptic strength as independent initialization choices

We consider RNNs whose autonomous dynamics follow

$$\tau \dot{\boldsymbol{r}} = -\boldsymbol{r} + \phi(\boldsymbol{W}^{\mathrm{rec}}\boldsymbol{r}), \qquad \boldsymbol{W}^{\mathrm{rec}} = \beta(N)\boldsymbol{A}\boldsymbol{B}, \tag{1}$$

where $\boldsymbol{r} \in \mathbb{R}^N$ contains the neuron state variables, τ is the neuronal time constant, and the nonlinearity ϕ acts element-wise. The matrices $\boldsymbol{A} \in \mathbb{R}^{N\times K}$ and $\boldsymbol{B} \in \mathbb{R}^{K\times N}$ specify recurrent connectivity of rank at most K. We parameterize the entry distributions of $\boldsymbol{A}$ and $\boldsymbol{B}$ using a gain g and rank K, with no dependence on N to leading order at fixed K. The factor $\beta(N)$ then sets how connection strengths scale with network size (Fig. 1). This separates the choice of initialization rank from the scaling of the synaptic strength. We allow K to range up to N, rather than requiring $K \ll N$, and thus use the name "rank-deficient initialization" to refer to Eq. (1).

In this work, we study four main rank-deficient initialization methods:

1. **Strong random synapses:** We sample independent entries $A_{i\alpha} \sim \mathcal{N}(0, g^2/K)$ and $B_{\alpha j} \sim \mathcal{N}(0,1)$, and set $\beta(N) = N^{-1/2}$. Jointly, these choices give $\mathrm{Var}(W^{\mathrm{rec}}_{ij}) = g^2/N$.
2. **Rank-tempered initialization [34]:** We use $\beta(N) = N^{-1}$ with independent entries $A_{i\alpha} \sim \mathcal{N}(0, g^2)$ and $B_{\alpha j} \sim \mathcal{N}(0,1)$, giving $\mathrm{Var}(W^{\mathrm{rec}}_{ij}) = Kg^2/N^2$. This initialization gives weak synaptic scaling at fixed K and strong scaling when $K \sim O(N)$.
3. **Weak correlated synapses:** We set $\boldsymbol{W}^{\mathrm{rec}} = (g/N)\boldsymbol{X}(\rho\boldsymbol{X} + \sqrt{1-\rho^2}\boldsymbol{Y})^T$, where $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{N\times K}$ have independent standard Gaussian entries and $\rho \in [0,1]$ induces correlations between the low-rank components.
4. **Weak orthogonal synapses:** We set $\boldsymbol{W}^{\mathrm{rec}} = g\boldsymbol{U}\boldsymbol{Q}\boldsymbol{U}^T$, where $\boldsymbol{U} \in \mathbb{R}^{N\times K}$ is sampled uniformly from matrices with orthonormal columns and $\boldsymbol{Q} \in \mathbb{R}^{K\times K}$ is independently sampled uniformly from the orthogonal group. This gives an average synaptic variance $\mathrm{Var}(W^{\mathrm{rec}}_{ij}) = g^2K/N^2$.

Whenever we train RNNs, we use this factorization only to initialize $\boldsymbol{W}^{\mathrm{rec}}$. In contrast to models that constrain connectivity to remain low rank throughout training [19, 20], all entries of the recurrent matrix are subsequently trainable. Varying K therefore changes the starting connectivity while retaining the same number of recurrent parameters available for learning.

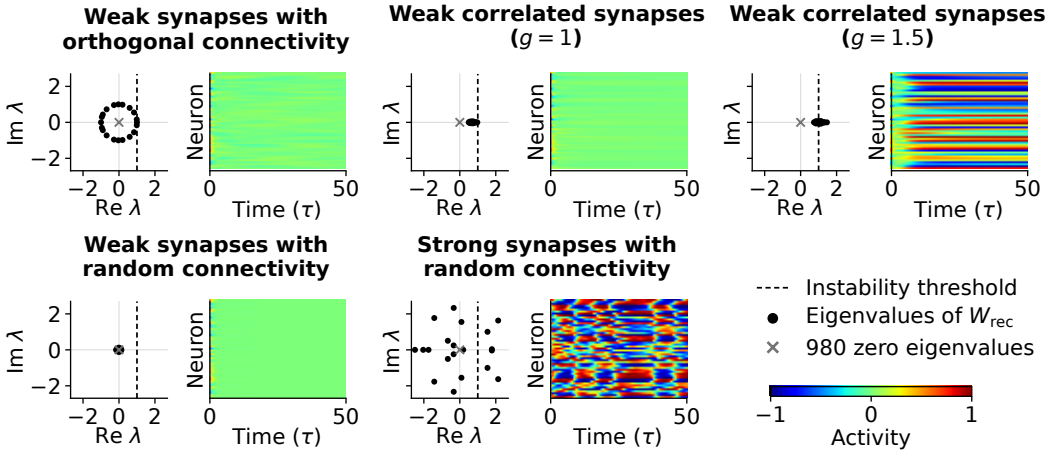

Figure 2: **Synaptic connectivity shapes spectral structure and spontaneous dynamics at initialization.** Eigenvalue spectra of the recurrent matrix $\boldsymbol{W}^{\mathrm{rec}}$ and activity kymographs for five rank-deficient initialization schemes ($N = 1{,}000$, $K = 20$). Dashed lines mark the instability threshold $\mathrm{Re}\,\lambda = 1$. Kymographs show the activity of 100 example neurons, whose initial activity was sampled uniformly and independently from $[-1, 1]$.

2.2 Spectral properties of RNNs at rank-deficient initialization

For the connectivity in Eq. (1), the recurrent dynamics admit a closed description in terms of K latent variables, as in low-rank RNNs [19, 21]. Defining the projection $\boldsymbol{\kappa} = \beta(N)\boldsymbol{B}\boldsymbol{r} \in \mathbb{R}^K$, Eq. (1) gives

$$\tau\dot{\boldsymbol{\kappa}} = -\boldsymbol{\kappa} + \beta(N)\boldsymbol{B}\phi(\boldsymbol{A}\boldsymbol{\kappa}). \tag{2}$$

This reduction allows us to study the initial dynamics using K variables even when N is large. In the rest of this section, we study networks whose connectivity remains fixed at initialization, whereas Sections 3 and 4 will study task-trained RNNs.

A straightforward calculation (Appendix S1.1) shows that the overlap matrix $\boldsymbol{O} = \beta(N)\boldsymbol{B}\boldsymbol{A}$ has the same non-zero eigenvalues as $\boldsymbol{W}^{\mathrm{rec}} = \beta(N)\boldsymbol{A}\boldsymbol{B}$, whereas the latter has further $N - K$ zero eigenvalues. For $\phi = \tanh$, as is the case for majority of our experiments below, linearizing Eq. (2) at the origin gives $\tau\boldsymbol{J}_0 = -\boldsymbol{I}_K + \boldsymbol{O}$. The origin is therefore linearly stable when all eigenvalues of $\boldsymbol{W}^{\mathrm{rec}}$ have real parts below one and unstable when any crosses this threshold.

Figure 2 illustrates how synaptic strength and connectivity structure shape the eigenvalue spectra and neural activity of the network. Weak random connectivity has eigenvalues near zero, which leads to activity decaying toward the origin in the example shown. Orthogonal connectivity places its nonzero eigenvalues on a circle of radius g. Correlated weak and strong random connectivity can cross the instability threshold and sustain nonzero activity.

2.3 Synaptic scaling shapes the richness of initial dynamics

Low-rank connectivity can support nonlinear computations by inducing correlations between entries of $\boldsymbol{A}$ and $\boldsymbol{B}$ [19, 21]. With independent and zero-centered entries and under weak scaling, however, the latent vector field becomes simply $\tau\dot{\boldsymbol{\kappa}} = -\boldsymbol{\kappa}$ as $N \to \infty$. Here, we show that rich nonlinear dynamics can also arise without prescribing these correlations, *i.e.*, by using strong synapses:

Proposition 1 (Self-averaging and realization-dependent latent dynamics)**.** *Weak scaling yields a deterministic population-averaged recurrent field, whereas strong scaling can retain nonzero variability across network realizations even when the mean recurrent field vanishes. Precisely,*
Weak synapses: *Following common low-rank RNN constructions [21], fix K and let $(\boldsymbol{a}_i, \boldsymbol{b}_i)$ denote the ith row of $\boldsymbol{A}$ and column of $\boldsymbol{B}$, respectively, written as vectors for $i = 1, \ldots, N$. Suppose these pairs are sampled independently across neurons from a common joint distribution that does not depend on N and has finite second moments. Define $\boldsymbol{F}(\boldsymbol{\kappa}) = \mathbb{E}[\boldsymbol{b}\tanh(\boldsymbol{a}^T\boldsymbol{\kappa})]$. Under weak scaling $\beta(N) = N^{-1}$, the recurrent field converges almost surely at each fixed $\boldsymbol{\kappa}$ to $\boldsymbol{F}(\boldsymbol{\kappa})$, giving the deterministic latent vector field $-\boldsymbol{\kappa} + \boldsymbol{F}(\boldsymbol{\kappa})$ for $\tau\dot{\boldsymbol{\kappa}}$ as $N \to \infty$.*
Strong synapses: *Additionally suppose all entries of $\boldsymbol{A}$ and $\boldsymbol{B}$ are independent and the entries of $\boldsymbol{B}$ have zero mean. Then $\boldsymbol{F}(\boldsymbol{\kappa}) = 0$ for every $\boldsymbol{\kappa}$. Under strong scaling $\beta(N) = N^{-1/2}$ and as $N \to \infty$, the recurrent term $\tau\dot{\boldsymbol{\kappa}} + \boldsymbol{\kappa}$ converges jointly in distribution at any finite collection of latent states to a zero-mean Gaussian process $\boldsymbol{G}$ with covariance*

$$\tau\dot{\boldsymbol{\kappa}} \to -\boldsymbol{\kappa} + \boldsymbol{G}(\boldsymbol{\kappa}), \quad \mathbb{E}[G_\alpha(\boldsymbol{\kappa})G_\beta(\boldsymbol{\kappa}')] = \mathbb{E}\big[b_\alpha b_\beta \tanh(\boldsymbol{a}^T\boldsymbol{\kappa})\tanh(\boldsymbol{a}^T\boldsymbol{\kappa}')\big] \tag{3}$$

The result follows from the law of large numbers and the multivariate central limit theorem (Appendix S1.2). For independent centered Gaussian entries, neuronal contributions cancel on average but fluctuate across realizations. Weak scaling suppresses these fluctuations as N grows, whereas strong scaling retains them at order one. With $\boldsymbol{a} \sim \mathcal{N}(0, g^2\boldsymbol{I}_K/K)$ and $\boldsymbol{b} \sim \mathcal{N}(0, \boldsymbol{I}_K)$ independent, Eq. (3) reduces to $\delta_{\alpha\beta}\mathbb{E}_{\boldsymbol{a}}[\tanh(\boldsymbol{a}^T\boldsymbol{\kappa})\tanh(\boldsymbol{a}^T\boldsymbol{\kappa}')]$. The resulting Gaussian process describes a fixed nonlinear field for each network, providing a route to dynamical richness without designing correlations between entries of $\boldsymbol{A}$ and $\boldsymbol{B}$. At a chosen rank K, a single random draw specifies an entire nonlinear field over the K latent variables. Correlations between its values at different latent states are described by Eq. (3), allowing nonlinear structure to arise from randomness in a single quenched realization.

Figure 1**b** illustrates this possibility already at rank one. Weak independent random realizations nearly coincide with $\tau\dot{\kappa} = -\kappa$, whereas strong realizations vary across seeds and can support multiple fixed points. At $K = 1$, weak scaling with centered jointly Gaussian entries of $\boldsymbol{A}$ and $\boldsymbol{B}$ can support monostability or bistability depending on their correlations in the large-N limit [21]. Strong random initialization can instead produce tri-stability, as in the highlighted realization.

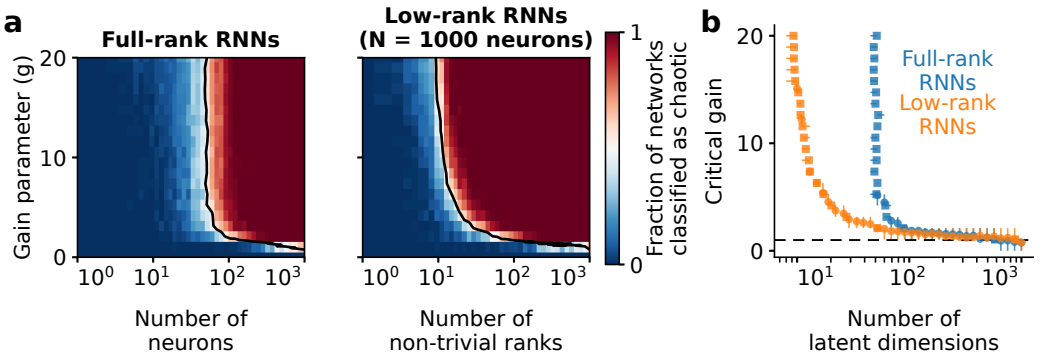

Figure 3: **Numerical signatures of chaos emerge at low initialization ranks.** **a** Fraction of networks classified as chaotic by finite-time divergence of nearby trajectories over gain g and latent dimensionality. *Left.* Full-rank RNNs with N neurons. *Right.* Low-rank RNNs with $N = 1000$ neurons and rank K. Black curves mark the boundary shown in panel **b**. $\mathrm{Var}[W_{ij}^{\mathrm{rec}}] = g^2/N$ for all RNNs. **b** Critical gain for full-rank (blue) and low-rank (orange) networks. Error bars span neighboring samples bracketing the 0.5 crossing. The dashed line marks $g = 1$, expected as $K \to \infty$ at fixed K/N [27, 29].

2.4 Numerical signatures of chaos in low-rank RNNs

Earlier work has shown that full-rank RNNs [27, 28] and rank-deficient RNNs whose rank grows proportionally with population size [29] can exhibit chaos. Both settings use strong synapses, raising the question of whether strong synapses could also support chaos when rank is much smaller than the population size, $K \ll N$. To explore this possibility, we compare full-rank networks with varying numbers of neurons to rank-deficient networks with $N = 1000$ and varying rank K, using zero-mean strong synapses with variance g^2/N in both ensembles.

First, it is instrumental to distinguish the edge of stability from the edge of chaos. The former marks where perturbations to the origin begin to grow, and approximately coincides between the two ensembles at matched dimensionality of the dynamics (Fig. S1, Proposition S1). An unstable origin, however, need not imply chaos, since trajectories may instead settle at another fixed point or a periodic orbit. The edge of chaos instead concerns persistent divergence of nearby trajectories along the network's ongoing dynamics. We therefore assess numerical signatures of chaos through finite-time trajectory divergence (Appendix S2.1). Unlike origin instability, these signatures become common at lower latent dimensionality in rank-deficient networks than in the full-rank comparison (Fig. 3) and suggest that strong random connectivity can support chaos at ranks much smaller than N.

3 Trainability with rank-deficient initialization

Strong random synapses can support rich dynamics even at low rank, but does this make rank-deficient connectivity a useful starting point for learning? To examine this question, we introduce inputs, biases, and trainable readouts into our RNNs (Appendix S2.2) and train networks with $N = 1000$ neurons. Below, we first showcase our results on two canonical working memory tasks and then discuss the generalization to more complex tasks and gated architectures.

3.1 Learning speed on working-memory tasks

In delayed cue discrimination (DCD), the network receives a $+1$ or -1 cue for a fixed duration and reports its identity with a matching ± 1 response after a predefined delay. The target response is zero during cue presentation and the delay. In delayed match-to-sample (DMTS), the network receives two cues separated by a delay and reports whether they agree, with a response of 1 for matching cues and 0 otherwise. Both tasks require retaining information until the designated response period (see Appendix S2.3 for further details). We vary the initialization rank K and gain g, and compare strong random and rank-tempered initialization. As defined in Section 2.1, the former uses strong synapses at all ranks, whereas the latter gives weak synaptic scaling at fixed K and strong scaling when K

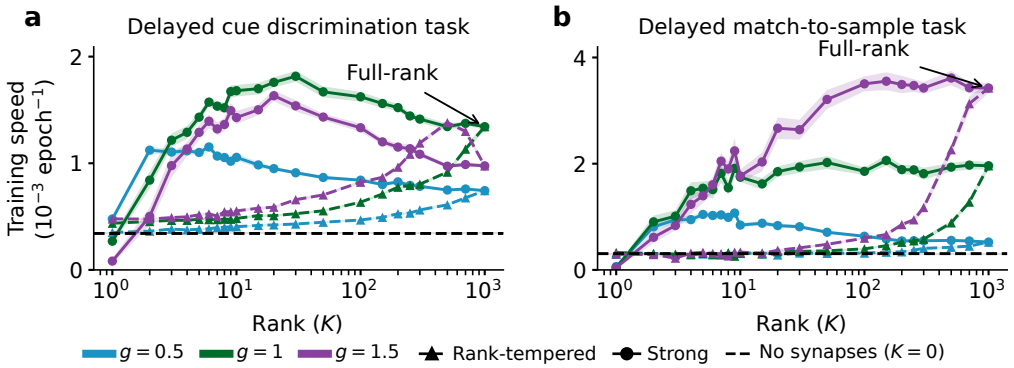

Figure 4: **Rank-deficient initialization with strong synapses supports rapid learning in two common behavioral tasks.** Training speed versus initialization rank K for delayed cue discrimination (**a**) and delayed match-to-sample (**b**) tasks in RNNs with $N = 1{,}000$. Strong and rank-tempered initializations [34] have weight variances g^2/N and Kg^2/N^2, respectively. Speed is the reciprocal of the final epoch of the first 50 consecutive epochs with training MSE below 0.05. Runs without such a window within 10,000 epochs receive zero speed. Curves and shading show mean $\pm$ SEM across all 20 seeds. The black dashed line marks zero recurrent initialization, whereas $K = N = 1{,}000$ marks the full-rank initialization. Both tasks use a learning rate of 3×10^{-5} without gradient clipping.

grows closer to N. By definition, the two constructions agree at $K = N$. All recurrent entries remain trainable, so changing K changes the initial connectivity matrix without changing the number of parameters available for learning. To measure learning speed, we record the first epoch at which the training mean-squared error has remained below 0.05 for 50 consecutive epochs, and take the reciprocal of this epoch number. Networks that reach this criterion earlier have higher learning speeds. Runs that never meet it within the training budget receive zero speed (Appendix S2.4). Results are shown in Fig. 4 for both tasks.

With rank-tempered initialization, learning speed generally increases with rank on both tasks, consistent with prior work [34]. With strong initialization, however, networks can learn as fast as their full-rank counterparts while starting at substantially lower ranks (Fig. 4). The rank required varies with task and gain, but ranks of a few tens were often sufficient. Thus, low-rank initialization need not come at a substantial cost for learning, provided the synapses are sufficiently strong.

3.2 Controls for connectivity structure and optimization

As noted earlier, weak synapses can also support nontrivial dynamics when correlations between entries of $\boldsymbol{A}$ and $\boldsymbol{B}$, or orthogonality constraints within the rank-K subspace, are imposed at initialization. Such constraints can therefore be useful when choosing initializations for artificial RNNs. Indeed, both constructions accelerated DCD learning at low ranks relative to rank-tempered random initialization at suitable gains (Fig. S2), although their learning trajectories were more sensitive to rank at $g = 1$ than those of strong random initialization (Fig. S3). These initializations, however, already contain connectivity structure that, in a biological circuit, would need an explanation through learning, development, or other mechanisms. In contrast, strong synapses offer another possibility, *i.e.*, useful dynamics can arise from random connectivity without prescribing these correlations or constraints beforehand.

To further examine whether the learning-speed comparison depends on optimization choices, we varied learning rates with and without gradient clipping on both working-memory tasks (Fig. S4), then repeated the DCD rank sweep using selected learning rates (Fig. S5). Structured weak initialization matched or exceeded strong random initialization at some gains and ranks, with comparable speeds observed near $K = 10$ for suitable gains. Strong random initialization nevertheless supported rapid learning across these settings, suggesting that prescribed connectivity structure is not necessary for low-rank networks to learn rapidly on these tasks.

3.3 Extensions across tasks and architectures

We next examined whether low-rank strong initialization could also support learning beyond the two working-memory tasks. We considered one-dimensional spatial navigation, where networks integrate input velocity to represent position on a circular track, and sequential MNIST, where networks classify images after receiving their pixels in sequence. In the navigation task, strong initialization again reduced the dependence of learning trajectories on initialization rank (Fig. S6**a**) and supported learning speeds comparable to or higher than the other low-rank initializations at suitable gains and learning rates (Fig. S6**b**). In sequential MNIST, however, low-rank strong initialization showed no consistent advantage over rank-tempered initialization or initialization with no recurrent synapses (Fig. S6**c**). Learning-rate sweeps likewise showed that zero recurrent initialization could achieve speeds comparable to those of the nonzero initializations (Fig. S6**d**). Thus, the benefit of initial recurrent connectivity depends on the task. These differences may also be related to the dimensionality of the connectivity acquired during learning, *i.e.*, sequential MNIST networks can develop effective ranks substantially above their initialization rank, whereas working-memory networks more closely retain the initial low-dimensional structure (Fig. S7). We examine these differences further in Section 4.

We also asked whether the same observations hold in gated architectures. We trained gated recurrent units (GRUs) and long short-term memory networks (LSTMs) on DCD, varying the initialization rank of the stacked recurrent gate matrix while allowing all entries to change during training (Appendix S2.2). At suitable gains, strong low-rank initialization supported learning speeds comparable to the full-rank endpoint and faster than the default initialization in both architectures, whereas rank-tempered initialization remained slower at low ranks (Fig. S8). Thus, the ability to learn rapidly from low-rank strong initialization extended beyond vanilla RNNs.

Together, these experiments suggest that starting with random low-rank connectivity need not substantially slow learning when synapses are sufficiently strong. We next ask what remains of the initial connectivity after learning, and how the learned changes relate to structural robustness.

4 Structural stability and learned synaptic scaling

Our results in Section 3 showed that strong random connectivity can support rapid learning even at low initialization ranks. Learning itself can also introduce low-rank structure, even when networks start with full-rank connectivity [17, 18]. Yet, the low dimensionality of these learned changes does not tell us whether their magnitudes follow the strong scaling of the initial network or a different scaling altogether. We therefore next ask how the magnitude of learned changes depends on network size, and what this implies for robustness to neuron loss.

4.1 Learning produces updates with a distinct dependence on network size

To examine how learning changes synaptic strength, we trained networks of different sizes on DCD, DMTS, three-bit flip-flop[2], and sequential MNIST. Throughout the empirical comparisons in this section, we included only networks that met the task-specific success criterion throughout a final evaluation window (Appendix S2.6). Consistent with prior work [17], networks starting from full-rank strong connectivity acquired updates that were low-dimensional relative to network size across all four tasks, which was measured by their singular-value participation ratios (Table S1).

To study how synaptic strength scales with N, we compared zero and full-rank strong initialization, separating the final connectivity into its initial weights and learned changes:

$$\boldsymbol{W}^{\mathrm{rec}}_{\mathrm{final}} = \boldsymbol{W}^{\mathrm{rec}}_{\mathrm{initial}} + \Delta\boldsymbol{W}^{\mathrm{rec}}. \tag{4}$$

For each successfully trained network, we measured synaptic strength using the mean absolute entries of the final matrix and the update, then fitted their dependence on network size (Fig. 5). When starting from zero synapses, learned synaptic strength decreased with network size, with fitted exponents closer to weak than strong synaptic scaling across the four tasks (Fig. 5**a**). Starting from strong connectivity, however, the final synaptic strength decreased more slowly with size, while the strength

[2] In the flip-flop task, commonly studied in computational neuroscience [14], each output retains the sign of the most recent pulse on its corresponding input channel (Appendix S2.3).

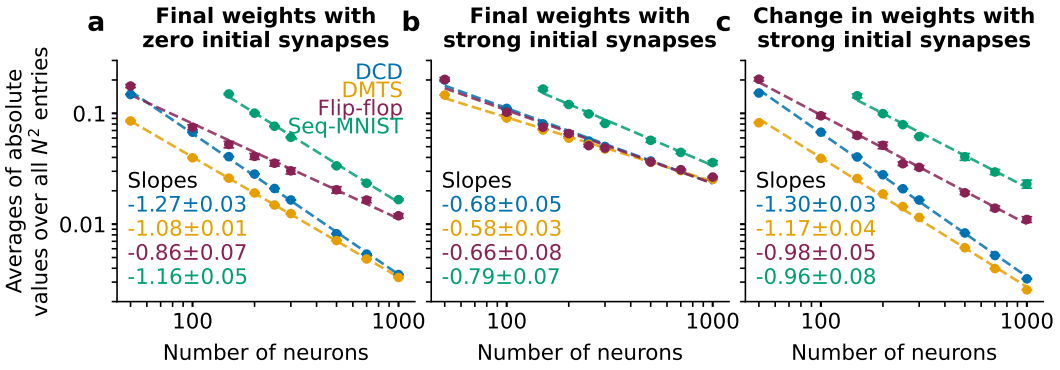

Figure 5: **Synaptic initialization shapes the scaling of learned weights and updates with network size.** Average absolute recurrent matrix entries versus network size N. Dashed lines show power-law fits obtained by linear regression in log-log coordinates. Annotations report slopes $\pm$ 1.96 regression standard errors. **a** Final weights starting from zero synapse initialization. **b** Final weights starting from full-rank strong initialization with independent entries sampled from $W_{ij}^{\mathrm{rec}} \sim \mathcal{N}(0, 1/N)$. **c** Mean absolute differences between final and initial recurrent weights for the networks in **b**. Points and error bars show mean $\pm$ SEM across networks satisfying the task-specific success criterion over the final evaluation window, up to 20 seeds per condition.

of learned updates showed a steeper dependence, close to or faster than $1/N$ (Fig. 5**b,c**). Thus, synaptic strength of the initial/final network and the strength of learned updates can follow different scaling rules within the same network. We also examined whether this pattern depends on how synaptic strength is measured or how the networks are trained. Root-mean-square measurements of synapses and a different learning-rate scaling gave similar qualitative results (Figs. S9-S10), while ReLU, gated, and spiking networks extended the comparison beyond tanh RNNs (Figs. S11-S14). The fitted exponents varied across tasks, gates, and training conditions, though the overall results suggested a tendency toward weak scaling.

As an important limitation, these analyses include all learned changes, which need not be required to perform the task. Optimization may also modify connections that are redundant or have little effect on the learned computation. Identifying which part of the synaptic connections supports a task remains challenging, although progress is being made [18]. Isolating these task-relevant contributions would be crucial for determining how their synaptic strength scales with network size, which we leave as a future work.

4.2 Sensitivity of latent dynamics to neuron loss

Above, we observed that even full-rank RNNs acquired low-dimensional updates whose synaptic strength tended toward weak scaling. Why might this scaling be useful for a learned computation? Biological circuits must maintain their computations despite changes in the neurons that implement them. To isolate the role of synaptic strength, we return to a fixed rank-K network and ask how its latent vector field changes when neurons are randomly lost. Realistically, this construction provides a simplified setting for studying low-dimensional connectivity that could support a task [18].

Let each neuron be independently retained with probability $1-p$, where $0 \leq p < 1$, and let $\boldsymbol{P}$ be the diagonal matrix with $P_j \equiv P_{jj} = 1$ for retained neurons and zero otherwise. We remove all connections to and from the lost neurons and rescale the surviving connectivity by $1/(1-p)$:

$$\widetilde{\boldsymbol{W}}^{\mathrm{rec}} = \frac{1}{1-p}\boldsymbol{P}\boldsymbol{W}^{\mathrm{rec}}\boldsymbol{P} = \frac{\beta(N)}{1-p}(\boldsymbol{P}\boldsymbol{A})(\boldsymbol{B}\boldsymbol{P}). \tag{5}$$

This rescaling compensates for the mean loss of recurrent contributions, allowing us to isolate the variability caused by which neurons are removed. We write the latent dynamics as $\tau\dot{\boldsymbol{\kappa}} = -\boldsymbol{\kappa} + \boldsymbol{g}(\boldsymbol{\kappa})$, with $\phi = \tanh$. Since the leak term is unchanged, the difference between the original and perturbed latent vector fields is simply $\widetilde{\boldsymbol{g}}(\boldsymbol{\kappa}) - \boldsymbol{g}(\boldsymbol{\kappa})$. The following proposition formalizes the effect of such neuron loss:

Proposition 2 (Structural stability to neuron loss)**.** *Consider a perturbation to the recurrent contribution to the latent dynamics, $g_i(\boldsymbol{\kappa}) = \beta(N)\sum_{j=1}^{N} B_{ij}\phi(\sum_{k=1}^{K} A_{jk}\kappa_k)$ for $i = 1, \ldots, K$, induced by the loss of physical neurons. Then, the dynamics are governed by the perturbed synaptic matrix $\widetilde{\boldsymbol{W}}^{\mathrm{rec}}$ [cf. Eq. (5)] and the adjusted recurrent contribution is given by*

$$\widetilde{g}_i(\boldsymbol{\kappa}) = \frac{\beta(N)}{1-p}\sum_{j=1}^{N} P_j B_{ij}\phi\left(\sum_{k=1}^{K} A_{jk}\kappa_k\right), \tag{6}$$

where the P_j are independent, each zero with probability p and 1 otherwise. The zeros in P_j label the dropped neurons. At the fixed latent state $\boldsymbol{\kappa}$, assume that

$$\frac{1}{NK}\sum_{i=1}^{K}\sum_{j=1}^{N}\left[B_{ij}\phi\left(\sum_{k=1}^{K} A_{jk}\kappa_k\right)\right]^2 \longrightarrow C(\boldsymbol{\kappa}) \in (0,\infty) \qquad \text{as } N \to \infty. \tag{7}$$

Thus, upon the deletion of an expected fraction p of neurons, with fixed $0 < p < 1$, we have the asymptotic error:

$$\sqrt{\mathbb{E}_{\boldsymbol{P}}\left[\|\boldsymbol{g}(\boldsymbol{\kappa}) - \widetilde{\boldsymbol{g}}(\boldsymbol{\kappa})\|_2^2\right]} \sim \sqrt{C(\boldsymbol{\kappa})}\sqrt{\frac{Kp}{1-p}}\cdot\beta(N)\sqrt{N}. \tag{8}$$

Here, the expectation is over neuron removal. In particular, for population coding with a fixed latent dimension K, this quantity vanishes as $N^{-1/2} \to 0$ with increasing numbers of physical neurons in the weakly connected regime described by $\beta(N) = N^{-1}$. So, the latent dynamics are structurally stable, in this vector-field sense, to any fixed neuron loss probability $p < 1$. In contrast, in the strongly connected regime $\beta(N) = N^{-1/2}$, the deviation remains $O(1)$ for any fixed $p > 0$, and the latent dynamics are structurally unstable in the same sense at states satisfying the assumption.

The proof is provided in Appendix S1.3. Weak synapses can therefore support structured dynamics whose sensitivity to compensated random neuron loss decreases as the population grows, even when the expected fraction removed remains finite. The root cause connects back to the population averaging in Proposition 1, *i.e.*, weak scaling suppresses fluctuations, whereas strong scaling allows them to persist. At initialization, this persistence permits diverse dynamics across random networks. Under neuron loss, it instead allows the identities of the removed neurons to produce a finite change in the latent vector field. Thus, the scaling that supports dynamical richness from randomness can also leave the dynamics sensitive to changes in the population, while weak scaling makes structured dynamics less dependent on individual neuronal contributions.

4.3 Training with recurrent dropout changes update scaling

Our neuron-loss calculation suggests that weak synaptic scaling can reduce sensitivity to missing neuronal contributions. We therefore asked whether introducing such perturbations during learning would encourage networks to acquire updates with weak synaptic scaling. We trained networks with strong random initialization at different recurrent dropout probabilities and measured how the strength of their learned updates scaled across six network sizes (Fig. S15; see Appendix S2.7).

For DCD and DMTS, the strength of learned updates already scaled near $1/N$ without dropout and remained close to this regime as dropout increased. For flip-flop and sequential MNIST, however, increasing dropout shifted the fitted exponents toward weak scaling, although they did not reach -1 over the tested probabilities. Both mean absolute and root-mean-square measures of synaptic strength showed similar trends (Fig. S16). Thus, perturbations during learning can shift the scaling of learned updates, with the effect depending on the task. Together with the neuron-loss calculation, these findings suggest that weak synaptic scaling may help networks accommodate missing neuronal contributions, although how the learned updates contribute to robustness in the full trained network remains to be established.

5 Discussion and related works

Our results bring together two perspectives on recurrent connectivity: strong random synapses can provide rich initial dynamics and rapid learning, while weak synapses can reduce sensitivity to compensated neuron loss. Trained networks may retain features of both regimes, preserving substantial initial strong connectivity while acquiring updates whose strength tends toward weak scaling.

Two regimes of RNN theory. Classical random-network theory studies strong recurrent interactions that sustain fluctuations and nonlinear dynamics [27, 30], while related work connects recurrent connectivity to learning and gradient propagation [15, 35]. Low-rank theory instead typically uses correlations between entries of $\boldsymbol{A}$ and $\boldsymbol{B}$ to support computations with weak synapses [19, 20, 21, 31, 32]. These models help extract computations, construct task solutions, and identify shared dynamical motifs [16, 25, 33]. Such correlations allow nonlinear recurrent contributions to survive population averaging, whereas independent, centered entries under weak scaling leave only decay toward the origin as $N \to \infty$. Our neuron-loss result permits correlated connectivity, showing how these structured weak networks can also become less sensitive to compensated neuron loss under the stated assumptions. Strong synapses offer a different starting point for learning: nonlinear dynamics can arise from randomness without establishing these correlations beforehand.

Connections to learning and low-rank structure. Dynamical richness before training differs from rich versus lazy learning, which concerns changes in representations during training [36]. Low-rank initialization can favor richer learning depending on task alignment [37], while readout strength and recurrent restructuring also shape learned representations [38, 39]. Following how learning modifies initial dynamics would help connect these observations to synaptic scaling. More broadly, low-rank structure can emerge during optimization in overparameterized networks [22, 40, 41], motivating work on compression, subnetwork selection, and adaptation [23, 24, 42, 43]. We use task-trained RNNs [13] to connect these questions to low-dimensional neural computation [32, 44, 45], while constraining rank only at initialization.

Our hypothesis moving forward. We hypothesize that learning can acquire task-relevant changes with weak synaptic scaling within an initially strong network, combining useful starting dynamics with distributed computation. The update-scaling measurements and their shift under dropout are consistent with this possibility. They do not establish that the entire network becomes weak or that all learned changes support the task. Testing this hypothesis requires isolating task-relevant connectivity [16, 18] and measuring its contribution to robustness. This hypothesis is particularly relevant for biological circuits, where connectivity must support both the acquisition of a computation and its maintenance as neurons and synapses change. Prescribed correlations can provide a useful task solution, but how those correlations arise is itself part of the learning problem. Our results suggest that strong random synapses could provide an initial repertoire of dynamics from which learning develops more structured computations. Whether the learned contributions then become less sensitive to neuron loss remains to be tested directly, without assuming that the full anatomical connectivity is low rank.

Limitations and future directions. Recent studies examined chaos at extensive ranks [29] and the influence of rank on training speed [34]. Our experiments show numerical signatures of chaos and rapid learning at ranks much smaller than the population size, but a theory connecting these observations remains open. In particular, finite-time trajectory divergence does not establish sustained chaos, and strong initialization is not universally faster, *i.e.*, structured weak initialization can match or exceed it under suitable optimization choices. Hence, applicability of our results to modern neural networks remain also open question.

Our theoretical scaling results concern large populations, while the empirical scaling fits cover finite sizes and successfully trained networks. The neuron-loss calculation describes changes in the latent vector field under compensated removal, but whether this translates into preserved task performance requires direct testing. Training-time dropout provides a related intervention but does not replace such a test. Other perturbations, including synaptic noise [46] and representational drift [32, 47], also require separate treatment. Finally, systematically varying task dimensionality and incorporating further realistic constraints would help clarify biological relevance.

Data and code availability. Code for reproducing this work can be found in https://github.com/fatihdinc/Rank-deficient-RNNs.

Acknowledgements

N. M. acknowledges partial support from NSF Career grant 2240158 and NSF grant 2602079. This research was supported in part by NSF Grant PHY-2309135, Gordon and Betty Moore Foundation Grant 2919.02 to the Kavli Institute for Theoretical Physics (KITP), and Simons Foundation International Grant SFI-MPS-T-Institutes-00021142, LB. This work also benefited from support from the Mitchel Postdoctoral Scholar Career Development Fund. We thank OpenAI for providing complimentary ChatGPT Pro subscriptions that supported this research. We also thank Chris V. Nicholson for facilitating this access.

References

[1] Vincent Thibeault, Antoine Allard, and Patrick Desrosiers. The low-rank hypothesis of complex systems. *Nature Physics*, 20(2):294–302, 2024.

[2] Matthew G Perich, Devika Narain, and Juan A Gallego. A neural manifold view of the brain. *Nature Neuroscience*, pages 1–16, 2025.

[3] Lea Duncker and Maneesh Sahani. Dynamics on the manifold: Identifying computational dynamical activity from neural population recordings. *Current opinion in neurobiology*, 70:163–170, 2021.

[4] Louis K Scheffer, C Shan Xu, Michal Januszewski, Zhiyuan Lu, Shin-ya Takemura, Kenneth J Hayworth, Gary B Huang, Kazunori Shinomiya, Jeremy Maitlin-Shepard, Stuart Berg, et al. A connectome and analysis of the adult drosophila central brain. *elife*, 9:e57443, 2020.

[5] Eric R Kandel, James H Schwartz, Thomas M Jessell, Steven Siegelbaum, A James Hudspeth, Sarah Mack, et al. *Principles of neural science*, volume 4. McGraw-hill New York, 2000.

[6] Carsen Stringer, Marius Pachitariu, Nicholas Steinmetz, Matteo Carandini, and Kenneth D Harris. High-dimensional geometry of population responses in visual cortex. *Nature*, 571(7765):361–365, 2019.

[7] Jason Manley, Sihao Lu, Kevin Barber, Jeffrey Demas, Hyewon Kim, David Meyer, Francisca Martínez Traub, and Alipasha Vaziri. Simultaneous, cortex-wide dynamics of up to 1 million neurons reveal unbounded scaling of dimensionality with neuron number. *Neuron*, 2024.

[8] Anne E Urai, Brent Doiron, Andrew M Leifer, and Anne K Churchland. Large-scale neural recordings call for new insights to link brain and behavior. *Nature neuroscience*, 25(1):11–19, 2022.

[9] 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, 2013.

[10] John P Cunningham and Byron M Yu. Dimensionality reduction for large-scale neural recordings. *Nature neuroscience*, 17(11):1500–1509, 2014.

[11] Brian DePasquale, David Sussillo, LF Abbott, and Mark M Churchland. The centrality of population-level factors to network computation is demonstrated by a versatile approach for training spiking networks. *Neuron*, 111(5):631–649, 2023.

[12] Saurabh Vyas, Matthew D Golub, David Sussillo, and Krishna V Shenoy. Computation through neural population dynamics. *Annual review of neuroscience*, 43(1):249–275, 2020.

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

[14] 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.

[15] David Sussillo and Larry F Abbott. Generating coherent patterns of activity from chaotic neural networks. *Neuron*, 63(4):544–557, 2009.

[16] 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.

[17] Friedrich Schuessler, Francesca Mastrogiuseppe, Alexis Dubreuil, Srdjan Ostojic, and Omri Barak. The interplay between randomness and structure during learning in rnns. *Advances in neural information processing systems*, 33:13352–13362, 2020.

[18] Renate Krause, Matthew Cook, Sepp Kollmorgen, Valerio Mante, and Giacomo Indiveri. Operative dimensions in unconstrained connectivity of recurrent neural networks. *Advances in Neural Information Processing Systems*, 35:17073–17085, 2022.

[19] Alexis Dubreuil, Adrian Valente, Manuel Beiran, Francesca Mastrogiuseppe, and Srdjan Ostojic. The role of population structure in computations through neural dynamics. *Nature Neuroscience*, pages 1–12, 2022.

[20] Francesca Mastrogiuseppe and Srdjan Ostojic. Linking connectivity, dynamics, and computations in low-rank recurrent neural networks. *Neuron*, 99(3):609–623, 2018.

[21] Manuel Beiran, Alexis Dubreuil, Adrian Valente, Francesca Mastrogiuseppe, and Srdjan Ostojic. Shaping dynamics with multiple populations in low-rank recurrent networks. *Neural Computation*, 33(6):1572–1615, 2021.

[22] 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.

[23] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[24] Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. A survey on deep neural network pruning-taxonomy, comparison, analysis, and recommendations, 2024.

[25] Laura N Driscoll, Krishna Shenoy, and David Sussillo. Flexible multitask computation in recurrent networks utilizes shared dynamical motifs. *Nature Neuroscience*, 27(7):1349–1363, 2024.

[26] John T Bruer. *The myth of the first three years: A new understanding of early brain development and lifelong learning*. Simon and Schuster, 1999.

[27] Haim Sompolinsky, Andrea Crisanti, and Hans-Jurgen Sommers. Chaos in random neural networks. *Physical review letters*, 61(3):259, 1988.

[28] Rainer Engelken, Fred Wolf, and Larry F Abbott. Lyapunov spectra of chaotic recurrent neural networks. *Physical Review Research*, 5(4):043044, 2023.

[29] David G Clark, Owen Marschall, Alexander Van Meegen, and Ashok Litwin-Kumar. Connectivity structure and dynamics of nonlinear recurrent neural networks. *Physical Review X*, 15(4):041019, 2025.

[30] Yasaman Bahri, Jonathan Kadmon, Jeffrey Pennington, Sam S Schoenholz, Jascha Sohl-Dickstein, and Surya Ganguli. Statistical mechanics of deep learning. *Annual review of condensed matter physics*, 11(1):501–528, 2020.

[31] Louis Pezon, Valentin Schmutz, and Wulfram Gerstner. Linking neural manifolds to circuit structure in recurrent networks. *bioRxiv*, pages 2024–02, 2024.

[32] 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.

[33] Anushri Arora and Jonathan W Pillow. Efficient training of minimal and maximal low-rank recurrent neural networks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[34] Fatih Dinc, Ege Cirakman, Bariscan Kurtkaya, Mert Yuksekgonul, Yiqi Jiang, Mark Schnitzer, and Hidenori Tanaka. Ghost mechanism: An analytical model of abrupt learning in recurrent networks. *Phys. Rev. X*, pages –, Apr 2026.

[35] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr, 2013.

[36] Greg Yang and Edward J Hu. Tensor programs iv: Feature learning in infinite-width neural networks. In *International Conference on Machine Learning*, pages 11727–11737. PMLR, 2021.

[37] Yuhan Helena Liu, Aristide Baratin, Jonathan Cornford, Stefan Mihalas, Eric SheaBrown, and Guillaume Lajoie. How connectivity structure shapes rich and lazy learning in neural circuits.

In *International Conference on Learning Representations*, volume 2024, pages 5188–5202, 2024.

[38] Friedrich Schuessler, Francesca Mastrogiuseppe, Srdjan Ostojic, and Omri Barak. Aligned and oblique dynamics in recurrent neural networks. *Elife*, 13:RP93060, 2024.

[39] David G. Clark, Blake Bordelon, Jacob A. Zavatone-Veth, and Cengiz Pehlevan. Structure, disorder, and dynamics in task-trained recurrent neural circuits. *bioRxiv*, 2026.

[40] Madhu S Advani, Andrew M Saxe, and Haim Sompolinsky. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132:428–446, 2020.

[41] Laura Balzano, Tianjiao Ding, Benjamin D Haeffele, Soo Min Kwon, Qing Qu, Peng Wang, Zhangyang Wang, and Can Yaras. An overview of low-rank structures in the training and adaptation of large models. *arXiv preprint arXiv:2503.19859*, 2025.

[42] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2019.

[43] Flavio Martinelli, Johanni Brea, and Wulfram Gerstner. The puzzling success of overparameterization: Lottery tickets or escape dimensions?, May 2026. Preprint, EPFL Infoscience.

[44] 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.

[45] Peiran Gao, Eric Trautmann, Byron Yu, Gopal Santhanam, Stephen Ryu, Krishna Shenoy, and Surya Ganguli. A theory of multineuronal dimensionality, dynamics and measurement. *BioRxiv*, page 214262, 2017.

[46] A Aldo Faisal, Luc PJ Selen, and Daniel M Wolpert. Noise in the nervous system. *Nature reviews neuroscience*, 9(4):292–303, 2008.

[47] Laura N Driscoll, Lea Duncker, and Christopher D Harvey. Representational drift: Emerging theories for continual learning and experimental future directions. *Current Opinion in Neurobiology*, 76:102609, 2022.

[48] Steven H Strogatz. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. CRC press, 2018.

[49] Alan Edelman and N Raj Rao. Random matrix theory. *Acta numerica*, 14:233–297, 2005.

[50] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[51] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zach DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *31st Conference on Neural Information Processing Systems*, 2017.

S1 Theoretical details and proofs

S1.1 Spectrum of the rank-deficient weight matrix

The nonzero eigenvalues of $\boldsymbol{AB}$ and $\boldsymbol{BA}$ coincide, which can be seen by the determinant identity

$$\det(\lambda I_N - \beta \boldsymbol{AB}) = \lambda^{N-K} \det(\lambda I_K - \beta \boldsymbol{BA}). \tag{S1}$$

Thus, the recurrent spectrum consists of the eigenvalues of the $K \times K$ overlap matrix $\boldsymbol{O} = \beta \boldsymbol{BA}$, together with $N - K$ additional zeros. At the origin, the full neuronal Jacobian is then $(-I_N + \boldsymbol{W}^{\mathrm{rec}})/\tau$, which has $N - K$ globally decaying modes due to zero eigenvalues and K modes whose stability depend on whether $\mathrm{Re}(\lambda) > 1$ for the eigenvalues of the overlap matrix $\boldsymbol{O}$.

Origin instability under random initialization. The spectrum determines whether small perturbations grow near the origin. In the classical full-rank random-network limit, origin instability coincides with the onset of chaos [27]. At finite rank, however, an unstable origin can also lead to nonchaotic dynamics. We therefore assess origin instability separately from trajectory divergence, which is used to diagnose chaos [28, 48].

Proposition S1 (Origin instability under random initialization)**.** *Fix K and $g > 0$, and sample all entries independently with $A_{i\alpha} \sim \mathcal{N}(0, g^2/K)$ and $B_{\alpha i} \sim \mathcal{N}(0, 1)$. The latent Jacobian at the origin is*

$$\tau \boldsymbol{J}_0 = -\boldsymbol{I}_K + \boldsymbol{O}, \qquad \boldsymbol{O} = \beta(N) \boldsymbol{BA}. \tag{S2}$$

As $N \to \infty$, weak scaling gives $\boldsymbol{O} \to 0$ almost surely, so the origin is stable with probability tending to one. Under strong scaling, $\boldsymbol{O}$ instead converges in distribution to a matrix with independent $\mathcal{N}(0, g^2/K)$ entries. Its origin-instability probability therefore approaches that of a full-rank Gaussian RNN with K neurons. Taking $K \to \infty$ afterward, random-matrix theory gives stability with probability tending to one for $g < 1$ and instability with probability tending to one for $g > 1$.

Proof. Since $\tanh'(0) = 1$, linearizing the latent dynamics gives Eq. (S2). The origin is unstable when an eigenvalue of $\boldsymbol{O}$ has real part above one. Each overlap entry is

$$O_{\alpha\beta} = \beta(N) \sum_{i=1}^{N} B_{\alpha i} A_{i\beta}. \tag{S3}$$

The summands are independent across neurons and have zero mean. Under weak scaling $\beta(N) = N^{-1}$, the strong law of large numbers gives $\boldsymbol{O} \to 0$ almost surely at fixed K, so all eigenvalues approach zero.

Under strong scaling $\beta(N) = N^{-1/2}$, the multivariate central limit theorem applies jointly to the matrix entries. Their covariance follows from

$$\mathbb{E}[B_{\alpha i} A_{i\beta} B_{\gamma i} A_{i\delta}] = \frac{g^2}{K} \delta_{\alpha\gamma} \delta_{\beta\delta}. \tag{S4}$$

Thus, $\boldsymbol{O}$ converges in distribution to a Gaussian matrix $\boldsymbol{G}_{\mathrm{mat}}$ with independent entries of variance g^2/K. This is the real Ginibre ensemble [49], and is also the recurrent-matrix distribution of the full-rank Gaussian comparison with K neurons. For large K, the circular law distributes eigenvalues throughout the disk of radius g and $\max_\alpha \mathrm{Re}\, \lambda_\alpha(\boldsymbol{G}_{\mathrm{mat}}) \to g$ in probability. The origin is therefore stable with probability tending to one for $g < 1$ and unstable with probability tending to one for $g > 1$. □

This result is consistent with our observation in Figure S1 that the origin-stability distribution of a strongly initialized rank-K network with many neurons approaches that of a full-rank Gaussian network with K neurons. The instability boundaries in both ensembles approach $g = 1$ at large dimensionality, consistent with the classical random-network limit [27, 49].

S1.2 Proof of Proposition 1

To understand how population averaging changes the latent dynamics, we first isolate the contribution of each neuron. Write $\boldsymbol{a}_i = \boldsymbol{A}_{i:}^T$, $\boldsymbol{b}_i = \boldsymbol{B}_{:i}$, and define

$$\boldsymbol{X}_i(\boldsymbol{\kappa}) = \boldsymbol{b}_i \tanh(\boldsymbol{a}_i^T \boldsymbol{\kappa}), \qquad \tau\dot{\boldsymbol{\kappa}} + \boldsymbol{\kappa} = \beta(N) \sum_{i=1}^{N} \boldsymbol{X}_i(\boldsymbol{\kappa}). \tag{S5}$$

The pairs $(\boldsymbol{a}_i, \boldsymbol{b}_i)$ are independent and identically distributed across neurons, but their entries may be correlated within each pair. Since $|\tanh(x)| \leq 1$, the assumed second moments give

$$\mathbb{E}\|\boldsymbol{X}_i(\boldsymbol{\kappa})\|^2 \leq \mathbb{E}\|\boldsymbol{b}_i\|^2 < \infty. \tag{S6}$$

Thus, every neuronal contribution has a finite mean $\boldsymbol{F}(\boldsymbol{\kappa})$ and finite covariance, as required for the two convergence arguments below.

Weak synapses. With $\beta(N) = N^{-1}$, the recurrent field is a sample average. At each fixed latent state, the strong law of large numbers gives

$$\frac{1}{N} \sum_{i=1}^{N} \boldsymbol{X}_i(\boldsymbol{\kappa}) \longrightarrow \mathbb{E}[\boldsymbol{X}_i(\boldsymbol{\kappa})] = \boldsymbol{F}(\boldsymbol{\kappa}) \qquad \text{almost surely.} \tag{S7}$$

The fluctuations around this average also vanish in mean square. Independence across neurons makes the cross terms zero, so

$$\mathbb{E}\left\| \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{X}_i(\boldsymbol{\kappa}) - \boldsymbol{F}(\boldsymbol{\kappa}) \right\|^2 = \frac{1}{N} \mathbb{E}\|\boldsymbol{X}_i(\boldsymbol{\kappa}) - \boldsymbol{F}(\boldsymbol{\kappa})\|^2. \tag{S8}$$

The latent vector field as $N \to \infty$ is therefore $-\boldsymbol{\kappa} + \boldsymbol{F}(\boldsymbol{\kappa})$. Self-averaging does not require $\boldsymbol{F}$ to vanish and correlations between entries of $\boldsymbol{A}$ and $\boldsymbol{B}$ can retain a deterministic nonlinear recurrent contribution [19, 21].

Strong synapses. Now impose the additional independence and zero-mean assumptions in the proposition. At every fixed latent state,

$$\mathbb{E}[\boldsymbol{X}_i(\boldsymbol{\kappa})] = \mathbb{E}[\boldsymbol{b}_i]\, \mathbb{E}[\tanh(\boldsymbol{a}_i^T \boldsymbol{\kappa})] = 0. \tag{S9}$$

To characterize the field jointly across latent states, choose any finite collection $\boldsymbol{\kappa}^{(1)}, \ldots, \boldsymbol{\kappa}^{(m)}$ and concatenate their contributions into

$$\boldsymbol{Y}_i = \left(\boldsymbol{X}_i(\boldsymbol{\kappa}^{(1)})^T, \ldots, \boldsymbol{X}_i(\boldsymbol{\kappa}^{(m)})^T\right)^T \in \mathbb{R}^{Km}. \tag{S10}$$

These vectors are independent and identically distributed across neurons, have zero mean, and satisfy $\mathbb{E}\|\boldsymbol{Y}_i\|^2 \leq m\mathbb{E}\|\boldsymbol{b}_i\|^2 < \infty$. The multivariate central limit theorem then gives

$$\frac{1}{\sqrt{N}} \sum_{i=1}^{N} \boldsymbol{Y}_i \xrightarrow{d} \mathcal{N}\Big(0, \mathbb{E}[\boldsymbol{Y}_i \boldsymbol{Y}_i^T]\Big). \tag{S11}$$

The covariance between component α at state $\boldsymbol{\kappa}$ and component β at state $\boldsymbol{\kappa}'$ is precisely

$$\mathbb{E}\big[b_\alpha b_\beta \tanh(\boldsymbol{a}^T \boldsymbol{\kappa}) \tanh(\boldsymbol{a}^T \boldsymbol{\kappa}')\big]. \tag{S12}$$

This covariance is positive semidefinite because it is the covariance of the neuronal contributions. The Gaussian distributions are consistent across choices of latent states and therefore define the process $\boldsymbol{G}$ in Eq. (3). Adding the deterministic leak term gives the latent vector field $-\boldsymbol{\kappa} + \boldsymbol{G}(\boldsymbol{\kappa})$ as $N \to \infty$, with the joint convergence in distribution stated in the proposition.

For the independent Gaussian entries used in our strong initialization, $\boldsymbol{a} \sim \mathcal{N}(0, g^2 I_K/K)$ and $\boldsymbol{b} \sim \mathcal{N}(0, I_K)$. Independence and $\mathbb{E}[b_\alpha b_\beta] = \delta_{\alpha\beta}$ reduce the covariance to $\delta_{\alpha\beta} \mathbb{E}_{\boldsymbol{a}}[\tanh(\boldsymbol{a}^T \boldsymbol{\kappa}) \tanh(\boldsymbol{a}^T \boldsymbol{\kappa}')]$, as stated in the main text.

S1.3 Proof of Proposition 2

We consider the loss of neurons, together with all their incoming and outgoing synapses. The synaptic weight matrix is therefore modified as

$$\widetilde{W}_{ij}^{\mathrm{rec}} = \frac{P_i W_{ij}^{\mathrm{rec}} P_j}{1-p} = \frac{\beta(N)}{1-p} \sum_{k=1}^{K} P_i A_{ik} B_{kj} P_j, \tag{S13}$$

where $P_j \equiv P_{jj}$ are the diagonal entries of the matrix $\boldsymbol{P}$. Each entry is independently zero with probability p and one otherwise. Thus, on average, pN neurons are lost and $(1-p)N$ remain. We rescale the surviving connections to compensate for the mean loss of presynaptic contributions, since $\mathbb{E}_{\boldsymbol{P}}[P_j/(1-p)] = 1$. Throughout the proof, connectivity and the latent state are fixed, and expectations are over neuron removal only.

The unperturbed recurrent contribution to the latent dynamics is

$$g_i(\boldsymbol{\kappa}) = \beta(N) \sum_{j=1}^{N} B_{ij} \phi\left(\sum_{k=1}^{K} A_{jk} \kappa_k \right), \qquad i = 1, \ldots, K. \tag{S14}$$

For the perturbed neuronal activities $\widetilde{r}_j$, we define the latent variables as

$$\widetilde{\kappa}_i = \frac{\beta(N)}{1-p} \sum_{j=1}^{N} P_j B_{ij} \widetilde{r}_j. \tag{S15}$$

Projecting the perturbed neuronal dynamics onto these variables gives a recurrent contribution which, evaluated at the same latent coordinate $\boldsymbol{\kappa}$ as the unperturbed system, is

$$\widetilde{g}_i(\boldsymbol{\kappa}) = \frac{\beta(N)}{1-p} \sum_{j=1}^{N} P_j B_{ij} \phi\left(P_j \sum_{k=1}^{K} A_{jk} \kappa_k \right). \tag{S16}$$

Since P_j is either zero or one, $P_j \phi(P_j x) = P_j \phi(x)$. The perturbed contribution therefore simplifies to

$$\widetilde{g}_i(\boldsymbol{\kappa}) = \frac{\beta(N)}{1-p} \sum_{j=1}^{N} P_j B_{ij} \phi\left(\sum_{k=1}^{K} A_{jk} \kappa_k \right). \tag{S17}$$

The leak term remains $-\boldsymbol{\kappa}$ in both systems, so comparing $\boldsymbol{g}$ and $\widetilde{\boldsymbol{g}}$ also gives the difference between the full latent vector fields.

We now compare the perturbed and unperturbed contributions. Averaging over which neurons are removed, their difference vanishes:

$$\mathbb{E}_{\boldsymbol{P}}[\widetilde{g}_i(\boldsymbol{\kappa}) - g_i(\boldsymbol{\kappa})] = \beta(N) \sum_{j=1}^{N} B_{ij} \phi\left(\sum_{k=1}^{K} A_{jk} \kappa_k \right) \mathbb{E}_{\boldsymbol{P}}\left[\frac{P_j}{1-p} - 1 \right] = 0. \tag{S18}$$

The variance of this difference is

$$\mathbb{E}_{\boldsymbol{P}}\left[(\widetilde{g}_i(\boldsymbol{\kappa}) - g_i(\boldsymbol{\kappa}))^2 \right] = \beta(N)^2 \frac{p}{1-p} \sum_{j=1}^{N} \left[B_{ij} \phi\left(\sum_{k=1}^{K} A_{jk} \kappa_k \right) \right]^2. \tag{S19}$$

Here, independence of the masks removes cross terms between distinct neurons, while $\mathbb{E}_{\boldsymbol{P}}[(P_j/(1-p) - 1)^2] = p/(1-p)$. Summing over all K latent components gives the exact identity

$$\mathbb{E}_{\boldsymbol{P}}\left[\|\widetilde{\boldsymbol{g}}(\boldsymbol{\kappa}) - \boldsymbol{g}(\boldsymbol{\kappa})\|_2^2 \right] = \beta(N)^2 \frac{p}{1-p} \sum_{i=1}^{K} \sum_{j=1}^{N} \left[B_{ij} \phi\left(\sum_{k=1}^{K} A_{jk} \kappa_k \right) \right]^2. \tag{S20}$$

By the assumption in Proposition 2, the double sum divided by NK approaches $C(\boldsymbol{\kappa}) \in (0, \infty)$. Taking the square root therefore gives

$$\sqrt{\mathbb{E}_{\boldsymbol{P}}\left[\|\widetilde{\boldsymbol{g}}(\boldsymbol{\kappa}) - \boldsymbol{g}(\boldsymbol{\kappa})\|_2^2 \right]} \sim \sqrt{C(\boldsymbol{\kappa})}\, \beta(N) \sqrt{N} \sqrt{\frac{Kp}{1-p}}, \tag{S21}$$

which is Eq. (8). At fixed K and $0 < p < 1$, this deviation vanishes as $N^{-1/2}$ under weak scaling, $\beta(N) = N^{-1}$. Under strong scaling, $\beta(N) = N^{-1/2}$, it instead approaches $\sqrt{C(\boldsymbol{\kappa}) K p/(1-p)} > 0$. If $p = 0$, no neurons are lost and the deviation is exactly zero.

S2 Methods

S2.1 Origin instability and finite-time divergence

We measured origin instability and trajectory divergence separately, since loss of stability at the origin does not by itself imply chaos. For origin instability, we used the recurrent spectrum: a realization is unstable when $\max \mathrm{Re}\,\lambda(W) > 1$. Figure S1 compares the fraction of unstable realizations for fixed-size rank-deficient networks and full-rank networks with varying size.

Both sweeps testing edge of stability and edge of chaos use 100 independently sampled recurrent matrices per parameter setting. Their shared dimensionality grid is

$$\left\{ \begin{array}{c} 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 13, 15, 17, 19, \\ 22, 26, 29, 34, 39, 45, 52, 60, 69, 79, 91, 105, 121, 139, \\ 160, 184, 212, 244, 281, 324, 373, 429, 494, 569, 655, 754, 869, 1000 \end{array} \right\}. \tag{S22}$$

This specifies N in the full-rank ensemble and K in the rank-deficient ensemble, whose population size remains $N = 1000$. Origin-instability sweeps use 20 equally spaced gains from 0 to 5; divergence sweeps use 20 equally spaced gains from 0 to 20. All recurrent entries have variance g^2/N, using dense Gaussian matrices for the full-rank comparison and independent Gaussian entries of $\boldsymbol{A}$ and $\boldsymbol{B}$ for the rank-deficient comparison.

Assessment of chaos signatures: We tested whether a small perturbation grows as a network evolves. For each sampled recurrent matrix, we simulated $\tau \dot{\boldsymbol{r}} = -\boldsymbol{r} + \tanh(\boldsymbol{W}\boldsymbol{r})$ from 200 initial conditions, drawing each neuronal activity independently from $\mathcal{N}(0, 0.1^2)$. Time was measured in units of τ, and simulations used forward Euler integration with step 0.1 and single-precision arithmetic. We first evolved each initial condition for 100 time units to allow initial transients to decay. We then made a copy of the network state and added a random perturbation with Euclidean norm 10^{-7}. The original and perturbed states evolved under the same connectivity for another 200 time units.

For each trajectory pair, we measured their separation, $d(t) = \|\boldsymbol{r}_{\mathrm{perturbed}}(t) - \boldsymbol{r}_{\mathrm{original}}(t)\|_2$. Exponential growth of this distance appears as a positive slope when plotting $\log d(t)$ against time. We therefore fitted a straight line to this relationship. To reduce the influence of the immediate response to the perturbation and the eventual saturation of the separation, we excluded the first 10 time units and retained only positive distances below half the maximum distance reached by that pair. A fit was retained only if at least two time points remained and its coefficient of determination satisfied $R^2 > 0$.

We then combined the fitted slopes across initial conditions to classify each network. At least two valid slopes were required. A network passed the positive-divergence criterion only if both of the following held: a one-sided one-sample t-test supported a mean slope greater than 10^{-8}, and the fifth percentile of the slopes also exceeded 10^{-8}. The second condition requires positive divergence across most tested initial conditions, rather than allowing a few large slopes to dominate the mean. We corrected the test for the 100 networks sampled at each parameter setting by multiplying its p-value by 100 and requiring the corrected value to be below 0.05. The plotted fraction is the number of networks passing both conditions divided by 100. Networks with insufficient valid fits remain in this denominator but do not pass the criterion.

Estimating the boundaries: In Figs. 3 and S1, we summarize each sweep by the point at which half of the sampled networks first meet the corresponding criterion: positive trajectory divergence or origin instability, respectively. We locate this point by interpolation between neighboring samples, using a linear scale for gain and a logarithmic scale for dimensionality (K for rank-deficient networks and N for full-rank networks). The bars span the two sampled values surrounding the crossing. They therefore indicate the resolution of the parameter grid. If a sweep does not cross 0.5, no boundary is assigned.

These measurements provide finite-time numerical signatures of chaos rather than proof of asymptotic chaos in the continuous-time system. Their interpretation depends on the integration step, simulation duration, and selection of points used for fitting. For example, no rank-one network passes the criterion, but up to 2% of rank-two networks pass at some gains. A full theoretical proof of chaos remains open for low-rank RNNs.

S2.2 Task-driven network models

Rate networks: For the task experiments, we added inputs and biases to the autonomous dynamics and discretized them using the Euler update

$$\boldsymbol{r}_{t+1} = (1-\alpha)\boldsymbol{r}_t + \alpha\phi(\boldsymbol{W}^{\mathrm{rec}}\boldsymbol{r}_t + \boldsymbol{W}_{\mathrm{in}}\boldsymbol{x}_t + \boldsymbol{b}_{\mathrm{in}})\,, \qquad \alpha = \Delta t/\tau = 0.5, \tag{S23}$$

where $\boldsymbol{r}_t$ is the neuronal activity and $\boldsymbol{x}_t$ is the task input at time step t. We used $\Delta t = 5$ ms and $\tau = 10$ ms. The activation ϕ acts elementwise and is $\tanh$ except in the ReLU control, where $\phi(x) = \max(0, x)$. Input and readout weights used Xavier-uniform initialization [50]. Unless stated otherwise, the corresponding biases were sampled uniformly from $[-1/\sqrt{n_{\mathrm{in}}}, 1/\sqrt{n_{\mathrm{in}}}]$, where n_{in} is the number of inputs to the corresponding layer.

Initial activities depended on the task. For working-memory and flip-flop tasks, we sampled independent standard Gaussian values at the start of each forward pass and applied the network's activation function. Sequential-MNIST networks started from zero activity. Navigation networks instead used a trainable initial-state vector, initialized with independent standard Gaussian entries and passed through $\tanh$. This initial state was shared across trajectories.

Rate-network readouts: The readout maps neuronal activity to the task response. DCD used a $\tanh$ output, DMTS a sigmoid output, and navigation and flip-flop linear outputs. Sequential MNIST used the updated state after the final pixel to produce ten classification logits. We trained all recurrent, input, readout, and bias parameters. Navigation also trained the initial-state vector.

Gated networks: To examine whether the observations extend to architectures with gates, we used gated recurrent units (GRUs) and long short-term memory networks (LSTMs). Both started with zero hidden states, and LSTMs also started with zero cell states.

In a GRU, the reset gate $\boldsymbol{a}_t$ determines how the previous hidden state contributes to the candidate state $\widehat{\boldsymbol{h}}_t$. The update gate $\boldsymbol{z}_t$ then combines this candidate with the previous state:

$$\boldsymbol{a}_t = \sigma(\boldsymbol{W}_{ia}\boldsymbol{x}_t + \boldsymbol{b}_{ia} + \boldsymbol{W}_{ha}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{ha}), \tag{S24}$$

$$\boldsymbol{z}_t = \sigma(\boldsymbol{W}_{iz}\boldsymbol{x}_t + \boldsymbol{b}_{iz} + \boldsymbol{W}_{hz}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{hz}), \tag{S25}$$

$$\widehat{\boldsymbol{h}}_t = \tanh(\boldsymbol{W}_{in}\boldsymbol{x}_t + \boldsymbol{b}_{in} + \boldsymbol{a}_t \odot (\boldsymbol{W}_{hn}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{hn}))\,, \tag{S26}$$

$$\boldsymbol{h}_t = (1-\boldsymbol{z}_t) \odot \widehat{\boldsymbol{h}}_t + \boldsymbol{z}_t \odot \boldsymbol{h}_{t-1}. \tag{S27}$$

Here, σ is the logistic sigmoid and $\odot$ denotes elementwise multiplication. Expressions such as $1-\boldsymbol{z}_t$ are also evaluated elementwise.

An LSTM additionally maintains a cell state $\boldsymbol{c}_t$. Its forget gate $\boldsymbol{f}_t$ determines how much of the previous cell state is retained, its input gate $\boldsymbol{i}_t$ determines how much new information is added, and its output gate $\boldsymbol{o}_t$ determines how the cell state contributes to the hidden state:

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_{ii}\boldsymbol{x}_t + \boldsymbol{b}_{ii} + \boldsymbol{W}_{hi}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{hi}), \tag{S28}$$

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_{if}\boldsymbol{x}_t + \boldsymbol{b}_{if} + \boldsymbol{W}_{hf}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{hf}), \tag{S29}$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_{io}\boldsymbol{x}_t + \boldsymbol{b}_{io} + \boldsymbol{W}_{ho}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{ho}), \tag{S30}$$

$$\widehat{\boldsymbol{c}}_t = \tanh(\boldsymbol{W}_{ic}\boldsymbol{x}_t + \boldsymbol{b}_{ic} + \boldsymbol{W}_{hc}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{hc}), \tag{S31}$$

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \widehat{\boldsymbol{c}}_t, \qquad \boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t). \tag{S32}$$

In both architectures, matrices with first subscript i act on the input, while those with first subscript h act on the previous hidden state. The remaining subscript identifies the gate or candidate state.

Input weights and biases in the gated cells were sampled uniformly from $[-1/\sqrt{N}, 1/\sqrt{N}]$. The rank-sweep experiments replaced only the recurrent weights at initialization, while all entries remained trainable. Readout weights used Xavier-uniform initialization with zero bias. Readouts acted on the hidden state after each gated update, using $\tanh$ for DCD, sigmoid for DMTS, a linear output for flip-flop, and final-step logits for MNIST.

Spiking networks: For the spiking comparison, we used a discrete-time leaky integrate-and-fire model. The vectors $\boldsymbol{u}_t$ and $\boldsymbol{s}_t$ collect the membrane potentials and binary spikes across neurons:

$$\boldsymbol{u}_t = 0.9\,\boldsymbol{u}_{t-1} \odot (1-\boldsymbol{s}_{t-1}) + \boldsymbol{W}^{\mathrm{rec}}\boldsymbol{s}_{t-1} + \boldsymbol{W}_{\mathrm{in}}\boldsymbol{x}_t + \boldsymbol{b}_{\mathrm{in}}, \tag{S33}$$

$$\boldsymbol{s}_t = \mathbb{I}[\boldsymbol{u}_t \geq 1], \qquad \boldsymbol{q}_t = 0.8\,\boldsymbol{q}_{t-1} + \boldsymbol{s}_t. \tag{S34}$$

Each neuron emits a spike when its membrane potential reaches one. The coefficient 0.9 produces membrane decay, while $(1 - \boldsymbol{s}_{t-1})$ removes the previous membrane contribution following a spike. The trace $\boldsymbol{q}_t$ accumulates spikes with decay coefficient 0.8, providing a temporally filtered signal for the readout. Readouts used this updated trace and the same task-specific output nonlinearities as the gated models.

Each trial began with zero membrane potentials and spike traces, with each initial spike independently set to one with probability 0.1. Since the spike threshold is not differentiable, backpropagation used the surrogate derivative $[1 + 10|u - 1|]^{-2}$ for each neuron's membrane potential u. Recurrent weights used the strong initialization $\boldsymbol{W}^{\mathrm{rec}} = \beta(N)\boldsymbol{A}\boldsymbol{B}$ with $K = N$ and gain $g = 2$. Input and readout weights used Xavier-uniform initialization. Input biases were zero, and readout biases were sampled uniformly from $[-1/\sqrt{N}, 1/\sqrt{N}]$.

S2.3 Tasks

Delayed cue discrimination (DCD): The network must retain the identity of a cue through a delay and report it during a designated response period. A cue of $+1$ or -1 was presented for six steps (30 ms), followed by ten zero-input delay steps (50 ms) and ten response steps (50 ms). The target output was zero during the cue and delay and equaled the cue sign during the response.

Each training batch contained ten copies of each cue condition. The input sequences remained fixed throughout training, while newly sampled initial activities provided trial-to-trial variability in the rate networks. We computed mean-squared error (MSE) over the complete batch, all time steps, and all output entries. To measure response accuracy, we converted outputs below -0.33 to -1, outputs above 0.33 to $+1$, and intermediate outputs to zero, then measured the fraction matching the target during the response period.

Delayed match-to-sample (DMTS): The network must report whether two cues agree. Each cue was $+1$ or -1 and lasted six steps. The cues were separated by a ten-step delay, and the second cue was followed by ten response steps. During the response, the target was one for matching cues and zero otherwise. At all earlier times, the target was zero.

Each batch contained ten copies of each of the four possible cue pairs. We computed MSE over the whole sequence. Response accuracy was measured during the response period by thresholding the sigmoid output at 0.5 and comparing the resulting binary prediction with the target.

Three-bit flip-flop: The network receives sparse signed pulses on three input channels. Each corresponding output must retain the sign of its most recent input pulse until another pulse arrives.

For each run, we generated a fixed batch of 100 sequences, each lasting 100 steps. At each step, a candidate pulse occurred independently on each channel with probability 0.05. A candidate was suppressed if that channel had already received an accepted pulse within the preceding four steps. The first step supplied an independent sign on every channel so that all target outputs were defined from the start. Targets changed immediately when a pulse arrived, and MSE was computed over the batch, all time steps, and all three channels.

Sequential MNIST: To turn image classification into a sequential task, we resized MNIST images to 7×7 using bilinear interpolation, scaled their pixel values to $[0, 1]$, and presented the 49 pixels one at a time in row-major order. After receiving the final pixel, the network produced ten logits and was trained to identify the digit using cross-entropy loss.

Each training iteration used 200 images sampled with replacement from the training set. Test accuracy was measured every ten iterations using 100 images sampled with replacement from the test set, at one-based epochs $1, 11, 21, \ldots$. The recorded accuracies therefore estimate performance on sampled test images rather than the entire test set. Training used global gradient-norm clipping at 100.

One-dimensional navigation: The network must integrate signed velocity to represent position on a periodic track of circumference $2L$, with $L = 1$. Each run used 64 fixed trajectories of 100 steps, all starting at position zero. Initial velocity was drawn from $\mathcal{N}(0, 4^2)$ and subsequently followed a discretized Ornstein–Uhlenbeck process:

$$v_{t+1} = v_t - \frac{\Delta t}{\tau_v} v_t + 4\sqrt{\frac{2\Delta t}{\tau_v}}\,\xi_t, \qquad \tau_v = 0.1, \quad \Delta t = 0.005, \quad \xi_t \sim \mathcal{N}(0, 1). \tag{S35}$$

Position advanced by $v_{t+1}\Delta t$ at each step and wrapped into $[-L, L)$.

The target output represented position through the activities of 100 place cells with evenly spaced centers. Each cell had an unnormalized Gaussian tuning curve with width $L/50$, evaluated using circular distance. We trained the linear readout by MSE over all trajectories, cells, and time steps. To measure position accuracy, we selected the most active predicted place cell and asked whether its center lay within circular distance $L/10$ of the true position. Accuracy was the fraction of samples meeting this criterion on the training trajectories. Training used global gradient-norm clipping at 1.

S2.4 Training protocols and learning-speed analysis

Optimization and epoch convention: We implemented the models using single-precision tensors in PyTorch [51]. Training used backpropagation through time and Adam with momentum parameters $(0.9, 0.999)$, numerical stabilizer 10^{-8}, and no weight decay. Where gradient clipping was used, gradients across all trainable parameters were rescaled together if their global Euclidean norm exceeded the stated threshold.

An epoch denotes one parameter-update iteration on the task batch. For MNIST, this means one sampled minibatch rather than a complete pass through the dataset. Loss and training accuracy were recorded from the forward pass before the parameter update. Epoch numbering starts at one throughout.

Working-memory rank sweeps: To compare learning across initialization ranks, we trained DCD and DMTS networks with $N = 1000$, gains $g \in \{0.5, 1, 1.5\}$, 20 seeds per condition, and the 22 ranks

$$K \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 30, 50, 100, 150, 200, 250, 300, 500, 700, 1000\}. \quad \text{(S36)}$$

Both tasks used 10000 epochs, learning rate 3×10^{-5}, and no gradient clipping (Fig. 4). Zero-initialization references came from the corresponding $N = 1000$ experiments.

Figures 1**c** and S3 use the $g = 1$ learning trajectories, with $K = 10$ and the full-rank endpoint shown in Fig. 1**c**. The structured DCD sweeps in Fig. S2 used the same epoch budget, learning rate, and 20 seeds, with gains $0.5, 1, 1.5, 2, 2.5, 3$.

Learning-rate controls: To examine whether the learning-speed comparisons depended on optimization choices, we swept learning rates on both working-memory tasks (Fig. S4). We compared rank-10 strong, rank-tempered, orthogonal, and correlated initializations with full-rank random and zero-recurrent references. Each condition used ten seeds and ten logarithmically spaced learning rates from 10^{-5} to 10^{-3}. Nonzero initializations were tested at gains $0.5, 1, 1.5$; zero initialization was not repeated over gains. Matched sweeps either used no clipping or clipped the global gradient norm at 1. All runs had a 10000-epoch budget.

We then repeated the DCD rank sweep with selected learning rates held fixed across ranks (Fig. S5). Rank-tempered, orthogonal, and correlated initializations shared a learning rate of approximately 2.154×10^{-4}. In the clipped DCD sweeps at $K = 10$, this rate attained the highest ten-seed mean speed for orthogonal and correlated initialization when comparing the tested rates and gains $0.5, 1, 1.5$. This rate was also near-optimal for rank-tempered initialization at $g = 1$. Strong and zero initializations used approximately 1.292×10^{-4} and 3.594×10^{-4}, respectively.

These rates were not retuned at each rank. The selected-rate sweep used 20 seeds and clipping at 1, with $g = 1$ for strong and rank-tempered initialization and six gains from 0.5 to 3 for the structured conditions.

Additional tasks: Navigation rank sweeps used the same 22 ranks, $N = 1000$, $g = 1$, 20 seeds, 10000 epochs, learning rate 3×10^{-5}, and clipping at 1. Sequential-MNIST rank sweeps used 20 seeds, gains $0.1, 0.3, 0.5, 0.7, 0.9, 1$, 10000 iterations, learning rate 3×10^{-4}, and clipping at 100. The MNIST zero-initialization reference reused the $N = 1000$ scaling runs. Although those runs had a longer training budget, we restricted their learning-speed analysis to the first 10000 iterations.

The learning-rate controls in Fig. S6 used ten seeds, $K = 10$ for rank-deficient conditions, and corresponding full-rank and zero-initialization references. Navigation used 19 logarithmically spaced rates from 10^{-5} to 10^{-1}; MNIST used ten rates from 10^{-5} to 10^{-3}. The displayed gains were $0.5, 1, 1.5$, and each task retained its clipping threshold.

Gated-network rank sweeps: We trained GRUs and LSTMs on DCD with $N = 1000$, the same 22 ranks, gains $0.5, 1, 1.5$, and ten seeds per condition (Fig. S8). Each run had a 20000-epoch budget.

Learning rates were 10^{-5} for GRUs and 3×10^{-5} for LSTMs, without clipping. Default-initialization and zero-recurrent references also used ten seeds each.

Measuring learning speed: We required sustained performance rather than a single successful epoch. For working-memory and gated-network comparisons, a run qualified when its MSE remained strictly below 0.05 for 50 consecutive epochs. We recorded the one-based epoch e_* ending the first such window and defined learning speed as $1/e_*$. Figures display $1000/e_*$, expressing speed in units of 10^{-3} epoch^{-1}. Once a run qualified, later increases in loss did not change its recorded learning speed.

Navigation used the same 50-epoch rule, requiring position accuracy strictly greater than 90% throughout the window. MNIST required six consecutive recorded test accuracies strictly greater than 90%. Because test evaluations were separated by ten iterations, this window spanned 50 training iterations, and e_* was the epoch of its sixth evaluation. With 100 sampled test images, each evaluation required at least 91 correct predictions. We applied these criteria directly to the recorded metrics, without rounding or smoothing.

Failures and summaries across runs: A completed run that never met the criterion received zero speed and remained included in the mean and standard error. Documented numerical aborts in the navigation learning-rate sweep also received zero speed. Missing or incomplete results without a documented stopping or abort condition were not silently treated as failed runs.

We computed mean speed across seeds, with standard error $s/\sqrt{n}$, where s is the sample standard deviation and n is the number of runs. Learning-trajectory plots instead show the median and interquartile range across seeds, including runs that did not converge.

S2.5 Learned spectra and effective dimensionality

Final recurrent spectra: To examine what remains of the initial low-rank structure after unrestricted training, we analyzed final recurrent matrices from the $N = 1000$ rank sweeps (Fig. S7). We used $g = 1$ for DCD, DMTS, navigation, GRU-DCD, and LSTM-DCD, and $g = 0.5$ for sequential MNIST. Each vanilla condition had 20 candidate seeds and each gated condition had ten. Only networks meeting the final-window success criteria in Appendix S2.6 entered this analysis.

We sorted the singular values within each network before averaging them across successful seeds. For gated networks, we analyzed the full stacked recurrent matrix, rather than treating the gates separately.

Effective dimensionality: We quantified how concentrated each spectrum was by counting the singular values needed to account for 99% of their squared sum. For ordered singular values $\sigma_1 \geq \sigma_2 \geq \cdots$, we defined

$$r_{99} = \min \left\{ r : \frac{\sum_{j=1}^{r} \sigma_j^2}{\sum_j \sigma_j^2} \geq 0.99 \right\}. \tag{S37}$$

The denominator equals the squared Frobenius norm of the recurrent matrix, without centering its entries. A small r_{99} therefore indicates that a few singular directions account for most of this norm; it does not imply that all remaining singular values are exactly zero.

For comparison, initialization guides show the approximate nonzero singular-value scales $g\sqrt{qN/K}$ for strong initialization and $g\sqrt{q}$ for rank-tempered initialization. Here $q = 1, 3, 4$ for vanilla, GRU, and LSTM recurrent matrices, respectively. These expressions describe the $K \ll N$ regime and are approximate guides, rather than singular values measured from saved initial matrices.

Participation ratio of learned updates: We separately measured the dimensionality of the changes acquired during learning. Table S1 uses the full-rank strong-initialization runs from Fig. 5, with the same successful-run selections. For each network, we computed

$$\Delta \boldsymbol{W}^{\mathrm{rec}} = \boldsymbol{W}^{\mathrm{rec}}_{\mathrm{final}} - \boldsymbol{W}^{\mathrm{rec}}_{\mathrm{initial}}. \tag{S38}$$

For singular values s_i of this update, we defined

$$p_i = \frac{s_i^2}{\sum_j s_j^2}, \qquad \mathrm{PR}(\Delta \boldsymbol{W}^{\mathrm{rec}}) = \frac{1}{\sum_i p_i^2}. \tag{S39}$$

The participation ratio describes how many singular directions contribute appreciably to the squared Frobenius norm. For example, d equal nonzero singular values give a participation ratio of d. We computed this quantity separately for each network and then reported its mean and SEM across successful seeds at each network size.

This analysis concerns learned updates, whereas r_{99} in Fig. S7 concerns the final recurrent matrix.

S2.6 Weight and update scaling

Network sizes and baseline training: To measure how synaptic strength depends on population size, we trained networks with $N \in \{50, 100, 150, 200, 250, 300, 500, 700, 1000\}$, using 20 seeds per condition and size. MNIST used the seven sizes with $N \geq 150$.

Figure 5 compares zero and dense strong initialization. DCD and DMTS used 10000 epochs and learning rate $3 \times 10^{-5}(1000/N)$. Flip-flop used 20000 epochs and $3 \times 10^{-4}(1000/N)$ for both initializations. MNIST used up to 100000 iterations and learning rate $10^{-3}(300/N)$. MNIST training could stop early when the mean of the last ten sampled test accuracies exceeded 95% and the current loss was below 0.01.

Learning-rate and architecture controls: To test whether update scaling depended on the learning-rate schedule, Fig. S10 used strong initialization with learning rates proportional to $N^{-1/2}$. DCD, DMTS, and flip-flop used 50000 epochs at $3 \times 10^{-5}\sqrt{1000/N}$; MNIST used up to 50000 iterations at $10^{-4}\sqrt{1000/N}$. The ReLU control in Fig. S11 instead retained the task-specific budgets and $1/N$ learning-rate schedules of Fig. 5.

GRU and LSTM scaling experiments used their default recurrent initialization (Figs. S12 and S13). DCD, DMTS, and flip-flop were trained for 20000 epochs, and MNIST for up to 100000 iterations. Learning rates were $3 \times 10^{-5}(1000/N)$ for DCD and DMTS, $3 \times 10^{-4}(1000/N)$ for flip-flop, and $10^{-3}(300/N)$ for MNIST. The spiking comparison used these same budgets and schedules, with the initialization described in Appendix S2.2 (Fig. S14).

These MNIST scaling controls used the same accuracy-and-loss early-stopping criterion described above. Gated and spiking models additionally required at least ten test evaluations before stopping. All MNIST scaling experiments used clipping at 100; the other scaling tasks were unclipped.

Selecting successfully trained networks: The scaling analysis asks how connectivity depends on size among networks that successfully perform the task. We therefore required successful performance throughout a final evaluation window, rather than merely at some earlier point during training.

For DCD, DMTS, and flip-flop, MSE had to remain strictly below 0.05 throughout the final 50 epochs. Navigation required position accuracy strictly greater than 90% throughout the final 50 epochs. MNIST required sampled test accuracy strictly greater than 90% at each of the final six evaluations. The spiking-MNIST control used a lower threshold, strictly greater than 70%, at each of those six evaluations. Runs that stopped early under the stated stopping rule were evaluated at their saved endpoints.

Within each condition, we used the same successful networks for all weight quantities, gates, and measures of synaptic strength. This selection also applied to the learned-spectrum analysis. It differs from the learning-speed analysis, where unsuccessful runs remain included with zero speed.

Measuring synaptic strength: We used two summaries of matrix entries to check whether the scaling depended on how synaptic strength was measured. For a matrix $\boldsymbol{X}$ containing m entries, these were the mean absolute entry and the root-mean-square (RMS) entry:

$$\frac{1}{m}\sum_{i,j}|X_{ij}|, \qquad \left(\frac{1}{m}\sum_{i,j}X_{ij}^2\right)^{1/2}. \tag{S40}$$

Here $\boldsymbol{X}$ is either the final recurrent matrix or its learned update. Updates were computed by subtracting the initial matrix from the final matrix of the same run. For gate-specific analyses, we split the recurrent matrix into its $N \times N$ gate blocks before computing either summary.

We first measured synaptic strength within each network, then averaged these values across successful seeds at each size and plotted their standard errors. Figure 5 shows mean absolute entries, and Fig. S9 shows the corresponding RMS measurements. The subsequent scaling supplements report both.

Fitting scaling exponents: A power law proportional to $N^{-\alpha}$ becomes a straight line with slope $-\alpha$ in logarithmic coordinates. We therefore fitted $\log_{10}$ mean synaptic strength against $\log_{10} N$ using unweighted ordinary least squares and all available sizes in each comparison. The mean at each size was taken across successful seeds before fitting. Reported slope uncertainties are 1.96 times the regression standard error.

S2.7 Recurrent dropout

Dropout during training: To test whether perturbations during learning change update scaling, we introduced recurrent dropout. Before computing the recurrent input, we independently set each neuronal activity to zero with probability p and divided each retained activity by $1 - p$. This compensation preserves the mean activity entering the recurrent multiplication when averaging over dropout masks.

Masks were sampled afresh for each trial, neuron, and time step. The leak term and readout were not directly masked. Thus, a neuron could contribute at one step and be masked at another, unlike the permanent neuron removal considered in Proposition 2.

Network sizes and training protocols: Figures S15 and S16 use six network sizes, $N = 500, 600, 700, 800, 900, 1000$, and six dropout probabilities, $p = 0, 0.1, 0.2, 0.3, 0.4, 0.5$, with 20 seeds per task, size, and probability. All networks began with dense Gaussian recurrent weights of variance $1/N$.

DCD and DMTS used 50000 epochs at learning rate $3 \times 10^{-5}\sqrt{1000/N}$. Flip-flop used the same learning rate with a maximum of 200000 epochs, stopping when both the current MSE and the mean of the last ten losses were below 0.01. MNIST used 50000 iterations at $10^{-4}\sqrt{1000/N}$ and clipping at 100.

Evaluation and scaling fits: Dropout remained active during the recorded forward passes, including sampled MNIST test evaluations. The final-window success criteria therefore assess performance under dropout. Both figures used identical successful-run selections and all six network sizes for each exponent fit. Figure S15 plots the fitted exponents against dropout probability, while Fig. S16 shows the underlying dependence of update strength on network size.

Each dropout probability had 120 candidate networks per task: six sizes with 20 seeds each. At increasing probabilities $p = 0, 0.1, 0.2, 0.3, 0.4, 0.5$, DCD retained $117, 120, 119, 120, 120, 120$ successful networks, respectively. DMTS and flip-flop retained all 120 at every probability. Sequential MNIST retained $120, 120, 119, 108, 97, 43$.

The high-dropout MNIST fits therefore describe a smaller subset of networks that continued to perform the task. In contrast, the flip-flop trend includes every trained network.

S3 Supplementary Tables

Table S1: **Participation ratios of learned recurrent updates across network sizes.** Full-rank strong-initialization runs from Fig. 5. For $\Delta W = W_{\mathrm{final}} - W_{\mathrm{initial}}$ with singular values s_i, $\mathrm{PR}(\Delta W) = (\sum_i s_i^2)^2 / \sum_i s_i^4$. Entries give mean $\pm$ SEM across successful seeds using the same final-window selection as Fig. 5. All cells use 20 seeds except DCD at $N = 700$ and flip-flop at $N = 150$ (19 each), and sequential MNIST at $N = 200, 500, 700, 1000$ (19,18,18,8). Dashes indicate untested sizes.

N	DCD	DMTS	Flip-flop	Seq-MNIST
50	4.76 ± 0.09	3.43 ± 0.05	10.71 ± 0.41	—
100	5.49 ± 0.14	3.39 ± 0.05	15.35 ± 0.55	—
150	5.64 ± 0.07	3.41 ± 0.05	17.89 ± 0.67	26.51 ± 0.53
200	5.77 ± 0.07	3.50 ± 0.05	18.64 ± 1.12	24.08 ± 0.42
250	5.78 ± 0.06	3.66 ± 0.06	20.02 ± 1.09	22.25 ± 0.31
300	5.91 ± 0.08	3.65 ± 0.05	20.13 ± 1.15	22.14 ± 0.37
500	5.69 ± 0.12	4.10 ± 0.06	24.60 ± 1.17	22.08 ± 0.51
700	5.57 ± 0.07	4.59 ± 0.05	22.46 ± 1.44	21.12 ± 0.42
1000	5.69 ± 0.07	4.85 ± 0.04	21.59 ± 1.57	19.75 ± 1.81

S4 Supplementary Figures

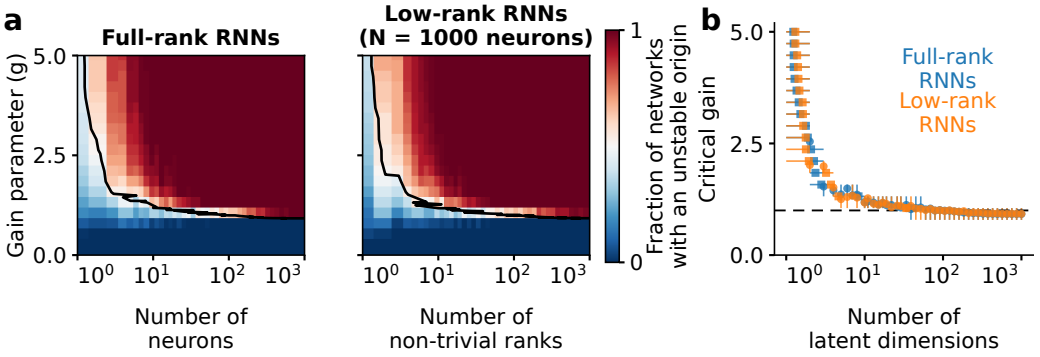

Figure S1: **Full-rank and low-rank RNNs show similar instability boundaries.** **a** Fraction of networks with an unstable origin, $\max \mathrm{Re}\,\lambda(W^{\mathrm{rec}}) > 1$, versus gain g and latent dimensionality. *Left.* Full-rank RNNs with N neurons. *Right.* Low-rank RNNs with $N = 1000$ neurons and rank K. Black curves mark the interpolated fraction-0.5 boundary. Both ensembles have $\mathrm{Var}[W_{ij}^{\mathrm{rec}}] = g^2/N$. **b** Critical gain for full-rank (blue) and low-rank (orange) networks. Error bars span neighboring sampled gains or ranks bracketing the 0.5 crossing. The dashed line marks $g = 1$.

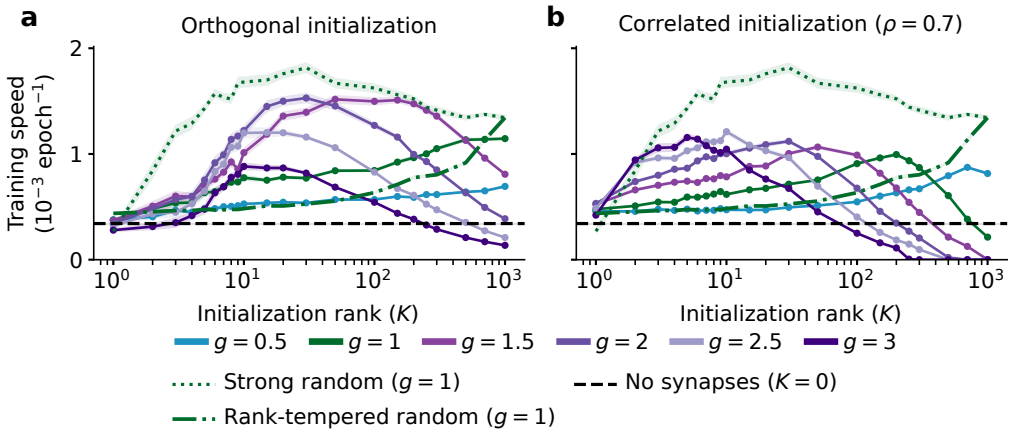

Figure S2: **Structured weak initialization can accelerate learning at low ranks.** Training speed versus rank K for orthogonal (**a**) and correlated (**b**) weak initialization on delayed cue discrimination ($N = 1{,}000$; solid lines). In **b**, the two vectors defining each rank-one component have correlation $\rho = 0.7$. Colors indicate gain g. Green dotted and dash-dot curves show strong random and rank-tempered random initialization at $g = 1$, respectively; black dashed lines mark zero initialization. Speed is the reciprocal of the final epoch of the first 50 consecutive epochs with training MSE below 0.05. Runs without such a window within 10,000 epochs receive zero speed. Curves and shading show mean $\pm$ SEM across all 20 seeds. All networks use a learning rate of 3×10^{-5} without gradient clipping.

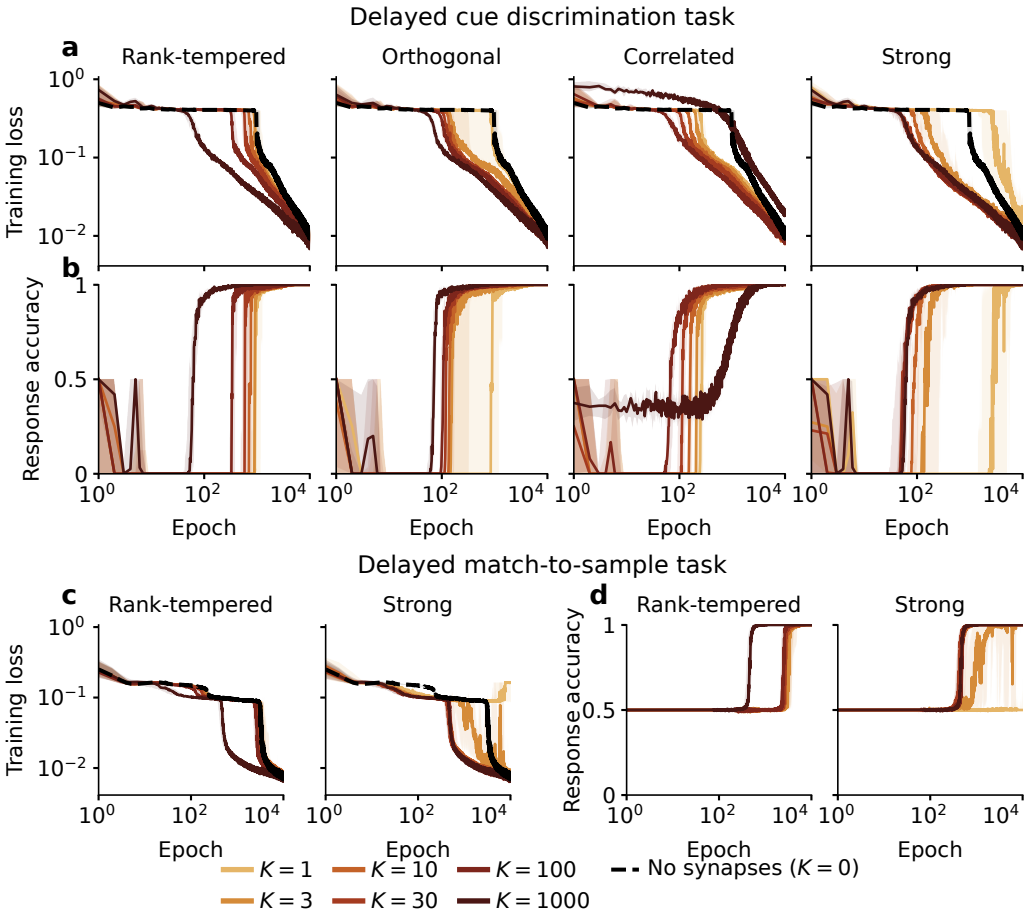

Figure S3: **Learning trajectories of distinct initialization methods in working-memory tasks.** Training loss and response accuracy for delayed cue discrimination (**a-b**) and delayed match-to-sample (**c-d**). Colors indicate initialization rank K. All networks use $N = 1{,}000$ and $g = 1$. Correlated initialization uses $\rho = 0.7$ between corresponding entries of the two low-rank matrices. Curves and shading show the median and interquartile range across 20 seeds. Black dashed curves in the loss plots show zero recurrent initialization. Both tasks use a learning rate of 3×10^{-5} without gradient clipping. With strong synapses, learning trajectories largely overlap beyond $K \sim 10 - 30$, whereas rank dependence remains pronounced for the other initialization methods.

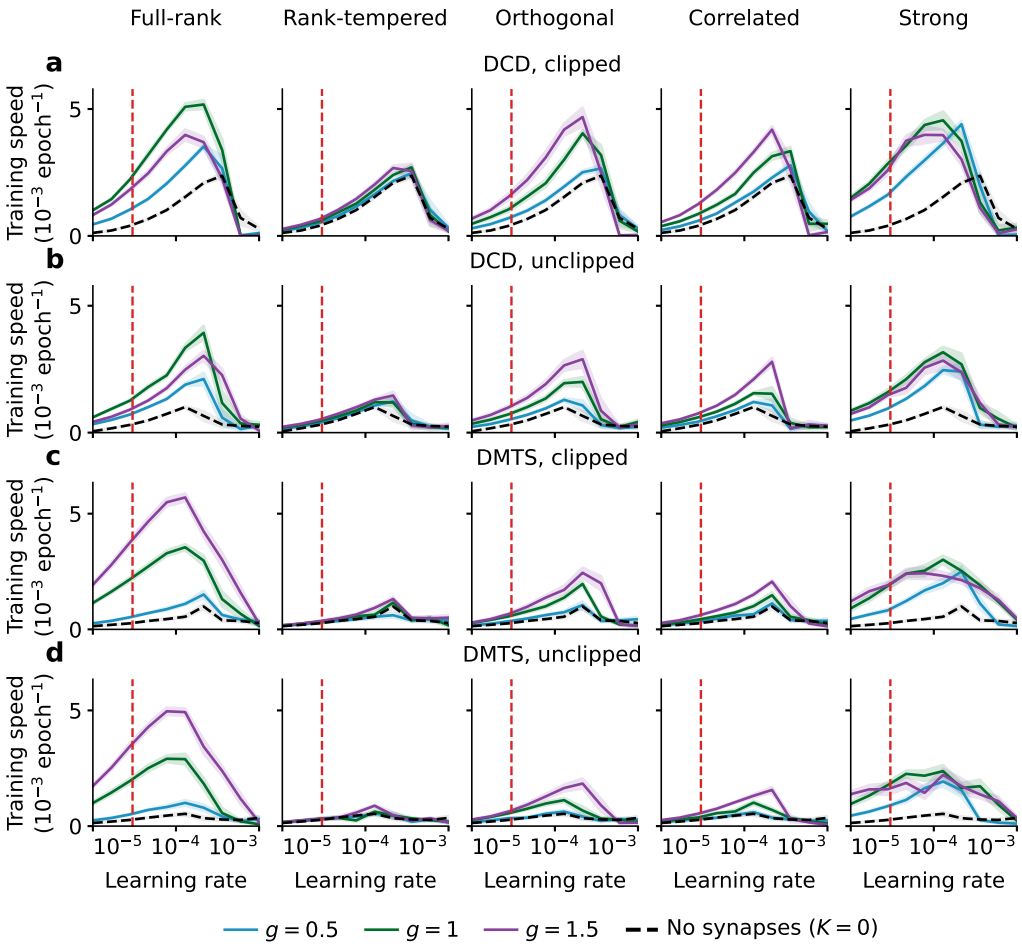

Figure S4: **Learning speed depends on learning rate and gradient clipping.** Training speed versus learning rate for delayed cue discrimination (**a-b**) and delayed match-to-sample (**c-d**), with global gradient clipping at 1 (**a,c**) or without clipping (**b,d**). Networks use $N = 1{,}000$ and initialization rank $K = 10$, except for the full-rank and zero-initialization references. Colors indicate gain g; black dashed curves show zero recurrent initialization. Correlated initialization uses $\rho = 0.7$. Red dashed lines mark the learning rate 3×10^{-5} used in Figures 4, S2, and S3. Speed is the reciprocal of the final epoch of the first 50 consecutive epochs with training MSE below 0.05. Runs without such a window within 10,000 epochs receive zero speed. Curves and shading show mean $\pm$ SEM across 10 seeds. Rank-tempered initialization, which yields weak synapses at $K = 10$, provides limited improvement over zero initialization, whereas strong and structured initializations achieve higher learning speeds at suitable learning rates. Random strong initialization can match or exceed the speeds of both structured weak-synapse initializations across the two tasks.

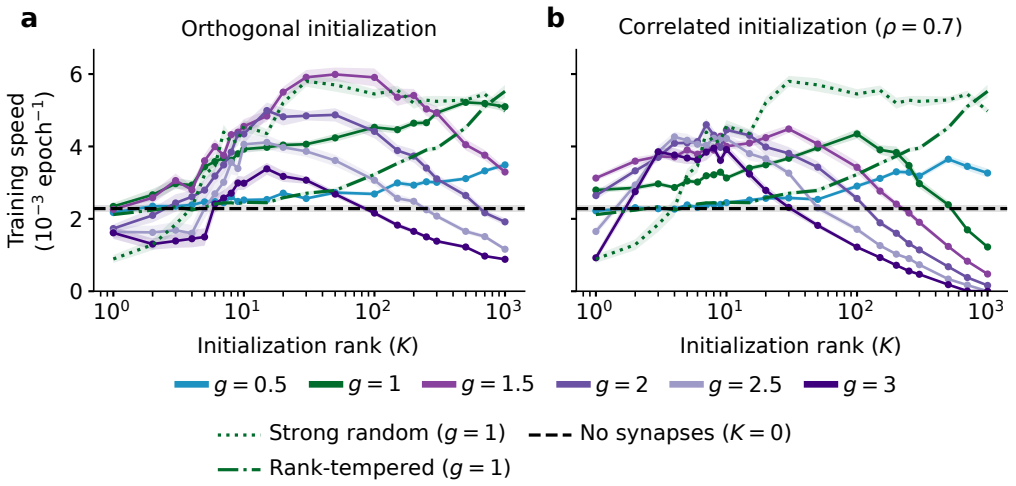

Figure S5: **Learning speed for distinct initialization methods and ranks at optimized learning rates.** Training speed versus initialization rank K for orthogonal (**a**) and correlated ($\rho = 0.7$; **b**) initialization on delayed cue discrimination, with $N = 1{,}000$. Colors indicate gain g. Dotted and dash-dotted curves show strong random and rank-tempered initialization at $g = 1$; black dashed lines mark zero recurrent initialization. Learning rates were selected from the $K = 10$ sweeps in Figure S4**a** and held fixed across ranks and gains, and were approximately 1.29×10^{-4} for strong initialization, 2.15×10^{-4} for orthogonal, correlated, and rank-tempered initialization, and 3.59×10^{-4} for zero initialization. Gradients were clipped to a global norm of 1. Speed is the reciprocal of the final epoch of the first 50 consecutive epochs with training MSE below 0.05. Runs without such a window within 10,000 epochs receive zero speed. Curves and shading show mean $\pm$ SEM across all 20 seeds.

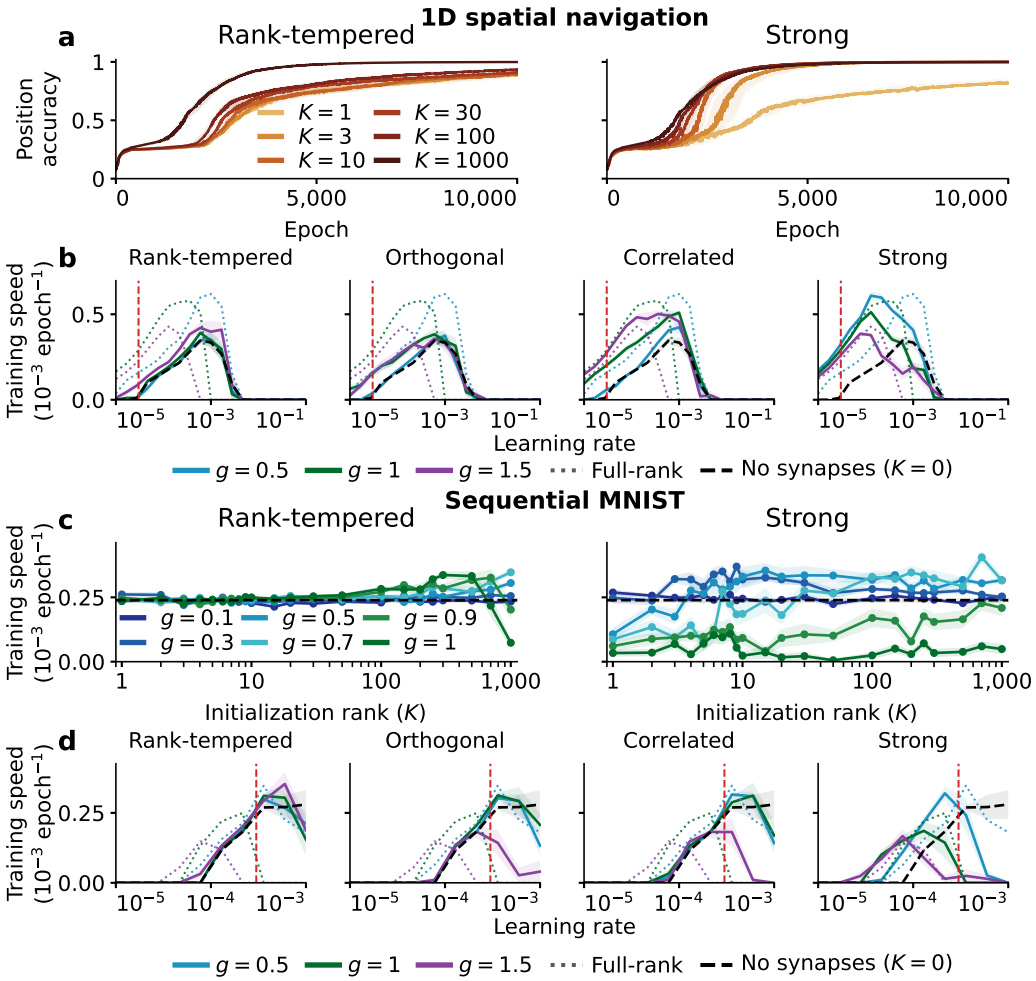

Figure S6: **Rank-deficient initialization in spatial navigation and sequential image classification.** All networks have $N = 1{,}000$ neurons, rank constraints apply only at initialization, and the full recurrent matrix is trainable. **a-b** 1D spatial navigation. Networks receive signed velocity and learn to integrate it into position on a circular track starting from a known location. The target output is the activity of 100 Gaussian place cells with evenly spaced preferred positions. At each time point, the preferred location of the most active output cell defines the predicted position. A prediction is counted as correct if the shortest distance around the track between the predicted and true positions is at most 5% of the track circumference. Position accuracy is the fraction of correct predictions across all time points and training trajectories. **a** Position accuracy versus epoch for rank-tempered and strong initialization at $g = 1$. Colors indicate initialization rank K. Curves and shading show the median and interquartile range across 20 seeds. **b** Training speed versus learning rate for four initialization methods at $K = 10$. **c-d** Sequential MNIST. Images are downsampled to 7×7 pixels and presented sequentially. Networks predict the digit class after the final pixel. Each epoch denotes one minibatch iteration with 200 training samples. Test accuracy is evaluated every ten iterations on 100 randomly sampled test images. **c** Training speed versus initialization rank for rank-tempered and strong initialization. **d** Training speed versus learning rate at $K = 10$. Colors in **b-d** indicate gain g. In **b,d**, dotted curves show gain-matched full-rank references, correlated initialization uses $\rho = 0.7$. Black dashed curves show zero recurrent initialization. Red dashed lines mark the learning rates used in **a,c**, respectively (3×10^{-5} and 3×10^{-4}). Speed is the reciprocal of the final epoch of the first qualifying window, which requires 50 consecutive epochs with position accuracy strictly greater than 90% for navigation, or six consecutive test evaluations with accuracy strictly greater than 90% for MNIST. Runs without a qualifying window within 10,000 epochs receive zero speed. Curves and shading in **b-d** show mean $\pm$ SEM across all 10 seeds in the learning-rate sweeps and all 20 seeds in the rank sweep. Gradients were clipped to a global norm of 1 in **a,b** and 100 in **c,d**.

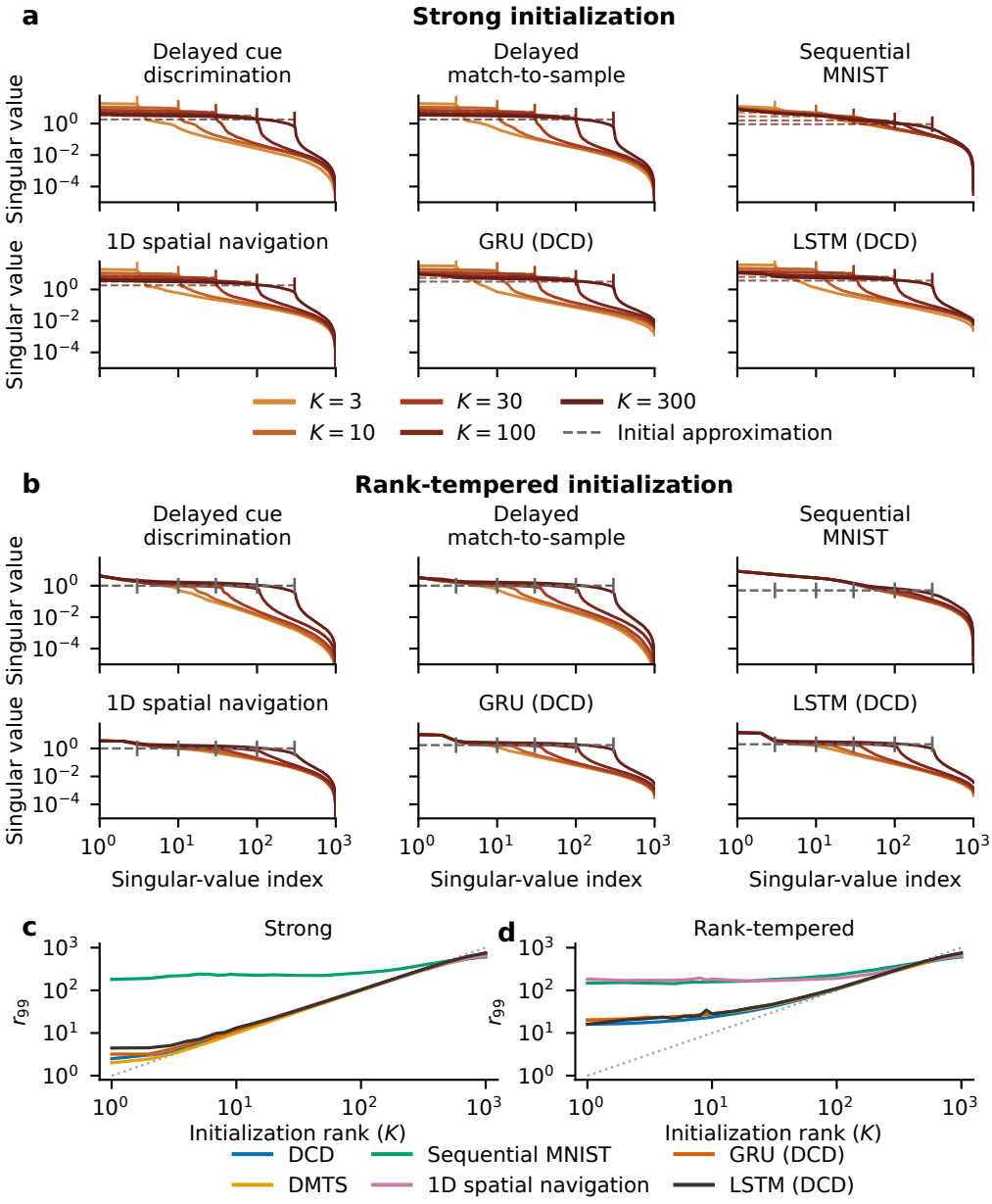

Figure S7: **Low-rank initialization shapes the effective rank of learned recurrent connectivity across tasks and architectures.** Singular value spectra of the learned recurrent weight matrices for strong (**a**) and rank-tempered (**b**) initialization across tasks and architectures, with $N = 1{,}000$ neurons, $g = 0.5$ for Sequential MNIST (cf. Fig. S6**c**) and $g = 1$ otherwise. Colors indicate initialization rank K. Dashed guides with endpoints at K approximate the initial nonzero singular values analytically. Gated networks use the stacked recurrent gate matrices. **c-d** Effective rank r_{99}, the smallest number of singular values capturing 99% of the total squared singular values, for strong and rank-tempered initialization. Colors indicate task or architecture, whereas dotted lines mark the identity line $r_{99} = K$. For all panels, curves and shading show mean $\pm$ SEM across networks satisfying the task-specific success criterion over the final evaluation window, from 20 seeds for vanilla RNNs and 10 for gated networks.

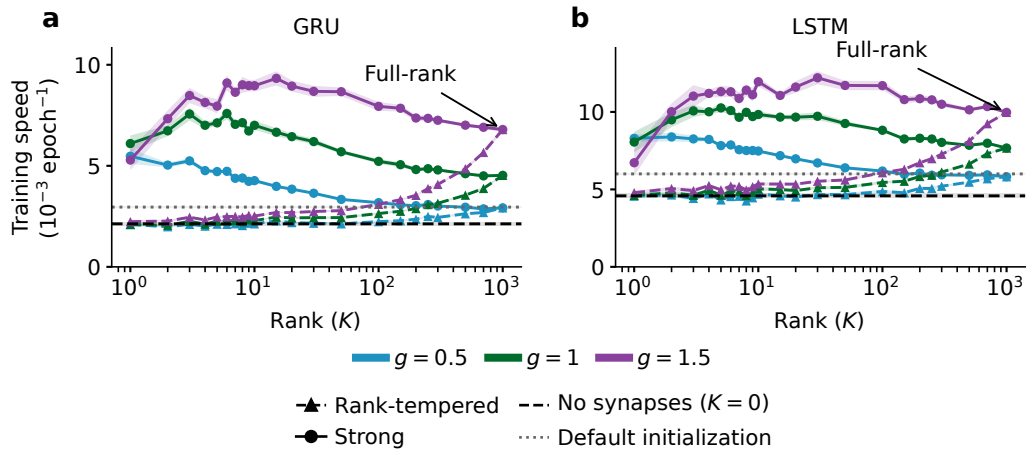

Figure S8: **Rank-deficient initialization with strong synapses supports rapid learning in gated RNNs on delayed cue discrimination.** Training speed versus initialization rank K for GRUs (**a**) and LSTMs (**b**) on delayed cue discrimination, with $N = 1{,}000$. Colors indicate gain g. Strong and rank-tempered initializations have recurrent weight variances g^2/N and Kg^2/N^2, respectively. Black dashed and gray dotted lines mark zero recurrent and default initializations. Speed is the reciprocal of the final epoch of the first 50 consecutive epochs with training MSE below 0.05. Runs without such a window within 20,000 epochs receive zero speed. Curves and shading show mean $\pm$ SEM across 10 seeds. Learning rates are 10^{-5} for GRUs and 3×10^{-5} for LSTMs, neither used gradient clipping.

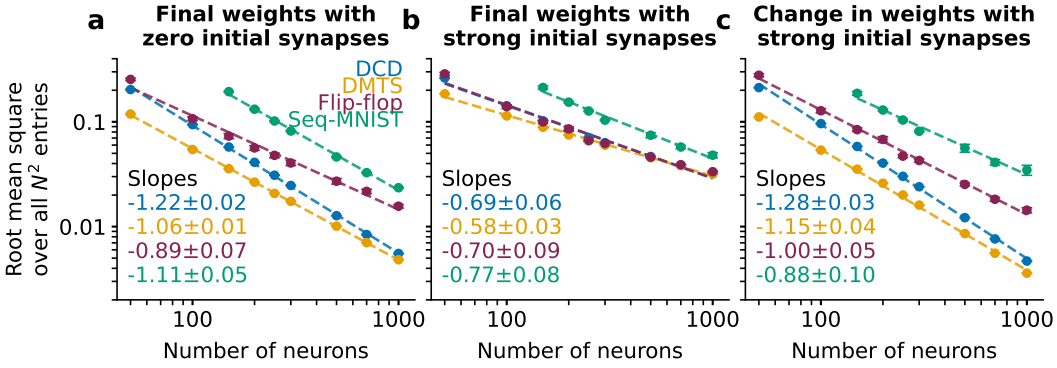

Figure S9: **Root-mean-square measures show similar scaling of learned weights and updates with network size.** Same as Figure 5, but using root-mean-square values instead of mean absolute values.

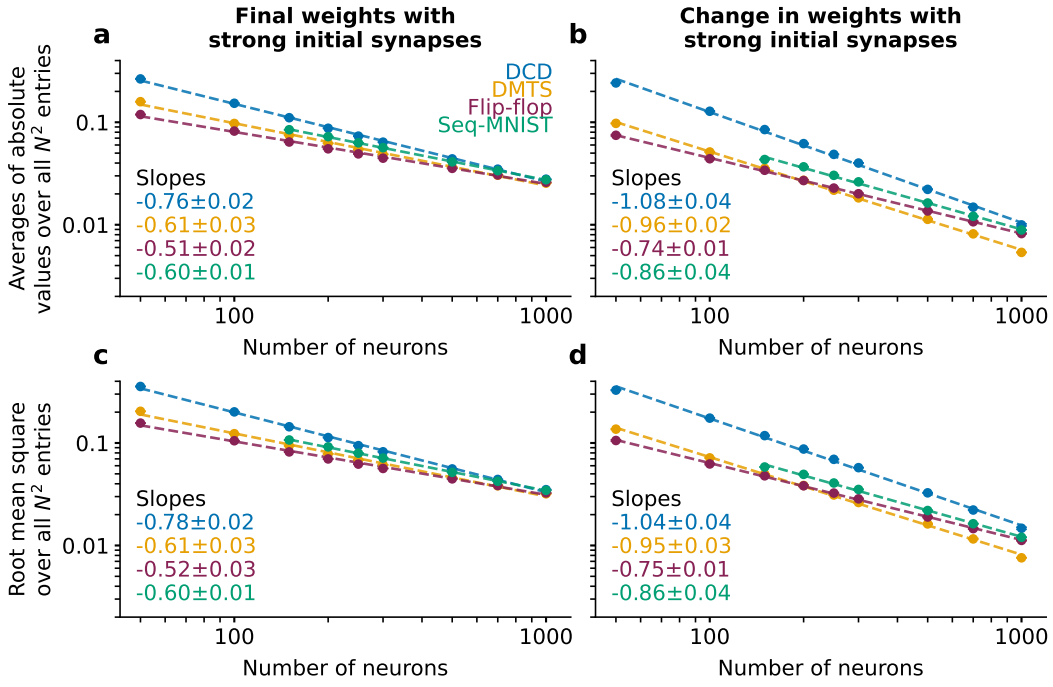

Figure S10: **Weight and update scaling under learning rates proportional to $N^{-1/2}$.** Same analysis as Figure 5**b-c**, using learning rates proportional to $N^{-1/2}$. Left: final recurrent weights. Right: changes from initialization. Top: mean absolute matrix entries. Bottom: root-mean-square entries. Colors, fits, error bars, and success criteria follow Figure 5.

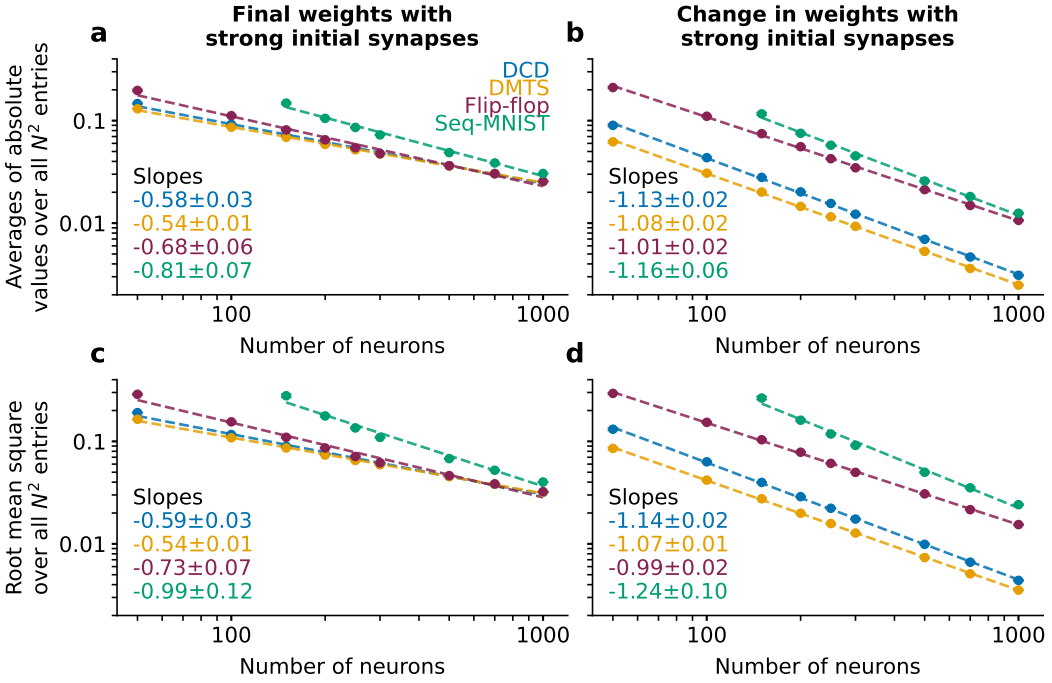

Figure S11: **Weight and update scaling in networks with ReLU activation.** Same analysis as Figure 5**b-c**, using ReLU networks with full-rank strong initialization and learning rates proportional to N^{-1}. Left: final recurrent weights. Right: changes from initialization. Top: mean absolute matrix entries. Bottom: root-mean-square entries. Colors, fits, error bars, and final-window success criteria follow Figure 5.

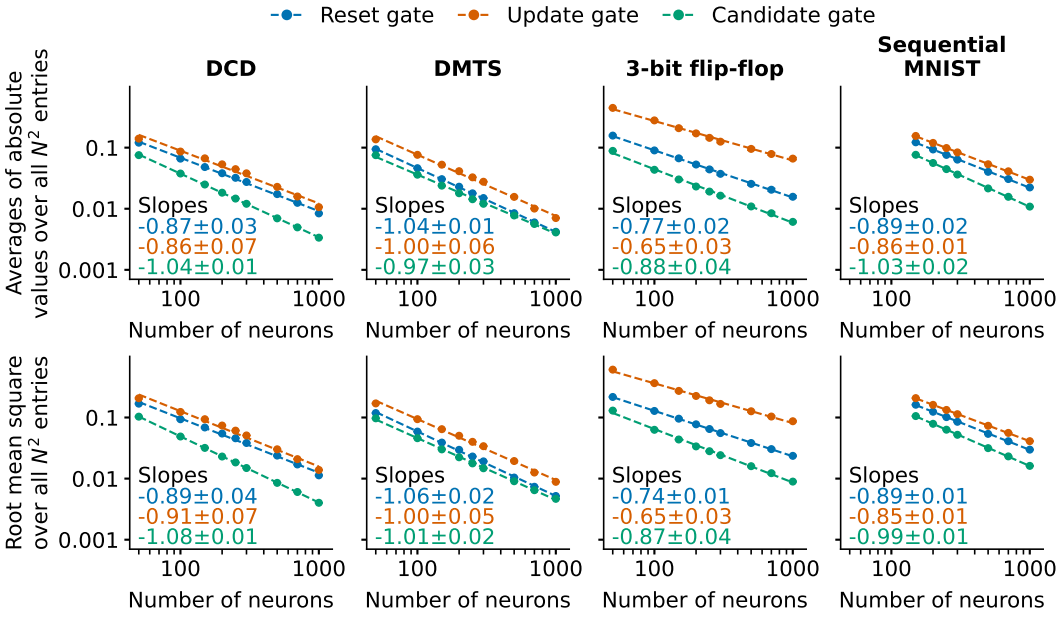

Figure S12: **Gate-specific recurrent updates in GRUs scale with network size across tasks.** Mean absolute (top) and root-mean-square (bottom) recurrent weight changes versus network size. Colors indicate reset, update, and candidate gates. Recurrent weight entries are sampled independently as $W_{ij}^{\mathrm{rec}} \sim \mathcal{U}(-N^{-1/2}, N^{-1/2})$. Learning rates proportional to N^{-1}. Fits, slope annotations, error bars, and final-window success criteria follow Figure 5, with up to 20 seeds per condition.

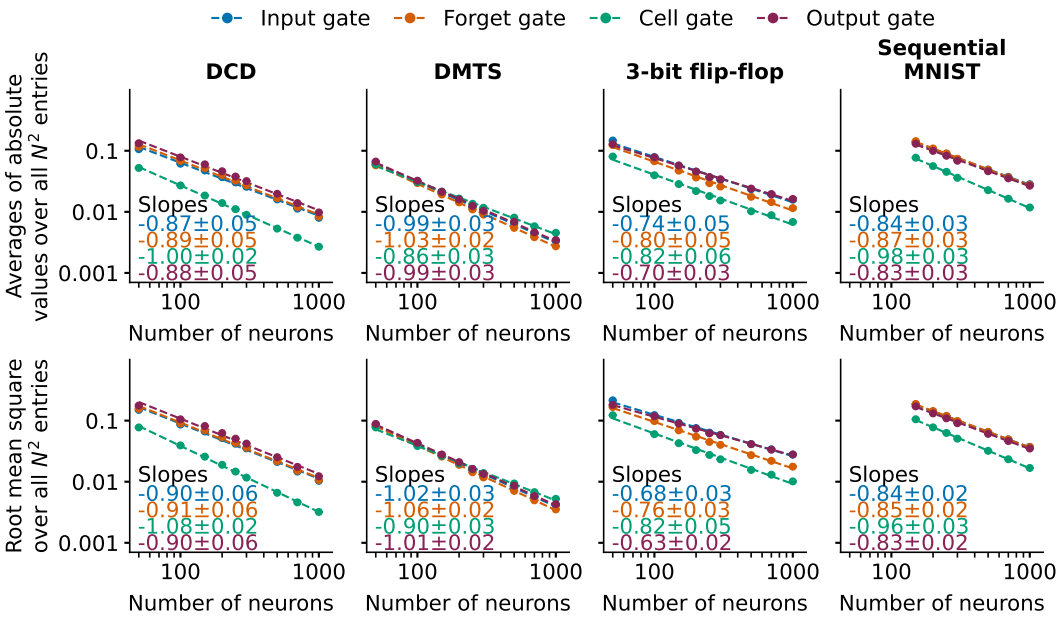

Figure S13: **Gate-specific recurrent updates in LSTMs scale with network size across tasks.** Mean absolute (top) and root-mean-square (bottom) recurrent weight changes versus network size. Colors indicate input, forget, cell, and output gates. Recurrent weight entries are sampled independently as $W_{ij}^{\text{rec}} \sim \mathcal{U}(-N^{-1/2}, N^{-1/2})$. Learning rates proportional to N^{-1}. Fits, slope annotations, error bars, and final-window success criteria follow Figure 5, with up to 20 seeds per condition.

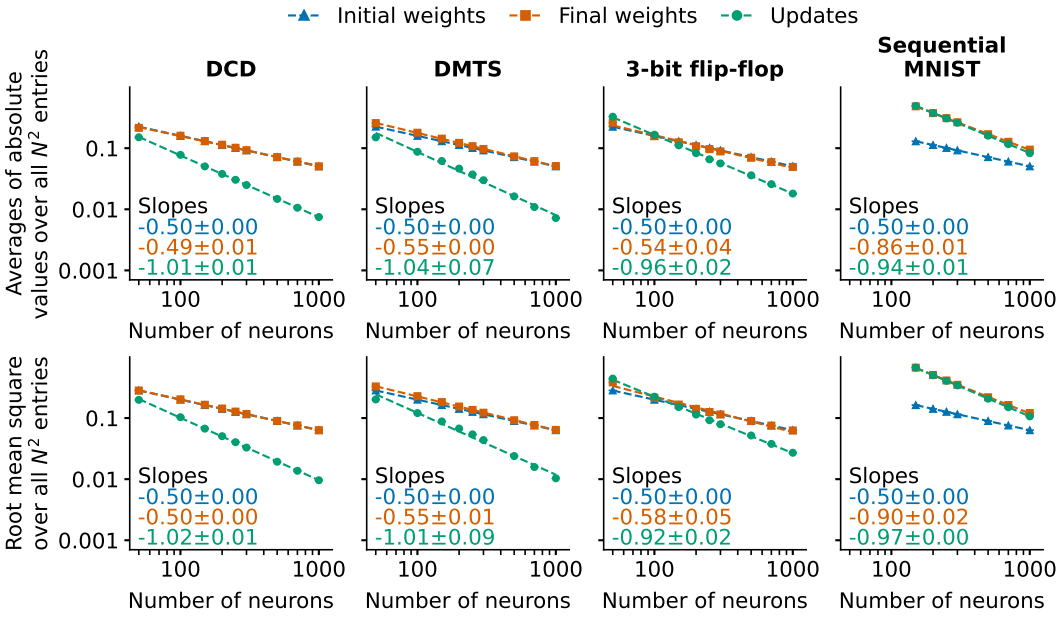

Figure S14: **Learned synaptic updates in spiking RNNs exhibit approximately inverse-size scaling.** Mean absolute (top) and root-mean-square (bottom) values of initial synaptic weights, final synaptic weights, and their changes versus network size N. Networks use full-rank factorized strong initialization and learning rates proportional to N^{-1}. Fits, slope annotations, error bars, and final-window success criteria follow Figure 5, except for the sequential MNIST task. Due to spiking RNNs' failure to achieve the criteria, for this plot, a successful run requires test accuracy strictly above 70% (instead of 90%) in all six final evaluations. Statistics include successfully trained networks from 20 seeds per condition.

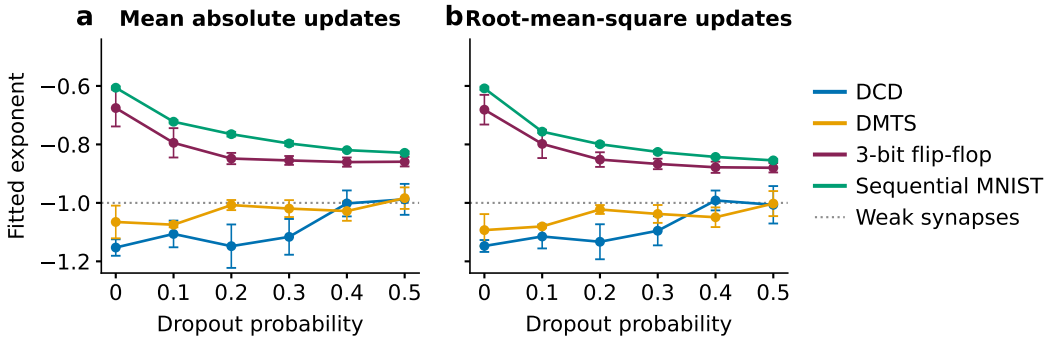

Figure S15: **Training-time dropout shifts learned synaptic updates toward inverse-size scaling.** Fitted scaling exponents $-\alpha$ of mean absolute (**a**) and root-mean-square (**b**) synaptic updates versus dropout probability. Colors indicate tasks. Exponents are obtained by linear regression in log-log coordinates using $N = 500, 600, 700, 800, 900, 1000$; error bars show ± 1.96 regression standard errors. The dotted line marks inverse-size scaling. Networks use strong initialization and learning rates proportional to $N^{-1/2}$. Fits include successfully trained networks under the final-window criteria of Figure 5, from 20 seeds per condition. Supporting fits appear in Figure S16.

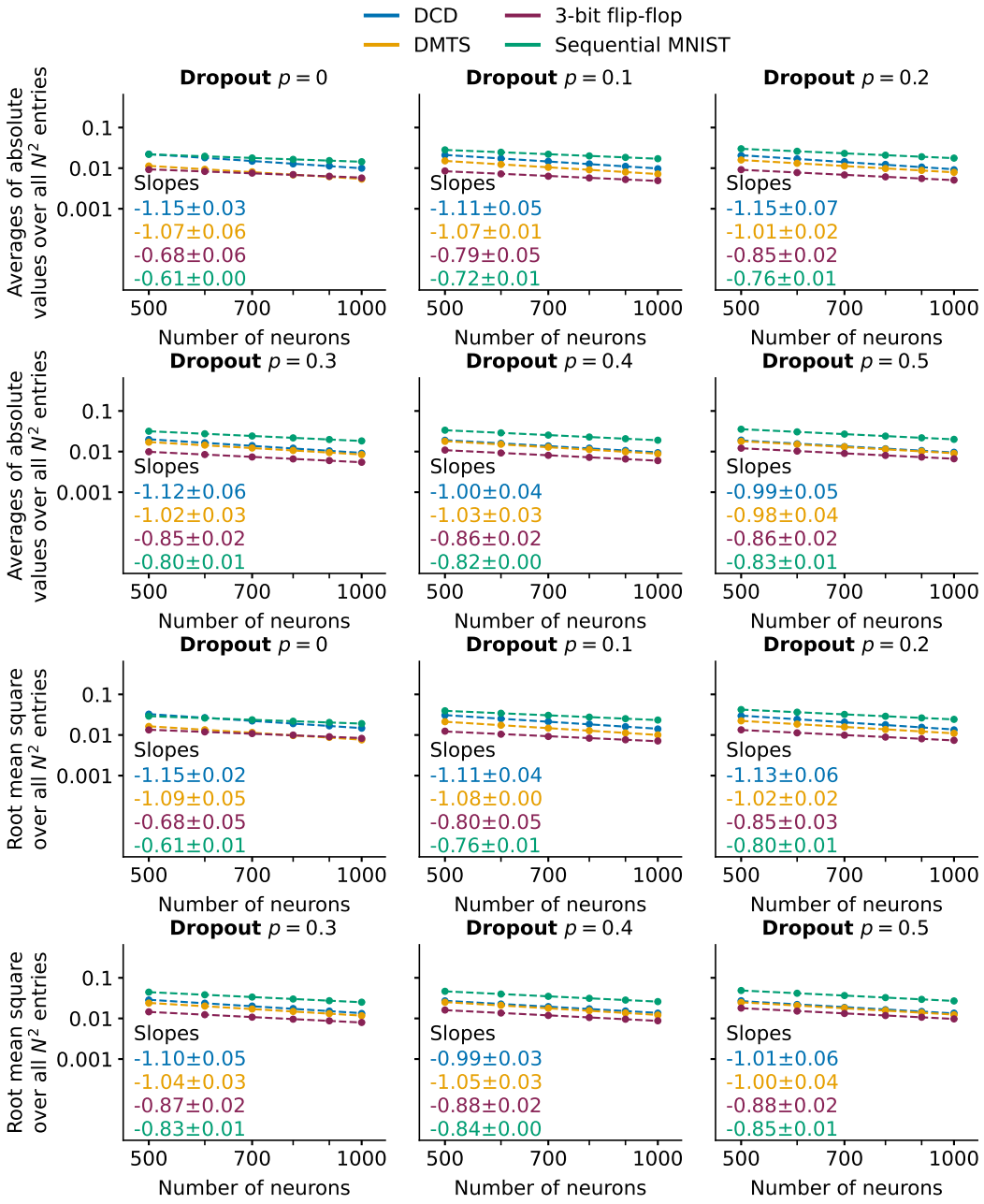

Figure S16: **Power-law fits underlying the dropout-dependent scaling exponents.** Mean absolute (top two rows) and root-mean-square (bottom two rows) synaptic updates versus network size for the dropout probabilities shown. Colors indicate tasks. Points and error bars show mean ± SEM across successfully trained networks, from 20 seeds per condition. Dashed lines show fits using all six network sizes; annotations report slopes ±1.96 regression standard errors. Training protocols and success criteria follow Figure S15.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: They do.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Please see discussion.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Please see propositions and proofs in appendices.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide the codebase here https://github.com/fatihdinc/Rank-deficient-RNNs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Here https://github.com/fatihdinc/Rank-deficient-RNNs.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Please see appropriate sections, appendices, and codebase.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: They are reported.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We used personal desktops to run most of the experiments, about 1 month-long GPU time would be sufficient to produce the results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: It does.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: There is no impact.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: No risk.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [N/A]

    Justification: NA.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Code is public.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: NA.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: NA

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: We used LLMs to assist in preparing the codebase for public release, to provide a secondary check of whether the revisions addressed the requests of the area chair and reviewers, and to check mathematical proofs for potential errors.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.