The interplay between randomness and structure during learning in RNNs

Friedrich Schuessler
Technion
schuessler@campus.technion.ac.il

Francesca Mastrogiuseppe
Gatsby Unit, UCL
f.mastrogiuseppe@ucl.ac.uk

Alexis Dubreuil
ENS Paris
alexis.dubreuil@gmail.com

Srdjan Ostojic
ENS Paris
srdjan.ostojic@ens.fr

Omri Barak
Technion
omri.barak@gmail.com

Abstract

Recurrent neural networks (RNNs) trained on low-dimensional tasks have been widely used to model functional biological networks. However, the solutions found by learning and the effect of initial connectivity are not well understood. Here, we examine RNNs trained using gradient descent on different tasks inspired by the neuroscience literature. We find that the changes in recurrent connectivity can be described by low-rank matrices, despite the unconstrained nature of the learning algorithm. To identify the origin of the low-rank structure, we turn to an analytically tractable setting: training a linear RNN on a simplified task. We show how the low-dimensional task structure leads to low-rank changes to connectivity. This low-rank structure allows us to explain and quantify the phenomenon of accelerated learning in the presence of random initial connectivity. Altogether, our study opens a new perspective to understanding trained RNNs in terms of both the learning process and the resulting network structure.

1 Introduction

Recurrent neural networks (RNNs) have been used both as tools for machine learning, and as models for neuroscience. In the latter context, RNNs are typically initialized with random connectivity and trained on abstractions of tasks used in experimental settings [3, 21, 24, 33, 34, 36, 38, 41]. The obtained networks are then compared to both behavioral and neural experimental results, with the added advantage that the RNNs are more amenable to analysis than their biological counterparts [35]. Despite this advantage, the understanding of how RNNs implement neuroscience tasks is still limited. Open questions concern especially the relationship between the final connectivity and the task, and its formation through training.

Here, we examine the relation between the initial connectivity of the RNN, the task at hand, and the changes to connectivity through training. We use unconstrained gradient descent that can potentially alter the connectivity completely. However, evaluating nonlinear RNNs trained on several neuroscience-inspired tasks, we observe that the connectivity changes are small compared to the initial connectivity. We thus split the connectivity matrix W at the end of training into the initial part W_0 and the changes ΔW, writing

$$W = W_0 + \Delta W \,. \tag{1}$$

For all tasks we consider, we find that the training-induced connectivity structure ΔW is of low rank, despite the unconstrained nature of training used. This finding directly connects gradient-based learning with a number of existing neuroscience frameworks based on low-rank aspects of connectivity

Preprint. Under review.

[4, 9, 13, 15, 19, 22, 25, 34, 37]. Despite the low-rank nature of the *changes* to connectivity ΔW, the initial, full-rank, random connectivity W_0 plays an important role in learning. Consistent with previous work [29, 34], we find that the initial connectivity accelerates learning. Moreover we show that the final, trained network relies on correlations between ΔW and W_0.

In the second part of our work, we analyze the mechanism behind these observations in a simplified and analytically tractable setting: nonlinear dynamics of learning in a linear RNN trained on a simple input-output mapping task. We show how the low-dimensional task structure leads to low-rank connectivity changes; importantly, the amplitude and geometry of these low-rank changes depend on the random initial connectivity. Our work reveals how this dependence accelerates learning and quantifies the degree of acceleration as a function of initial connectivity strength.

Finally, we show that our results extend to real-world settings of an LSTM network trained on a natural language processing task, suggesting practical applications of our results.

2 Training RNNs on low-dimensional tasks

Tasks We trained RNNs on three tasks inspired by the neuroscience literature. All tasks are characterized by a small number of input and output channels. The first task is a working memory task, in which the network receives pulses from two different input channels and needs to remember the sign of the last pulse in each channel independently [35]. The second task is a context-dependent decision task: The network receives two noisy signals, as well as one of two context inputs which indicates the relevant signal. After the input presentation, it needs to output whether the average of the relevant signal was positive or negative [21]. The third task is a delayed-discrimination task [26] in which the network receives two positive pulses separated by a delay. After yet another delay, it needs to output which of the two pulses had the larger amplitude. Based on their origin, we refer to the three tasks as "flip-flop" [35], "Mante" [21], and "Romo" [26] task, respectively. For each task, we plotted a single trial for a successfully trained network in Fig. 1**(a-c)**. Detailed parameters can be found in the supplementary.

RNN model Each RNN model consists of N neurons whose state vector evolves according to

$$\dot{\mathbf{x}}(t) = -\mathbf{x}(t) + W\phi(\mathbf{x}(t)) + \sqrt{N}\sum_{i=1}^{N_{\text{in}}} \mathbf{m}_i u_i(t)\,. \tag{2}$$

The recurrent input is given by the firing rate vector $\phi(\mathbf{x})$ multiplied by the weight matrix W. We use the element-wise nonlinearity $\phi = \tanh$. The network receives time-dependent inputs $u_i(t)$ through input vectors $\mathbf{m}_i$. The output is the projection of the firing rate onto readout vectors $\mathbf{w}_i$, namely

$$z_i(t) = \frac{\mathbf{w}_i^T \phi(\mathbf{x}(t))}{\sqrt{N}} \quad \text{for } i \text{ in } \{1, \ldots, N_{\text{out}}\}\,. \tag{3}$$

We formulate target values $\hat{z}_i(t)$ during specific segments of the trial [see dark lines for output panels in Fig. 1**(a-c)**]. The task determines the numbers N_{in} and N_{out} of input and output vectors. For example, the Mante task requires four input vectors (for both signals and contexts) and a single output vector. We are interested in the behavior of large networks, $N >> 1$, while the dimension of the tasks is small, $N_{\text{in}}, N_{\text{out}} \sim \mathcal{O}(1)$. For the simulation, we chose N to be large enough so that learning dynamics become invariant under changes in N (see supplementary Fig. S1).

Training and initialization For training the RNNs, we formulated a quadratic cost in $z_i(t)$ and applied the gradient descent method "Adam" [16] to the internal connectivity W as well as to the input and output vectors $\mathbf{m}_i$, $\mathbf{w}_i$. Restricting the updates to W or training with SGD impaired the convergence times but yielded similar results (not shown). The initial input and output vectors were drawn independently from $\mathcal{N}(0,\, 1/N)$. We initialized the internal weights as a random matrix W_0 with independent elements drawn from $\mathcal{N}(0,\, g^2/N)$. The parameter g thus scales the strength of the initial connectivity.

Learning dynamics in the absence of initial connectivity To understand what kind of connectivity arises during learning, we first looked at the simplest case without initial connectivity, $g = 0$. The loss curves indicate convergence for all three tasks [see darker lines in Fig. 1**(d-f)**]. We analyzed the

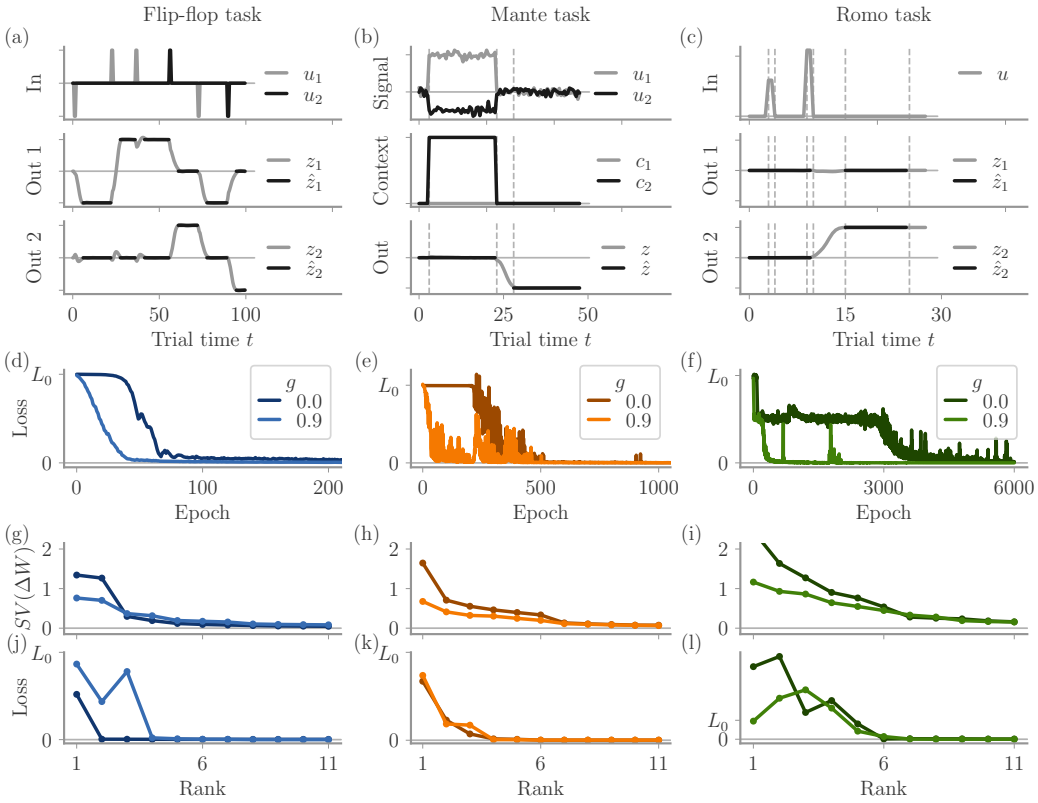

Fig. 1: Learning dynamics in three different neuroscience tasks. **(a-c)** Task summary: inputs u_i, outputs z_i, and targets $\hat{z}_i$ for each task. Dashed lines indicate task phases. **(d-f)** Loss throughout training process for different initial connectivity strengths g. L_0 is the loss at the beginning of training for $g = 0$ (L_0 is different for different tasks). Note the different epoch numbers plotted. **(g-i)** First 11 singular values of final connectivity changes ΔW. **(j-l)** Loss for truncated networks, where ΔW is replaced with the rank-R approximation $\Delta W^{(R)}$. Parameters: $N = 256$, learning rate $\eta = 0.05/N$.

connectivity at the end of training by computing its singular values (SVs). For the flip-flop task, we found that the first two SVs were much larger than the remaining ones [Fig. 1**(g)**]. To see whether the network utilizes this approximate rank-two structure, we replaced the changes ΔW with the singular value decomposition truncated at rank R,

$$\Delta W^{(R)} = \sum_{r=1}^{R} s_r \mathbf{u}_r \mathbf{v}_r^T \,. \tag{4}$$

Note that we keep the initial connectivity W_0. The loss after truncation indeed drops to zero at rank 2 [Fig. 1**(j)**]. A similar situation is observed for the Mante and Romo tasks, see Fig. 1**(h, k)** and **(i, l)**, respectively. Although for these tasks the SVs drop more slowly, the first six SVs are discernibly larger than the remaining tail; the truncation loss drops to zero at rank 4 and 6, respectively. In sum, we observe that for $g = 0$, training via gradient descent yields an effective low-rank solution for all three tasks.

Effects of initial connectivity on learning dynamics and connectivity The loss-curves in Fig. 1**(d-f)** indicate a strong influence of the initial connectivity strength g on the training dynamics (lighter colors for $g = 0.9$). We observe that learning becomes faster and smoother with initial connectivity. In Fig. 2**(a)**, we quantify the acceleration of learning with the number of epochs needed to reach 5% of the initial loss. We observe that convergence time smoothly decreases as a function of connectivity strength g; for very large g, networks finally transition to chaotic activity [32], and convergence time increases again.

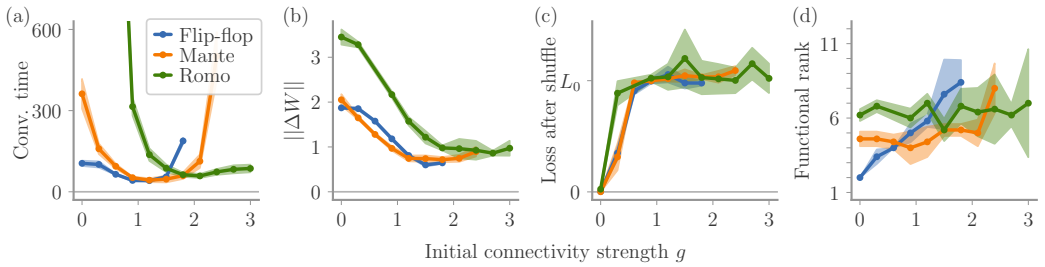

Fig. 2: Dependence of learning dynamics on initial connectivity strength g in the three tasks. Lines and shades indicate mean and standard deviation of five independent simulations for each g, respectively. **(a)** Number of epochs at which the loss falls below 5% of L_0. **(b)** Frobenius norm of ΔW at the end of training. **(c)** Loss for shuffled initial connectivity, so that the full network connectivity is given by $W_0^{\text{shuffle}} + \Delta W$. **(d)** Rank R at which the loss of the network with rank-truncated connectivity $\Delta W^{(R)}$ drops below 5% of the initial loss L_0.

After observing the drastic decrease in learning time, we wondered how initial connectivity affects the resulting connectivity changes. The first observation is that, for increasing g, the final connectivity $W = W_0 + \Delta W$ is dominated by W_0, since $||W_0|| = \sqrt{N}g$. In fact, the norm of ΔW not only remains unchanged for increasing N (see supplementary), but further decreases with increasing g, see Fig. 2**(b)**. If a smaller ΔW solves the task for larger initial connectivity, it is reasonable to assume that W_0 amplifies the effect of ΔW. To test this idea, we shuffled the elements of W_0, destroying any correlation between W_0 and ΔW, while maintaining its statistics. The loss after replacing the connectivity with $W_0^{\text{shuffle}} + \Delta W$ is shown in Figure 2**(c)**. For all tasks, shuffling strongly degraded performance except for cases with very weak initial connectivity.

Low-rank changes in connectivity Despite the effects of the initial connectivity on convergence time and the norm of ΔW, the low-rank nature of ΔW remains similar to the case with $g = 0$. In Fig. 1**(g-h)**, the SVs of ΔW are plotted in lighter colors. We see that the pattern and overall amplitude is very similar to the darker lines for $g = 0$: only a small number of SVs dominates over a tail. To assess the functional rank, we replaced ΔW in our RNN with the rank-R truncation, Eq. (4), while keeping the initial connectivity W_0 identical. The resulting loss, Fig. 1**(j-l)**, indicates that the effective connectivity change is indeed low-rank: for all three tasks, it drops to a value close to zero before rank 10. We quantified this observation by computing the "functional rank", the rank at which the loss decreases below 5% of the initial value [see Fig. 2**(d)**]. This functional rank is between 2 and 10 for all three tasks (averaged over independent simulations). It increases with g for the flip-flop task, while it remains less affected for the other two tasks.

3 Analytical results for linear system

The observation of effective low-rank changes in connectivity and accelerated learning for random initial connectivity were general across the three different tasks considered. To understand the underlying mechanisms, we turn to a much simpler task and a linear RNN model. This setting allows us to analytically describe the learning dynamics, understand the origin of the low-rank connectivity changes, and quantify how correlations between W_0 and ΔW accelerate learning. Our approach is similar to that of Saxe et al. [28], who analyzed gradient descent dynamics in linear feed-forward networks. Both for the feed-forward and the recurrent model, the learning dynamics are nonlinear despite the linearity of the networks. Nevertheless, we will see that the recurrent nature of our models results in very different dynamics compared to the linear feed-forward model. Below we will present our main results for the simplified model; the details of all our analytical derivations can be found in the supplementary.

Simplified setting Our simple task is an input-output transformation: Given a constant input $u(t) = 1$, the output $z(t)$ has to reach a target value $\hat{z}$ at time T. The corresponding loss is $L = (\hat{z} - z(T))^2/2$. An example with two different target values $\hat{z} = 0.5,\ 2.0$ is plotted in Fig. 3**(a)**. The linear RNN model is obtained by replacing the nonlinearity in Eq. (2) with the identity, $\phi(\mathbf{x}) = \mathbf{x}$,

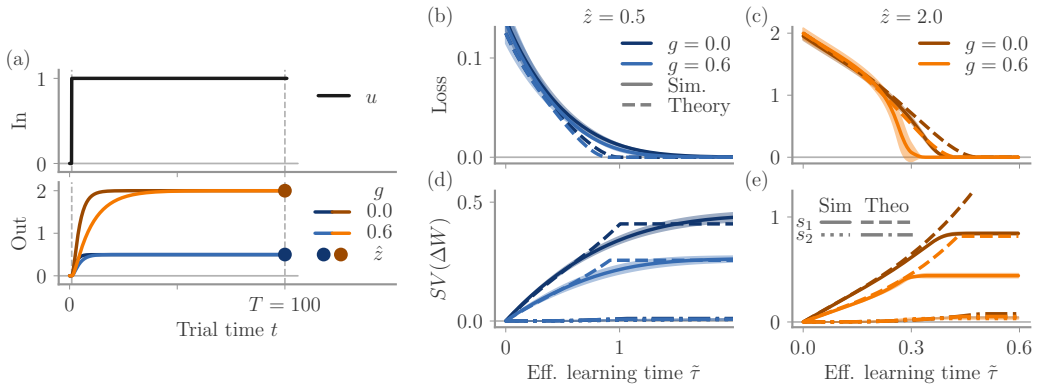

Fig. 3: Learning a simple input-output transformation in a linear network. **(a)** Task summary. Output for trained networks with two different initial connectivity strengths $g = 0.0,\ 0.6$ and target amplitudes $\hat{z} = 0.5,\ 2.0$. Input starts at $t = 1$, loss is evaluated at $T = 100$. **(b,c)** Loss over training for target values $\hat{z} = 0.5$ and $\hat{z} = 2.0$. Full lines indicate simulation results, dashed lines our theoretical prediction. **(d,e)** First two SVs of ΔW at the end of training (full, dotted lines) and theoretical predictions (dashed, dashed-dotted). In panels **(b-e)**, the simulation results are averaged over five independent instances. Shades, if visible, indicate the standard deviation. Note that the x-axes in **(b-e)** show the rescaled, effective learning time $\tilde{\tau} = \beta^2\tau$, with $\beta = 1/(1-g^2)$. Simulation parameters: $N = 1024$, training for 200 epochs with learning rate η adapted (see supplementary).

and keeping only a single input and output. All weights are initialized as before. We keep the initial connectivity strength $g < 1$ so that the linear network remains stable. To further simplify, we constrain weight changes to the recurrent weights W only, and apply plain gradient descent. To compare between different simulations, we define the learning time $\tau = \eta \cdot \text{epochs}$.

Evaluating the trained networks reveals similar phenomena as observed for the nonlinear, more complex tasks. Figure 3**(b-e)** shows the loss and SVs of ΔW over learning time for two values of g. We observe that learning induces low-rank connectivity changes – in fact, a single SV dominates. Because of the small magnitude of the second SV, truncating ΔW at rank 1 does not lead to increased loss (not shown), so that the functional rank as defined in the previous section is 1. Comparing between $g = 0$ and $g = 0.6$, we further see that learning is accelerated by the initial connectivity, and that the magnitude of the first SV decreases with increasing g. These observations will be quantified with our analytical results.

Gradient descent dynamics For our analytical treatment, we only consider the limit of long trials, with the output $z = \lim_{T\to\infty} z(T)$ at the end of a trial. In this limit, the network converges to its fixed point $\mathbf{x}^* = \sqrt{N}\,(I - W)^{-1}\,\mathbf{m}$ with identity matrix I, and the readout is

$$z = \frac{\mathbf{w}^T\mathbf{x}^*}{\sqrt{N}} = \mathbf{w}^T(I - W)^{-1}\,\mathbf{m}\,. \tag{5}$$

The input and output vectors, $\mathbf{m}$ and $\mathbf{w}$, remain fixed during training, and only W is changed. We can explicitly compute the changes induced by the gradient of the loss:

$$\frac{\mathrm{d}W(\tau)}{\mathrm{d}\tau} = -\frac{\mathrm{d}L}{\mathrm{d}W} = [\hat{z} - z(\tau)]\,\left[I - W^T(\tau)\right]^{-1}\mathbf{w}\mathbf{m}^T\left[I - W^T(\tau)\right]^{-1}\,, \tag{6}$$

with initial connectivity $W(0) = W_0$. We made a continuous-time approximation of the weight updates ("gradient flow"), valid to small learning rates η. Note that the readout z at the fixed point depends on the learning time τ through $W(\tau)$.

Note that, unlike the feed-forward case [27], the inverse of W appears in Eq. (6), opening the possibility of divergence during learning. It also precludes a closed-form solution to the dynamics. However, we can obtain analytical insight by expanding the learning dynamics in learning time around the initial connectivity [5]. We write

$$W(\tau) = \sum_{k=0}^{\infty} W_k\,\frac{\tau^k}{k!}\,. \tag{7}$$

The changes in connectivity are obtained by subtracting W_0, which yields $\Delta W(\tau) = W_1\tau + W_2\tau^2/2 + \dots$. We analytically computed the coefficients W_k by evaluating $\mathrm{d}^k W/\mathrm{d}\tau^k$ at $\tau = 0$. A comparison of the expansion up to third order with the numerical results from gradient descent learning indicates close agreement during most of the learning [see Fig. 3**(b-e)** full vs. dashed lines].

Learning dynamics in absence of initial connectivity It is instructive to first consider the case of no initial connectivity, $g = 0$. The readout at the beginning of training is then $z_0 = \mathbf{w}^T\mathbf{m}$. Due to the independence of $\mathbf{m}$ and $\mathbf{w}$, the expected value of z_0 vanishes. Moreover, the standard deviation scales as $1/\sqrt{N}$ with the network size. In this work, we are interested in the learning dynamics for large networks; all our analytical results are valid in the limit $N \to \infty$. We therefore write $z_0 = 0$. Similar reasoning goes for all scalar quantities of interest: they are of order $\mathcal{O}(1)$, with deviations $\mathcal{O}(1/\sqrt{N})$. With this self-averaging quality, we omit stating the limit as well as the expectation symbol and use the equality sign instead.

Inserting W_0 and z_0 – both zero – into the gradient descent, Eq. (6), yields the first order coefficient

$$W_1 = \hat{z}\mathbf{w}\mathbf{m}^T. \tag{8}$$

Hence, the weight changes at linear order in τ are described by a rank-one matrix, and the readout is $z(\tau) = \tau\hat{z} + \mathcal{O}(\tau^2)$. The gradient descent for $g = 0$ would therefore converge at $\tau_1^* = 1$, if it only depended on the first-order term. The numerical results already show deviations in the form of faster or slower convergence, depending on the target $\hat{z}$ [see dark lines in Fig. 3**(b,c)** and note that $\tilde{\tau} = \tau$ for $g = 0$]. This indicates the importance of higher order terms.

We observe that the gradient in Eq. (6) contains the transpose W^T. At higher orders, this term introduces other outer-product combinations of $\mathbf{m}$ and $\mathbf{w}$. In fact, for $g = 0$, these are the only vectors present in the gradient, so that the connectivity can always be written as

$$\Delta W(\tau) = [\mathbf{w} \quad \mathbf{m}] \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} \mathbf{w}^T \\ \mathbf{m}^T \end{bmatrix} . \tag{9}$$

This form implies that ΔW will be at most a rank-two matrix. An analysis of the SVs [Eq. (14) below for general g] reveals that the second SV remains very small, as visible in Fig. 3**(d,e)**.

The entries of the 2×2 matrix $A(\tau)$ up to order $\mathcal{O}(\tau^3)$ are (see supplementary)

$$A_{11} = \frac{\hat{z}^2}{2}\left(\tau^2 - \tau^3\right) , \qquad A_{12} = \hat{z}\left(\tau - \frac{\tau^2}{2} + \frac{\tau^3}{6}(1 + 2\hat{z}^2)\right) , \qquad A_{21} = \frac{\hat{z}^3\tau^3}{3} , \tag{10}$$

and $A_{22} = A_{11}$. The first surprising observation is that the target value $\hat{z}$ enters nonlinearly into the expressions above. This is the origin of the qualitative difference between learning curves for different values of the target output in Fig. 3**(b,c)**.

We further observe that the connectivity changes develop a nonzero eigenvalue only at $\mathcal{O}(\tau^2)$. This is because the off-diagonal terms, which grow linearly with τ contribute a zero eigenvalue because $\mathbf{m}^T\mathbf{w} = 0$. At second order the diagonal entries of A – and, with it, the eigenvalues – change. Changes in connectivity eigenvalues imply changes in time scales of network dynamics, which may be necessary for some tasks (for example, those involving memory), but can also lead to problems of exploding gradients (see supplementary).

Effects of initial connectivity In the presence of initial connectivity, we can still apply the expansion introduced above. Due to the independence of W_0, $\mathbf{m}$, and $\mathbf{w}$, the initial readout z_0 remains zero. The gradient descent, Eq. (6), then directly yields the first-order connectivity coefficient

$$W_1 = \hat{z}\, B^T\mathbf{w}\,\mathbf{m}^T B^T, \qquad \text{with} \qquad B = (I - W_0)^{-1} . \tag{11}$$

Thus, W_1 is still a rank-one matrix despite the full-rank initial connectivity. However, the connectivity changes now include the initial connectivity W_0 via the matrix B. As a consequence, the norm of the first-order coefficient, $||W_1|| = \hat{z}\beta$ (see supplementary), increases with g by the factor

$$\beta = \mathbf{w}^T B B^T\mathbf{w} = \mathbf{m}^T B^T B\mathbf{m} = \frac{1}{1 - g^2} . \tag{12}$$

The readout is also affected by the initial connectivity. We compute (see supplementary)

$$z(\tau) = \tau\hat{z}\beta^2 + \mathcal{O}(\tau^2) . \tag{13}$$

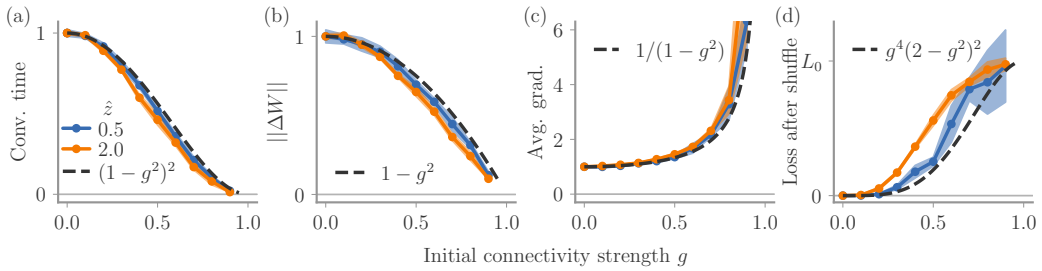

Fig. 4: Dependence of learning dynamics on initial connectivity strength g in the simplified task. **(a)** Learning time τ^* until loss reached 5% of its initial value. **(b)** Norm of final weight changes ΔW. **(c)** Norm of gradient $\mathrm{d}W/\mathrm{d}\tau$, averaged over the interval $\tau \in [0, \tau^*]$. The quantities in panels **(a-c)** are normalized by their value at $g = 0$. **(d)** Loss after shuffling the initial connectivity W_0, normalized by initial loss. In all panels, thick full lines indicate the average over five independent simulations, shades the standard deviation, and dashed lines the first-order theoretical prediction.

Learning converges when $z(\tau)$ reaches the target value $\hat{z}$. The first-order prediction of the convergence time is therefore $\tau_1^* = 1/\beta^2$, and the initial connectivity accelerates learning by the factor $1/\beta^2 = (1-g^2)^2$. We can decompose this acceleration into two factors: The growth rate is increased by β, and the norm of the final connectivity changes decreased by $1/\beta$. For the first contribution, we note that the first-order coefficient W_1 is, by definition, the constant part of the gradient, and hence the rate at which connectivity changes. For the second contribution, we compute the norm of $\Delta W(\tau)$ at the predicted convergence time τ_1^* (see supplementary).

In Fig. 4**(a-c)**, we compare our first-order predictions with numerical simulations. In panels **(a,b)**, we plot the convergence time τ^* and the norm of ΔW at the end of training. As for the more complex, nonlinear tasks [see Fig. 2**(a,b)**], we defined the numerical τ^* as the point in time where the loss drops to 5% of the initial value. For the gradient, panel **(c)**, we averaged the norm $||\mathrm{d}W/\mathrm{d}\tau||$ over the interval $[0, \tau^*]$. To compare the collapsed curves with the predicted scalings, we normalized the curves for the different target values $\hat{z}$ by their value at $g = 0$ for all three quantities. We observe good agreement between the numerical results and the theory, even though we only used the first-order predictions, and τ^* often shows notable differences between theory and simulation [for example in Fig. 3**(b,c)**].

Finally, we assess the role of correlations between ΔW and W_0 by shuffling W_0. After shuffling, the readout loses the amplification by β^2 and is hence $z^{\text{shuff}} = \tau_1^* \hat{z}$. The corresponding loss is $L_1^{\text{shuff}} = L_0\, g^4(2-g^2)^2$, with initial loss $L_0 = \hat{z}^2/2$. A comparison of this first-order prediction with numerical results shows qualitative agreement with notable quantitative differences especially for the larger target amplitude, see Fig. 4**(d)**. A comparison with the nonlinear case, Fig. 2**(c)** shows that our simple model captures the phenomenon qualitatively.

Higher-order terms Does the initial connectivity lead to higher-rank changes in connectivity? For $g > 0$, the explicit rank-two expression for the weight changes, Eq. (9), does not hold anymore: The input and output vectors accumulate multiples of B and B^T (such as $B^T\mathbf{w}$ and $BB^T\mathbf{w}$) which increase the number of possible outer products – and hence potentially the rank. However, computing the first two SVs, s_1 and s_2, up to order $\mathcal{O}(\tau^3)$ (see supplementary) shows that ΔW remains approximately rank one:

$$s_1 = \frac{\hat{z}}{\beta}\left[\tilde{\tau} - \frac{\tilde{\tau}^2}{2} + \left(1 + \frac{7}{2}\hat{z}^2\beta\right)\frac{\tilde{\tau}^3}{6}\right], \qquad s_2 = \hat{z}^3\frac{\tilde{\tau}^3}{12}. \tag{14}$$

where $\tilde{\tau} = \beta^2\tau$ is the effective learning time. We observe that s_1 grows linearly, but s_2 only at third order of τ. Different parts of connectivity therefore grow on top of each other, giving rise to a temporal hierarchy in the learning dynamics. Numerical simulations show good agreement with this prediction (see supplementary).

We further state the resulting readout up to $\mathcal{O}(\tau^3)$:

$$z(\tau) = \hat{z}\left[\tilde{\tau} - \frac{\tilde{\tau}^2}{2} + (1 + 8\hat{z}^2\beta)\frac{\tilde{\tau}^3}{6}\right]. \tag{15}$$

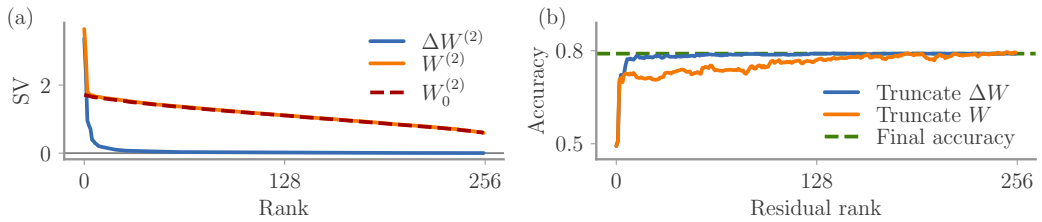

Fig. 5: Low-rank changes for a two-layer LSTM model trained on a sentiment analysis task. **(a)** Singular values (SVs) of the recurrent weights in the second layer (256 neurons). The initial, random W_0 is full rank, and the final W visibly differs from it only for the first SVs. The changes, ΔW, are approximately low-rank. **(b)** Validation accuracy after truncating the lower singular values of connectivity. We either truncated W directly, or applied truncation only to ΔW while keeping W_0. We truncated the recurrent weights of both layers and the input weights of layer 2.

The appearance of β in the third-order contributions in Eqs. (14) and (15) shows that the learning with different values of g does not entirely collapse onto one curve after rescaling the time by β^2. Instead, there is an additional acceleration, which increases with increasing target amplitude $\hat{z}$. This effect can be appreciated in Fig. 3**(b,c)**, where for larger $\hat{z}$ the loss curve becomes concave. Note that our approximation up to $\mathcal{O}(\tau^3)$ predicts this trend, despite quantitative disagreement. As we saw in Fig. 4, the scaling of the convergence time τ^* with g is not strongly affected by the higher order terms.

4 Beyond neuroscience tasks

We asked whether our observation that connectivity changes are low-rank despite full-rank initial connectivity would extend to more complex network architectures and tasks, specifically those not restricted to a small input or output dimension. We therefore trained a two-layer LSTM network on a natural language processing task, sentiment analysis of movie reviews [31] (details in supplementary).

The SVs at the end of training showed the pattern that we predicted: learning only leads to small changes in the connectivity so that the final connectivity W is dominated by the initial connectivity and has full rank. The changes ΔW only have a small number of large SVs. For the recurrent weights of layer 2, the SVs are plotted in Fig. 5**(a)**; other weights behave similarly (see supplementary).

Like before, we evaluated the accuracy of networks after truncation at a given rank, see Fig. 5**(b)**. We truncated the recurrent weights of both layers as well as input weights to layer 2. If we keep the random parts and truncate the changes as in Eq. (4) a rank-10 approximation already yields the final training accuracy. In contrast, if we truncate the entire weight matrices, as previously suggested [39], it takes more that half of the network rank (256 neurons per layer) to get close to the final accuracy.

5 Discussion

Summary of results Our key finding is that the connectivity changes ΔW induced by unconstrained training on low-dimensional tasks are of low rank. With our simplified analytical model, we demonstrated why: The connectivity changes are spanned by a small number of existing directions, determined by the input and output vectors. Without initial connectivity, the maximum rank that linear networks can obtain through learning is in fact bounded by this number. The initial connectivity W_0 enlarges the pool of available directions. The fact that learning arrives at a low-rank solution even in presence of initial connectivity is then a result of the temporal structure of learning: Initially, only a small number of available directions grow, inducing a low-rank structure. For our simplified task, the first of these structures already reduces the loss, and learning converges before other structures emerge; the final connectivity changes are hence rank-one. For other tasks, the available input and output directions alone may not be sufficient, so that initial connectivity becomes necessary for successful learning (see supplementary). Note that our theoretical analysis is limited to linear networks; however, nonlinearity may also contribute to generate novel learning directions.

Our numerical simulations further showed that initial connectivity significantly accelerated learning. Our analytical results revealed the underlying mechanism: The input and output vectors spanning the gradient are multiplied by powers of W_0, which strongly correlates ΔW to W_0. This correlation amplifies the effect of ΔW, and removing the correlation by shuffling W_0 indeed degrades performance. This is in line with a recent study demonstrating such amplification through correlation between a random matrix and a low-rank perturbation in a model without learning [30].

Finally, we showed that the general observation of low-rank weight changes indeed holds even in a much more complex setting: a sentiment analysis task and a two-layer LSTM network. This implies a large potential for network compression [39]: one may truncate the changes in connectivity at a very low rank and recover the specific random initial connectivity using the seed of its random number generator.

Task dimension and rank Low-rank connectivity structures have previously been studied and applied. On the one hand, a number of RNN frameworks explicitly rely on low-rank feedback for training [4, 9, 15, 19, 34]. On the other hand, low-rank networks are amenable to analysis, because the network activity is low-dimensional and evolves in directions determined by the vectors spanning the connectivity [13, 22, 25, 30, 37]. Our surprising observation that unconstrained gradient descent also leads to low-rank connectivity opens new possibilities for studying general gradient-based learning with the tools developed by previous works.

We observed that the functional rank of the training-induced connectivity changes is strongly task dependent. A better understanding of the relation between task and connectivity calls for a concept of a task dimension, ideally based on the underlying abstract computations and independent of the specific implementation [10, 18, 20, 41]. Such a concept would allow to compare the solutions obtained by different algorithms and define a necessary minimal rank for a given task [8].

Learning as a dynamical process and relation to feed-forward networks Our approach stresses a dynamical perspective on learning, in which the solutions are not determined by the task alone, but also by the initial connectivity and the temporal evolution of weight changes. In particular, our expansion in learning time shows that some components in the connectivity only grow after others are present, which induces a temporal hierarchy. This affects the solutions the network arrives at. The temporal structure may also induce pitfalls for learning, for example divergent gradients when the networks undergo a phase transition [23] (see supplementary). A better understanding of the learning dynamics could be used to circumvent such problems, for example by introducing adapted learning curricula [6].

Learning in feed-forward networks has previously been analyzed from a similar perspective. It was found that the statistical structure of the training data induces a temporal hierarchy with long plateaus between step-like transitions in the learning curve [1, 11, 17, 27, 28, 42]. The hierarchy in our work originates in the dynamics of the RNN rather than the structure of the training data. For example, the plateaus seen in Fig. 1**(d-f)** can be related to phase transitions in the network dynamics, such as the emergence of new fixed points. Combining such internal learning dynamics with structured training data would be an interesting future direction.

Finally, recent work on feed-forward networks identified two different learning regimes: a kernel regime vs. a rich, feature-learning regime [2, 7, 14, 40]. In the prior, the change in weights vanishes as the network width increases, and the network function can be linearized around the weights at initialization. In our work, too, the weight changes ΔW become infinitely small in the limit of wide networks. However, even such vanishing ΔW may significantly change the dynamics of the neural network by inducing large outlier eigenvalues [30]. For example, the readout for our linear network, Eq. (5), diverges for a eigenvalue of W approaching 1. In such a case, the network function cannot be approximated by linearization around the initial weights. Understanding the relation between learning regimes in feed-forward and recurrent networks constitutes an interesting field for future studies.

Broader Impact

This work is a theoretical study on the dynamics of learning in RNNs. We show which kind of connectivity changes are induced by gradient descent. We expect that our insights will help to understand learning in RNNs, which benefits the research community as a whole and may ultimately lead to the development of improved learning algorithms or schemes. As a possible application, we show that one can use our results to efficiently compress a multi-layer RNN trained on a natural language processing task. In this work, there are no new algorithms, tasks, or data sets introduced. Therefore, the questions regarding any disadvantages, failures of the system, or biases do not apply.

Acknowledgments and Disclosure of Funding

This work was supported in part by the Israeli Science Foundation (grant number 346/16, OB). The project was further supported by the ANR project MORSE (ANR-16-CE37-0016), the program "Ecoles Universitaires de Recherche" launched by the French Government and implemented by the ANR, with the reference ANR-17-EURE-0017. F.S. acknowledges the Max Planck Society for a Minerva Fellowship. There are no competing interests.

References

[1] Madhu S Advani and Andrew M Saxe. High-dimensional dynamics of generalization error in neural networks. *arXiv:1710.03667*, 2017.

[2] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. *arXiv:1901.08584*, 2019.

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

[4] Omri Barak and Sandro Romani. Mapping low-dimensional dynamics to high-dimensional neural activity: A derivation of the ring model from the neural engineering framework. *arXiv:2002.03420*, 2020.

[5] Carl M Bender and Steven A Orszag. *Advanced mathematical methods for scientists and engineers I: Asymptotic methods and perturbation theory*. Springer Science & Business Media, 2013.

[6] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48, 2009.

[7] Lenaic Chizat and Francis Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. In *Advances in neural information processing systems*, pages 3036–3046, 2018.

[8] Alexis Dubreuil, Adrian Valente, Manuel Beiran, Francesca Mastrogiuseppe, and Srdjan Ostojic. Complementary roles of dimensionality and population structure in neural computations. *bioRxiv doi: 10.1101/185942*, 2020.

[9] Chris Eliasmith and Charles H Anderson. *Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems*. MIT press, 2004.

[10] Peiran Gao, Eric Trautmann, Byron M Yu, Gopal Santhanam, Stephen Ryu, Krishna Shenoy, and Surya Ganguli. A theory of multineuronal dimensionality, dynamics and measurement. *bioRxiv doi: 10.1101/214262v2*, 2017.

[11] Sebastian Goldt, Marc Mézard, Florent Krzakala, and Lenka Zdeborová. Modelling the influence of data structure on learning in neural networks. *arXiv:1909.11500*, 2019.

[12] David A Harville. *Matrix algebra from a statistician's perspective*. Taylor & Francis Group, 1998.

[13] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

[14] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

[15] Herbert Jaeger and Harald Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667):78–80, 2004.

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

[17] Andrew K Lampinen and Surya Ganguli. An analytic theory of generalization dynamics and transfer learning in deep linear networks. *arXiv:1809.10374*, 2018.

[18] Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. *arXiv:1804.08838*, 2018.

[19] Laureline Logiaco, Larry F Abbott, and Sean Escola. A model of flexible motor sequencing through thalamic control of cortical dynamics. *bioRxiv doi: 10.1101/2019.12.17.880153*, 2019.

[20] Niru Maheswaranathan, Alex Williams, Matthew Golub, Surya Ganguli, and David Sussillo. Universality and individuality in neural dynamics across large populations of recurrent networks. In *Advances in Neural Information Processing Systems*, pages 15603–15615, 2019.

[21] Valerio Mante, David Sussillo, Krishna V Shenoy, and William T Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, 503(7474):78, 2013.

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

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

[24] Blake A Richards, Timothy P Lillicrap, Philippe Beaudoin, Yoshua Bengio, Rafal Bogacz, Amelia Christensen, Claudia Clopath, Rui Ponte Costa, Archy de Berker, Surya Ganguli, et al. A deep learning framework for neuroscience. *Nature Neuroscience*, 22(11):1761–1770, 2019.

[25] Alexander Rivkind and Omri Barak. Local dynamics in trained recurrent neural networks. *Physical Review Letters*, 118(25):258101, 2017.

[26] Ranulfo Romo, Carlos D Brody, Adrián Hernández, and Luis Lemus. Neuronal correlates of parametric working memory in the prefrontal cortex. *Nature*, 399(6735):470, 1999.

[27] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv:1312.6120*, 2013.

[28] Andrew M Saxe, James L McClelland, and Surya Ganguli. A mathematical theory of semantic development in deep neural networks. *Proceedings of the National Academy of Sciences*, 116(23):11537–11546, 2019.

[29] Samuel S Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. *arXiv:1611.01232*, 2016.

[30] Friedrich Schuessler, Alexis Dubreuil, Francesca Mastrogiuseppe, Srdjan Ostojic, and Omri Barak. Dynamics of random recurrent networks with correlated low-rank structure. *Physical Review Research*, 2(1): 013111, 2020.

[31] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

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

[33] David Sussillo. Neural circuits as computational dynamical systems. *Current Opinion in Neurobiology*, 25: 156–163, 2014.

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

[35] 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.

[36] David Sussillo, Mark M Churchland, Matthew T Kaufman, and Krishna V Shenoy. A neural network that finds a naturalistic solution for the production of muscle activity. *Nature Neuroscience*, 18(7):1025, 2015.

[37] B Tirozzi and M Tsodyks. Chaos in highly diluted neural networks. *EPL (Europhysics Letters)*, 14(8):727, 1991.

[38] Jing Wang, Devika Narain, Eghbal A Hosseini, and Mehrdad Jazayeri. Flexible timing by temporal scaling of cortical responses. *Nature Neuroscience*, 21(1):102, 2018.

[39] Genta Indra Winata, Andrea Madotto, Jamin Shin, Elham J Barezi, and Pascale Fung. On the effectiveness of low-rank matrix factorization for lstm model compression. *arXiv:1908.09982*, 2019.

[40] Blake Woodworth, Suriya Gunasekar, Jason D Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and rich regimes in overparametrized models. *arXiv:2002.09277*, 2020.

[41] Guangyu Robert Yang, Madhura R Joglekar, H Francis Song, William T Newsome, and Xiao-Jing Wang. Task representations in neural networks trained to perform many cognitive tasks. *Nature Neuroscience*, 22 (2):297, 2019.

[42] Yuki Yoshida and Masato Okada. Data-dependence of plateau phenomenon in learning with neural network—statistical mechanical analysis. In *Advances in Neural Information Processing Systems*, pages 1722–1730, 2019.

Supplementary information

S1 Simulation parameters

All simulations were based on pytorch [46]. For the nonlinear neuroscience tasks, we applied the gradient descent method "Adam" [16] to the recurrent weights W as well as to the input and output vectors $\mathbf{m}_i$, $\mathbf{w}_i$. We checked that our results did not depend qualitatively on the choice of the "Adam" algorithm over plain gradient descent; however, training converged more easily for this choice of algorithm. We also checked that restricting training to W only (as for the simple model) did not alter our results qualitatively (although, with this restriction, training on the Romo task for small values of g did not converge). Code for reproducing our results can be found on https://github.com/frschu/neurips_2020_interplay_randomness_structure/.

The network size for the results in Figures 1 and 2 was $N = 256$, and the learning rate $\eta = 0.05/N$. We trained the networks for a maximum number of 1000, 2000, and 6000 epochs for the flip-flop, Mante, and Romo task, respectively. Each epoch consisted of a batch of 32 independent task trials. For evaluation of the loss after rank-truncation or shuffling W_0, we used a single batch of 512 independent task trials. Note that for "Adam", the learning rate is scaled with N to obtain approximate invariance of the loss curve for different network sizes N. Further note that Fig 1 does not always show the loss over all learning epochs (so that the differences in the initial phase are more clearly visible).

For the simpler, linear model, we applied plain gradient descent and only adapted W. We trained all models for 200 epochs, and the learning rate was adapted in order to obtain smooth convergence within these 200 epochs. We set $\eta = \eta_0(1-g^2)^2$, with $\eta_0 = 0.015, 0.003$ for $\hat{z} = 0.5, 2.0$, respectively. We checked that our numerical results do not depend on this choice, as long as a sufficiently small learning rate and large enough number of epochs is chosen.

The network dynamics are described by the continuous dynamics

$$\dot{\mathbf{x}}(t) = -\mathbf{x}(t) + W\phi(\mathbf{x}(t)) + \sqrt{N}\sum_{i=1}^{N_{\text{in}}} \mathbf{m}_i u_i(t)\,, \tag{16}$$

with initial condition $\mathrm{x}(0) = \mathbf{0}$. For the simulation, we discretized these using the Euler-forward scheme:

$$\mathbf{x}_{k+1} = (1-\Delta t)\mathbf{x}_k + \Delta t\left[W\phi(\mathbf{x}_k) + \sqrt{N}\sum_{i=1}^{N_{\text{in}}} \mathbf{m}_i u_{i,k}\right]\,, \tag{17}$$

with a discrete time step $\Delta t = 0.5$ and $\mathbf{x}(t = k\Delta t) = \mathbf{x}_k$. We checked that our results did not change qualitatively for choosing a smaller Δt or fully discrete dynamics ($\Delta t = 1$).

For the gradient-based updates, we defined the quadratic loss

$$l(t) = \frac{1}{N_{\text{out}}}\sum_{i=1}^{N_{\text{out}}} \frac{1}{2}\left[z_i(t) - \hat{z}_i(t)\right]^2\,, \tag{18}$$

with readout $z_i(t)$, target $\hat{z}_i(t)$, and number of outputs N_{out}. Depending on the task, the loss was defined only during specific times of the task (during decision or fixation periods, see task descriptions). Accordingly, for each task we defined a boolean mask M_k, indicating the points k on the discrete time grid were the loss was active. The full loss was the average over this mask:

$$L = \frac{1}{N_M}\sum_{k=1}^{k_{\max}} M_k\, l(k\Delta t)\,, \tag{19}$$

with $N_M = \sum_{k=0}^{k_{\max}} M_k$, $k_{\max} = T/\Delta t$ and trial time T.

S2 Task details

All task share a broad overall structure: a trial of length T contains an initial "fixation" period without input of length t_{fix}, followed by the first input. During each input phase of duration t_{stim}, all or some

Table S1: Task parameters

Parameter	Symbol	Flip-flop	Mante	Romo	Simple task
# inputs	N_{in}	2	4	1	1
# outputs	N_{out}	2	1	2	1
Trial duration	T	50	48	30	101
Fixation duration	t_{fix}	1	3	3	1
Stimulus duration	t_{stim}	1	20	1	-
Decision delay	t_{delay}	5	5	5	-
Stimulus delay	t_{sd}	$\mathcal{U}(5, 25)$	-	$\mathcal{U}(2, 8)$	-
Decision duration	t_{dec}	-	20	10	1
Input amplitude	u_{amp}	1	1	$\mathcal{U}(0.5, 1.5)$	1
Target amplitude	$\hat{z}_{\text{amp}}$	0.5	0.5	0.5	$\{0.5, 2.0\}$

of the inputs u_i have a nonzero value with amplitude u_{amp}. Finally, there are distinct decision periods during which the target $\hat{z}$ is nonzero, with amplitude $\hat{z}_{\text{amp}}$. The decision periods are preceded by a decision delay, in which the loss is inactive, and which allows the output to converge to the target value. For the flip-flop task and the simple task, the loss is inactive outside of the decision periods; for the Mante and Romo tasks, all output channels are supposed to stay at zero until the beginning of the decision delay (the corresponding target is $\hat{z}_i = 0$ for all channels i). Below, we describe further details for each task. The parameters and their numerical values used in the simulations reported in the main text are summarized in Table S1.

Flip-flop task During each trial, the network receives a number of short pulses of duration t_{stim}. During such a pulse, one input channel is set to $u_i(t) = s\, u_{\text{amp}}$, the others remain zero. The channel and sign $s \in \{\pm 1\}$ are chosen at random. After each pulse and a following delay period t_{delay}, a decision period starts (the loss is activated). During the decision period, the target value is set to $\hat{z}_i(t) = s\, \hat{z}_{\text{amp}}$. The other channel is supposed to remain silent, $\hat{z}_j(t) = 0$ for $j \neq i$. The decision period ends with the next pulse. The delays between stimuli t_{sd} are drawn randomly. Note that the plotted trial time in Fig. 1 in the main text is $T = 100$, while training was done for $T = 50$.

Mante task Each trial for the Mante task contains only a single, longer input period of duration t_{stim}. Half of the input channels correspond to the signal $u_i(t)$, the other half to a context variable $u_{N_s+i}(t)$, with number of signals $N_s = N_{\text{in}}/2$. The signals each consist of a constant mean and random noise part: $u_i(t) = u_{\text{amp}}[s_i + a_{\text{noise}}\eta_i(t)]$ with random sign $s_i \in \{\pm 1\}$ and Gaussian white noise $\eta_i(t)$. For our simulations, we chose the relative noise amplitude $a_{\text{noise}} = 0.05$. For the discretization, the white noise at time step k is $\eta_{i,k} = n_{i,k}/\sqrt{\eta}$ with standard normal variable $n_{i,k} \sim \mathcal{N}(0, 1)$. During each trial, only a single context is active, $u_{N_s+i} = u_{\text{amp}}\delta_{i,j}$, where j is chosen randomly from the number of inputs N_s. Outside of the input period, all mean values of u_i are set to zero (the noise terms remain active). The input period is followed by a decision phase of length t_{dec}, with a delay t_{delay} in between. During the decision period, the output is supposed to communicate the sign s_j of the relevant input j. The target is constant: $\hat{z}(t) = \hat{z}_{\text{amp}} s_j$, and $\hat{z}_i(t) = 0$ for all $i \neq j$.

Romo task For the Romo task, the RNN model has only one input channel, and each trial contains two input pulses of length t_{stim} each. During the input pulses, the input is $u(t) = u_{\text{amp},1}$ and $u(t) = u_{\text{amp},2}$, with amplitudes drawn from a uniform distribution. Both input amplitudes are redrawn if their difference $|u_{\text{amp},1} - u_{\text{amp},2}|$ is below a minimal difference $u_{\text{min diff}} = 0.2$. The two pulses are separated by a random delay t_{sd}. The end of the second pulse is followed by a delay t_{delay} and a decision period of length t_{dec}. During the decision period, the output should indicate which input pulse was larger: $\hat{z}_j(t) = \hat{z}_{\text{amp}}$ for $j = \arg\max_i(u_{\text{amp,i}})$. The other output should remain at zero.

Simple task The simple task only has a single input and output channel. The input is constant starting from the end of the fixation period: $u(t) = u_{\text{amp}}$ for $t > t_{\text{fix}}$. The decision period is a short interval at the end of the trial, $[T - t_{\text{dec}}, T]$. The target value during the decision period is

$\hat{z}(t) = \hat{z}_{\text{amp}}$. There is no decision delay, and the input remains constant during the decision period. Hence, this task does not contain a memory element like the other three tasks.

S3 Supplementary figures

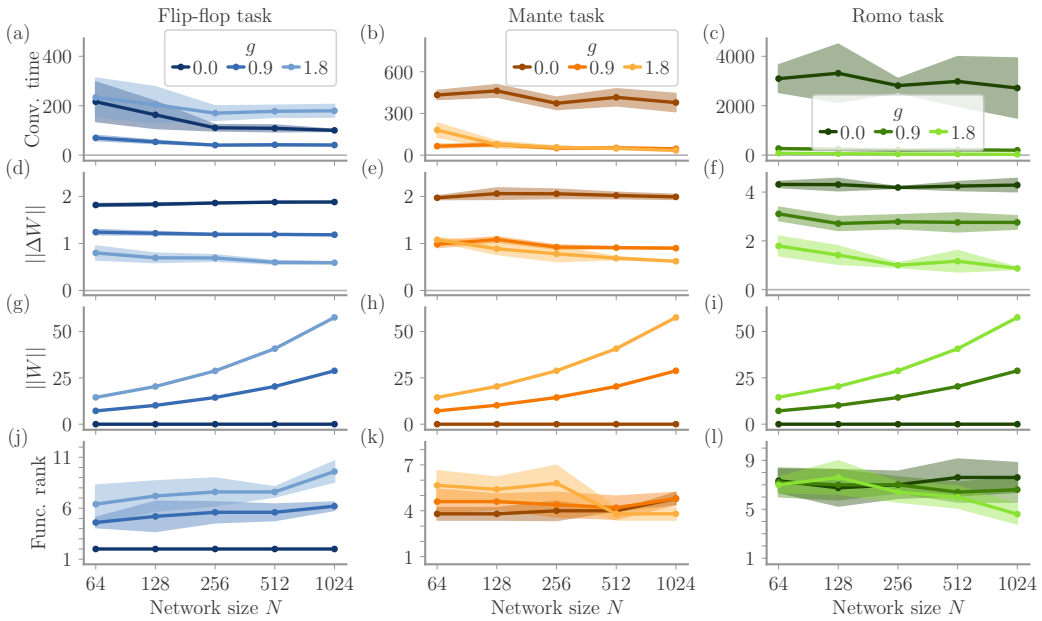

Fig. S1: Scaling of learning dynamics with network size N for all three nonlinear tasks and three different values of initial connectivity g (indicated by line colors). Lines indicate average over 5 independent simulations, shades the standard deviation. Note the log-scale for networks size (x-axes). **(a-c)** Number of epochs until loss reached 5% of its initial value. **(d-f)** Frobenius norm of final connectivity changes ΔW. **(g-i)** Frobenius norm of total connectivity $W = W_0 + \Delta W$. **(j-l)** Functional rank as defined in the main text (the rank at which truncation loss falls below 5% of the initial loss).

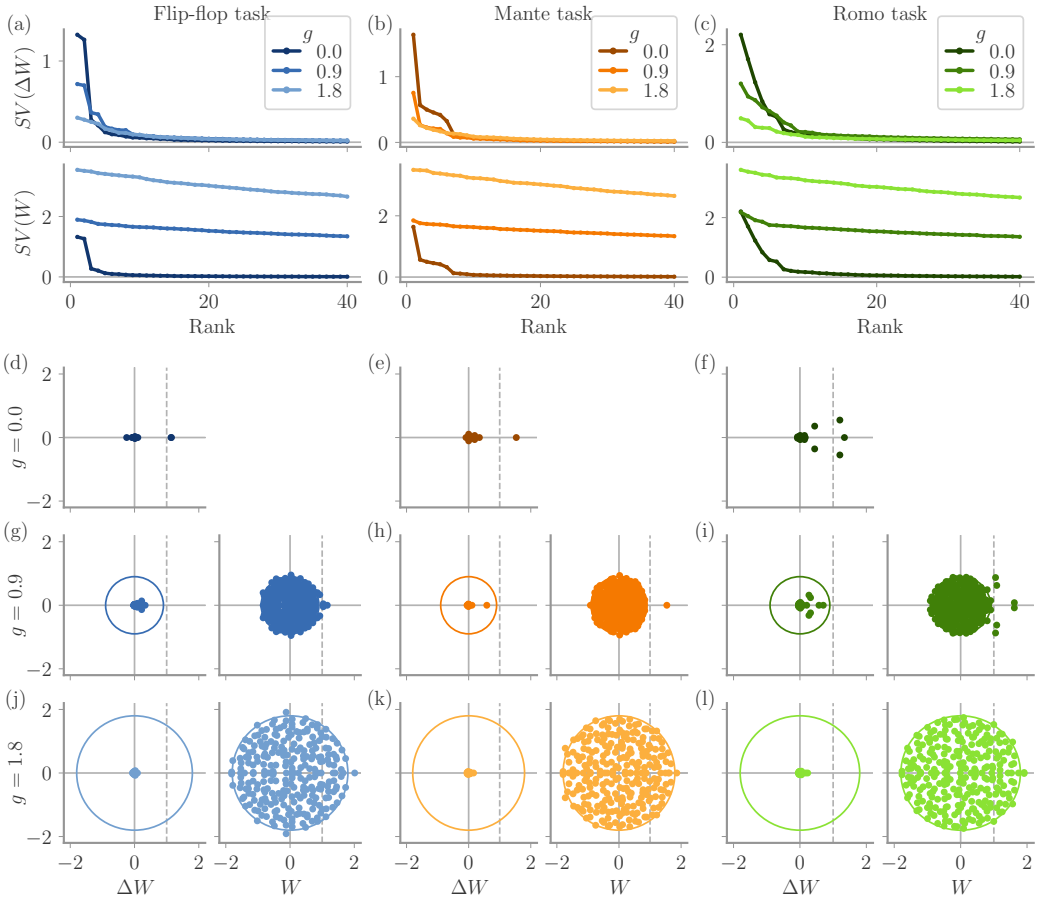

Fig. S2: Singular values (SVs) and eigenvalues (EVs) of RNNs trained for all three tasks with different initial connectivity strength $g \in \{0.0, 0.9, 1.8\}$. **(a-c)** First 40 SVs of the weight changes ΔW (top) and the final weight matrix $W = W_0 + \Delta W$ (bottom). Note the different y-scales: For $g = 0$ (darkest lines), the SVs in both plots are the same. For larger g, the SVs of ΔW tend to become smaller, while those of W increase. **(d-l)** Eigenvalue spectra for ΔW (left) and W (right). The x- and y-coordinates are the real and imaginary part, respectively. For $g = 0$, **(d-f)**, the EVs of ΔW and W are the same. For $g > 0$, we plot the circles with radius g for comparison. Inside this radius, the eigenvalues of W_0 are distributed uniformly [43]. Note that most EVs of W still remain with in this circle. Parameters as in Fig. 1 of the main text, specifically $N = 256$.

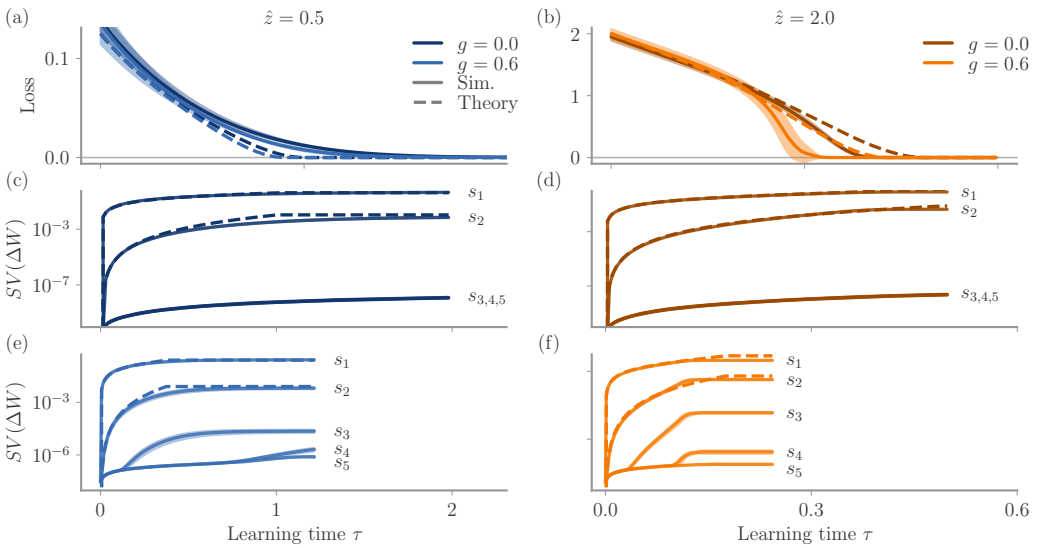

Fig. S3: Evolution of SVs on log scale for the simple task, as a supplement to Fig. 3 of the main text. There, the SVs are shown on a linear scale, which does not allow to observe the evolution of any but the largest SVs. Our theory predicts only the first two SVs (dashed lines); any higher SVs are zero at order $\mathcal{O}(\tau^3)$. **(a,b)** Loss curves as a reference for the learning process. **(c,d)** First five SVs for $g = 0$. Note that the curves of s_3, s_4, and s_5 overlap. **(e,f)** First five SVs for $g = 0.6$.

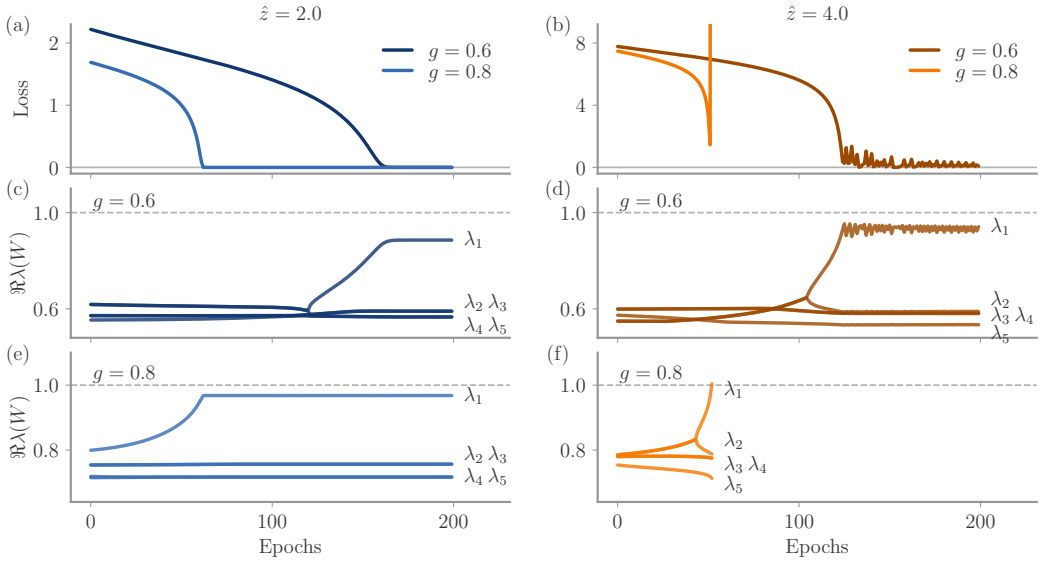

Fig. S4: Exploding gradient when the real part of the largest eigenvalue λ_1 of W crosses 1. For infinitely small learning rate η, the readout z crosses the target value $\hat{z}$ before λ_1 crosses 1, so that learning stops. However, for a finite learning time, z may become larger than an $\hat{z}$, and the divergent gradient may induce oscillations and failure of learning. This failure happens for large target values $\hat{z}$ and initial connectivity strength g, which promote the growth of λ_1. **(a,b)** Loss curves for two different target values and initial connectivity strengths. For $\hat{z} = 4$ and $g = 0.8$, the gradient diverges and learning stops. **(c-f)** Real parts of first five EVs λ_i (order by decreasing real parts). Symbols at the end of each trajectory indicate the eigenvalues. In case of complex conjugates, the two corresponding λ_i are written next to each other. The dashed grey line indicates the critical value $\Re\lambda = 1$ for which the gradients diverge. Parameters: $N = 256$, $\eta = \eta_0(1 - g^2)^2$ with $\eta_0 = 0.002$, 0.001 for $\hat{z} = 2$, 4, respectively. Task parameters as in the main text but with longer trial time, $T = 201$ (so that the network still converges to the fixed point despite the slower time scales).

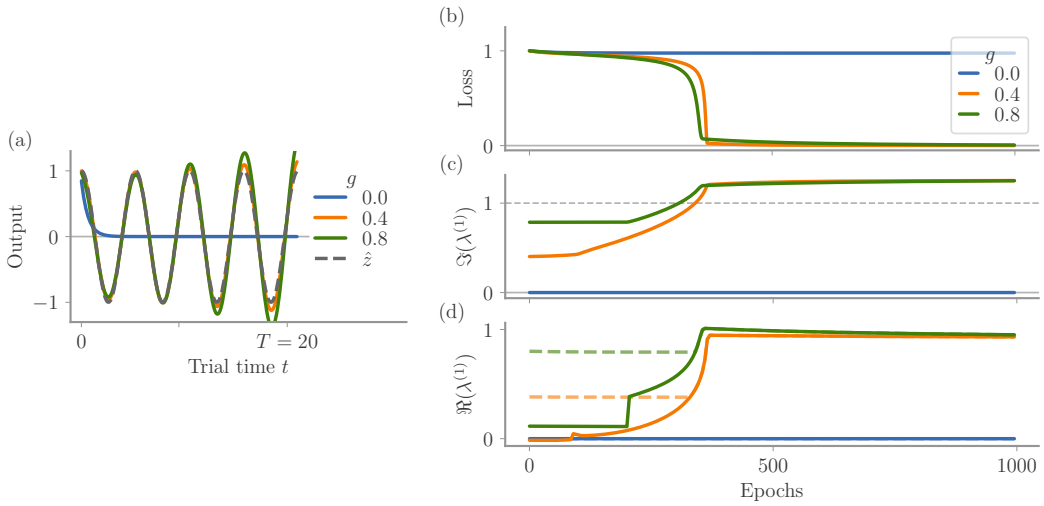

Fig. S5: Example of learning only in presence of initial connectivity. For linear RNNs without initial connectivity, gradient descent-induced connectivity changes are always constructed from the input- and output vectors. If the space of these vectors is too small, learning fails. Here, we take the simple example of a linear network learning a cosine oscillation, starting from a fixed initial condition [see **(a)**]. The initial condition is set by a delta pulse through the input vector; otherwise, the input is zero. We set both input and output vector to $\mathbf{w}$, so that there is only a single vector available. However, creating the necessary complex conjugate eigenvalues needs a rank-two connectivity and hence at least two different directions. Random initial connectivity enlarges the pool of available directions. **(a)** Output of networks at the end of training for three different values of g. Dashed line shows target $\hat{z}(t) = \cos(2\pi f t)$ with frequency $f = 0.2$. Learning failed for $g = 0$. For the other two values, the network finds a slightly unstable solution (perfect marginal stability is not achieved because of the limited trial time $T = 20$). **(b)** Loss over training epochs. **(c)** Imaginary part of largest eigenvalue $\lambda^{(1)}$, sorted by *imaginary parts*. **(d)** Real part of $\lambda^{(1)}$. The dashed lines show the real part of the largest eigenvalue sorted by real parts. For $g = 0$, no nonzero eigenvalue emerges throughout training. Parameters: $N = 256$, $\eta = (0.2, 0.15, 0.05)$ for $g = (0.0, 0.4, 0.8)$, respectively (adapted heuristically for smooth convergence); training for 1000 epochs (batch size = 1, since there is not stochastic part). Simulation step size was reduced to $\Delta t = 0.1$.

S4 Expansion of linear learning

For the simple learning problem, the readout in the limit $t \to \infty$ is given by

$$z = \mathbf{w}^T (I - W)^{-1} \mathbf{m} \,. \tag{20}$$

The loss is quadratic: $L = (\hat{z} - z)^2/2$. The weights change according to the gradient of the loss w.r.t. to recurrent weights W, namely

$$\frac{\mathrm{d}W(\tau)}{\mathrm{d}\tau} = -\frac{\mathrm{d}L}{\mathrm{d}W} = [\hat{z} - z(\tau)] \left[I - W^T(\tau)\right]^{-1} \mathbf{w}\mathbf{m}^T \left[I - W^T(\tau)\right]^{-1} . \tag{21}$$

We expand these dynamics in orders of τ. In the main text, we introduced the expansion

$$W(\tau) = \sum_{k=0}^{\infty} W_k \frac{\tau^k}{k!} \,, \tag{22}$$

with coefficients W_k obtained from $\mathrm{d}^k W/\mathrm{d}\tau^k$ at $\tau = 0$.

S4.1 First order

Because of the independence of W_0, $\mathbf{w}$, and $\mathbf{m}$, the initial readout z_0 is zero, and we directly obtain

$$W_1 = \hat{z} B^T \mathbf{w}\mathbf{m}^T B^T \,, \tag{23}$$

with $B = (I - W_0)^{-1}$. The weight changes linear in τ are

$$\Delta W(\tau) = \mathbf{u}_1 \mathbf{v}_1^T + \mathcal{O}(\tau^2) \,, \tag{24}$$

with

$$\mathbf{u}_1 = a_1 B^T \mathbf{w} \,, \qquad \mathbf{v}_1^T = a_1 \mathbf{m}^T B^T \,, \tag{25}$$

and the coefficient

$$a_1^2 = \tau \hat{z} \,. \tag{26}$$

Note that we chose to split the norm of the rank-one matrix equally between the two vectors, which simplifies notation later on. To compute the readout, we note that W_1 is a rank-one matrix. This allows us to apply the matrix inversion lemma (a.k.a. Sherman-Morrison formula; [12]): The matrix $I - W_0$ is invertible for $g < 1$, and subtracting a rank-one matrix $\mathbf{u}\mathbf{v}^T$ changes its inverse to

$$\left(I - W_0 - \mathbf{u}\mathbf{v}^T\right)^{-1} = B + \frac{1}{1 - \mathbf{v}^T B \mathbf{u}} B\mathbf{u}\mathbf{v}^T B \,, \tag{27}$$

To compute the readout at linear order, we first realize that the scalar product in the denominator in Eq. (27) vanishes:

$$\mathbf{v}_1^T B \mathbf{u}_1 = a_1^2 \mathbf{m}^T B^T B B^T \mathbf{w} = 0 \,. \tag{28}$$

To show this, we note that $\mathbf{m}$ and $\mathbf{w}$ are independent of $M = B^T B B^T$, and therefore

$$\mathbb{E}\left[\mathbf{m}^T M \mathbf{w}\right] = \sum_{i=1}^{N} \sum_{j=1}^{N} \underbrace{\mathbb{E}[m_i w_j]}_{=0} \mathbb{E}[M_{ij}] \,. \tag{29}$$

The variance of $\mathbf{m}^T M \mathbf{w}$ is of order $1/N$, so that in the limit of $N \to \infty$, the term self-averages to zero. With this, we can compute the readout:

$$\begin{aligned} z &= \mathbf{w}^T \left(I - W_0 - \mathbf{u}_1 \mathbf{v}_1^T\right)^{-1} \mathbf{m} \\ &= \underbrace{\mathbf{w}^T B \mathbf{m}}_{=0} + \mathbf{w}^T B \mathbf{u}_1 \, \mathbf{v}_1^T B \mathbf{m} \\ &= \tau \hat{z} \, \mathbf{w}^T B B^T \mathbf{w} \, \mathbf{m}^T B^T B \mathbf{m} \\ &= \tau \hat{z} \beta^2 + \mathcal{O}(\tau^2) \,. \end{aligned} \tag{30}$$

The term $\mathbf{w}^T B B^T \mathbf{w}$ (and likewise $\mathbf{m}^T B^T B \mathbf{m}$) has expectation

$$\mathbb{E}\left[\mathbf{w}^T B B^T \mathbf{w}\right] = \sum_{i=1}^{N} \sum_{j=1}^{N} \underbrace{\mathbb{E}[w_i w_j]}_{=\delta_{ij}/N} \mathbb{E}[(BB^T)_{ij}] = \frac{1}{N} \mathbb{E}[\mathrm{Tr}(BB^T)] = \beta \,. \tag{31}$$

The expected trace $\beta = 1/(1-g^2)$ is computed in Section S5. Due to self-averaging in the limit $N \to \infty$, we omit the expectation.

The singular values of W_1 are the square roots of the eigenvalues of

$$W_1 W_1^T = \hat{z}^2 B^T \mathbf{w}\mathbf{m}^T B^T B \mathbf{m}\mathbf{w}^T B \,. \tag{32}$$

Since this is again a rank-one matrix, we compute the only nonzero eigenvalue via the trace:

$$s^2 = \mathrm{Tr}(W_1 W_1^T) = \hat{z}^2 \mathbf{w}^T B B^T \mathbf{w}\mathbf{m}^T B^T B \mathbf{m} = \hat{z}^2 \beta^2 \,. \tag{33}$$

The singular value, which is also the norm of W_1, is therefore

$$s = ||W_1|| = \hat{z}\beta \,. \tag{34}$$

The learning time τ_1^* is the solution to the equation $z(\tau_1^*) = \hat{z}$, namely $\tau_1^* = 1/\beta^2$. The connectivity changes at this learning time are $\Delta W = \tau_1^* W_1$, with norm $||\Delta W|| = \tau_1^* ||W_1|| = \hat{z}/\beta$.

S4.2 Second order

We again make use of the matrix inversion lemma, Eq. (27), and compute

$$\begin{aligned}
W_2 &= \frac{\mathrm{d}^2 W}{\mathrm{d}\tau^2}\bigg|_{\tau=0} \\
&= \frac{\mathrm{d}}{\mathrm{d}\tau}\left[(\hat{z}-z)\left(I - W_0 - \mathbf{u}_1\mathbf{v}_1^T\right)^{-T}\mathbf{w}\mathbf{m}^T\left(I - W_0 - \mathbf{u}_1\mathbf{v}_1^T\right)^{-T}\right]\bigg|_{\tau=0} \\
&= \frac{\mathrm{d}}{\mathrm{d}\tau}\left[(\hat{z}-z)B^T\left(I + \mathbf{v}_1\mathbf{u}_1^T B^T\right)\mathbf{w}\mathbf{m}^T\left(I + B^T\mathbf{v}_1\mathbf{u}_1^T\right)B^T\right]\bigg|_{\tau=0} \\
&= \frac{\mathrm{d}}{\mathrm{d}\tau}\left[(\hat{z}-\tau\hat{z}\beta^2)B^T(\mathbf{w} + \tau\hat{z}\beta B\mathbf{m})\left(\mathbf{m}^T + \tau\hat{z}\beta\mathbf{w}^T B\right)B^T\right]\bigg|_{\tau=0} \\
&= \hat{z}\beta\, B^T\left[-\beta\mathbf{w}\mathbf{m}^T + \hat{z}\left(\mathbf{w}\mathbf{w}^T B + B\mathbf{m}\mathbf{m}^T\right)\right]B^T.
\end{aligned} \tag{35}$$

We notice that the weight changes up to order $\mathcal{O}(\tau^2)$ can be written as the outer product of two vectors and is thus a rank-one matrix:

$$\begin{aligned}
\Delta W &= \tau W_1 + \frac{\tau^2}{2}W_2 + \mathcal{O}(\tau^3) \\
&= B^T\left[\left(\tau\hat{z} - \frac{\tau^2}{2}\hat{z}\beta^2\right)\mathbf{w}\mathbf{m}^T + \frac{\tau^2}{2}\hat{z}^2\beta\left(\mathbf{w}\mathbf{w}^T B + B\mathbf{m}\mathbf{m}^T\right)\right]B^T + \mathcal{O}(\tau^3) \\
&= B^T(a_2\mathbf{w} + b_2 B\mathbf{m})\left(a_2\mathbf{m}^T + b_2\mathbf{w}^T B^T\right)B^T + \mathcal{O}(\tau^3) \\
&= \mathbf{u}_2\mathbf{v}_2^T + \mathcal{O}(\tau^3)\,,
\end{aligned} \tag{36}$$

with

$$\mathbf{u}_2 = B^T(a_2\mathbf{w} + b_2 B\mathbf{m}) \,, \qquad \mathbf{v}_2^T = \left(a_2\mathbf{m}^T + b_2\mathbf{w}^T B^T\right)B^T \,. \tag{37}$$

The coefficients are implicitly defined by

$$a_2^2 = \tau\hat{z} - \frac{\tau^2}{2}\hat{z}\beta^2 \,, \qquad a_2 b_2 = \frac{\tau^2}{2}\hat{z}^2\beta \,. \tag{38}$$

Note that the correction b_2^2 from completing the square is of order $\mathcal{O}(\tau^3)$.

Similarly to the first order, we can compute the readout z:

$$z_2 = \frac{\mathbf{w}^T B\mathbf{u}_2\mathbf{v}_2^T B\mathbf{m}}{1 - \mathbf{v}_2^T B\mathbf{u}_2} = a_2^2\beta^2 + \mathcal{O}(\tau^3)\,, \tag{39}$$

with

$$\mathbf{w}^T B\mathbf{u}_2 = \mathbf{v}_2^T B\mathbf{m} = a_2\beta \,. \tag{40}$$

The denominator is of order $\mathcal{O}(\tau^2)$ and hence does not contribute to z_2:

$$\begin{aligned}
\mathbf{v}_2^T B\mathbf{u}_2 &= \left(a_2\mathbf{m}^T + b_2\mathbf{w}^T B^T\right)B^T B B^T(a_2\mathbf{w} + b_2 B\mathbf{m}) \\
&= 2a_2 b_2\gamma + \mathcal{O}(\tau^3)\,.
\end{aligned} \tag{41}$$

The random matrix term $\gamma = \mathbf{w}^T B B^T B B^T \mathbf{w} = \beta^4$ is compute Section S5. Terms of the form $\mathbf{m}^T M\mathbf{w}$, with M constructed from B and B^T are zero due to the independence of all three quantities.

S4.3 Third order

Since ΔW at order $\mathrm{O}(\tau^2)$ is a rank-1 matrix, we can use the same formalism as for the second order, cf. Eq. (35). We now only keep terms with τ^2:

$$\begin{aligned}
W_3 &= \frac{\mathrm{d}^3 W}{\mathrm{d}\tau^3}\bigg|_{\tau=0} \\
&= \frac{\mathrm{d}^2}{\mathrm{d}\tau^2}\left[(\hat{z}-z)\left(I-W_0-\mathbf{u}_2\mathbf{v}_2^T\right)^{-T}\mathbf{w}\mathbf{m}^T\left(I-W_0-\mathbf{u}_2\mathbf{v}_2^T\right)^{-T}\right]\bigg|_{\tau=0} \\
&= \frac{\mathrm{d}^2}{\mathrm{d}\tau^2}\left[(\hat{z}-z)B^T\left(I+\mathbf{v}_2\mathbf{u}_2^T B^T\right)\mathbf{w}\mathbf{m}^T\left(I+B^T\mathbf{v}_2\mathbf{u}_2^T\right)B^T\right]\bigg|_{\tau=0} \\
&= \frac{\mathrm{d}^2}{\mathrm{d}\tau^2}\left[(\hat{z}-a_2^2\beta^2)B^T\left[\mathbf{w}+\beta B\left(a_2^2\mathbf{m}+a_2b_2B^T\mathbf{w}\right)\right]\left[\mathbf{m}^T+\beta\left(a_2^2\mathbf{w}^T+a_2b_2\mathbf{m}^TB^T\right)B\right]B^T\right]\bigg|_{\tau=0} \\
&= \hat{z}\beta^2B^T\left[\beta^2\mathbf{w}\mathbf{m}^T-3\hat{z}\beta\left(\mathbf{w}\mathbf{w}^TB+B\mathbf{m}\mathbf{m}^T\right)+2\hat{z}^2B\mathbf{m}\mathbf{w}^TB+\hat{z}^2\left(\mathbf{w}\mathbf{m}^TB^TB+BB^T\mathbf{w}\mathbf{m}^T\right)\right]B^T.
\end{aligned} \tag{42}$$

The changes up to order $\mathcal{O}(\tau^2)$ are now of rank two:

$$\begin{aligned}
\Delta W &= \tau W_1+\frac{\tau^2}{2}W_2+\frac{\tau^3}{6}W_2+\mathcal{O}(\tau^4) \\
&= B^T\Bigg[\left(\tau\hat{z}-\frac{\tau^2}{2}\hat{z}\beta^2+\frac{\tau^3}{6}\hat{z}\beta^4\right)\mathbf{w}\mathbf{m}^T+\left(\frac{\tau^2}{2}\hat{z}^2\beta-\frac{\tau^3}{2}\hat{z}^2\beta^3\right)\left(\mathbf{w}\mathbf{w}^TB+B\mathbf{m}\mathbf{m}^T\right) \\
&\qquad+\frac{\tau^3}{3}\hat{z}^3\beta^2B\mathbf{m}\mathbf{w}^TB+\frac{\tau^3}{6}\hat{z}^3\beta^2\left(\mathbf{w}\mathbf{m}^TB^TB+BB^T\mathbf{w}\mathbf{m}^T\right)\Bigg]B^T+\mathcal{O}(\tau^4) \\
&= B^T\left(a_3\mathbf{w}+b_3B\mathbf{m}+c_3BB^T\mathbf{w}\right)\left(a_3\mathbf{m}^T+b_3\mathbf{w}^TB^T+c_3\mathbf{m}^TB^TB\right)B^T+\hat{b}_3^2B^TB\mathbf{m}\mathbf{w}^TBB^T+\mathcal{O}(\tau^4) \\
&= \mathbf{u}_3\mathbf{v}_3^T+\hat{\mathbf{u}}_3\hat{\mathbf{v}}_3^T+\mathcal{O}(\tau^4),
\end{aligned} \tag{43}$$

with

$$\mathbf{u}_3 = B^T\left(a_3\mathbf{w}+b_3B\mathbf{m}+c_3BB^T\mathbf{w}\right), \tag{44}$$

$$\mathbf{v}_3^T = \left(a_3\mathbf{m}^T+b_3\mathbf{w}^TB+c_3\mathbf{m}^TB^TB\right)B^T, \tag{45}$$

$$\hat{\mathbf{u}}_3 = \hat{b}_3B^TB\mathbf{m}, \tag{46}$$

$$\hat{\mathbf{v}}_3^T = \hat{b}_3\mathbf{w}^TBB^T. \tag{47}$$

The coefficients are implicitly defined by

$$a_3^2 = \tau\hat{z}-\frac{\tau^2}{2}\hat{z}\beta^2+\frac{\tau^3}{6}\hat{z}\beta^4, \tag{48}$$

$$a_3b_3 = \frac{\tau^2}{2}\hat{z}^2\beta-\frac{\tau^3}{2}\hat{z}^2\beta^3, \tag{49}$$

$$a_3c_3 = \frac{\tau^3}{6}\hat{z}^3\beta^2, \tag{50}$$

$$b_3^2 = \frac{(a_3b_3)^2}{a_3^2} = \frac{\tau^3}{4}\hat{z}^3\beta^2, \tag{51}$$

$$\hat{b}_3^2 = \frac{\tau^3}{3}\hat{z}^3\beta^2-b_3^2 = \frac{\tau^3}{12}\hat{z}^3\beta^2. \tag{52}$$

The remaining corrections b_3c_3 and c_3^2 are of order $\mathcal{O}(\tau^4)$ or higher.

The changes ΔW can be written in a compact rank-two form:

$$\Delta W(\tau) = \begin{bmatrix}\mathbf{u}_3 & \hat{\mathbf{u}}_3\end{bmatrix}\begin{bmatrix}\mathbf{v}_3^T\\ \hat{\mathbf{v}}_3^T\end{bmatrix}+\mathcal{O}(\tau^4) = UV^T+\mathcal{O}(\tau^4). \tag{53}$$

With this, we compute the readout, using the matrix inversion lemma [12]:

$$\begin{aligned} z &= \mathbf{w}^T \left(I - W_0 - UV^T\right)^{-1} \mathbf{m} + \mathcal{O}(\tau^4) \\ &= \mathbf{w}^T \left[B + BU \left(I_2 - V^T BU\right)^{-1} V^T B\right] \mathbf{m} + \mathcal{O}(\tau^4) \\ &= \mathbf{w}^T BU \left(I_2 - V^T BU\right)^{-1} V^T B\mathbf{m} + \mathcal{O}(\tau^4)\,. \end{aligned} \tag{54}$$

Here, I_2 is the 2×2 identity matrix. We compute the entries of $V^T BU$ up to $\mathcal{O}(\tau^3)$:

$$\mathbf{v}_3^T B \mathbf{u}_3 = 2a_3 b_3 \gamma\,, \tag{55}$$

$$\mathbf{v}_3^T B \hat{\mathbf{u}}_3 = a_3 \hat{b}_3 \gamma\,, \tag{56}$$

$$\hat{\mathbf{v}}_3^T B \mathbf{u}_3 = a_3 \hat{b}_3 \gamma\,, \tag{57}$$

$$\hat{\mathbf{v}}_3^T B \hat{\mathbf{u}}_3 = 0\,. \tag{58}$$

The factor $\gamma = \beta^4$ is computed in Section S5. Therefore,

$$I_2 - V^T BU = \begin{bmatrix} 1-x & -y \\ -x & 1 \end{bmatrix}\,, \tag{59}$$

with $x = \mathbf{v}_3^T B \mathbf{u}_3$, and $y = \mathbf{v}_3^T B \hat{\mathbf{u}}_3$. Since p and q are $\mathcal{O}(\tau^2)$, we have

$$\left(I_2 - V^T BU\right)^{-1} = \frac{1}{1-x-y^2} \begin{bmatrix} 1 & y \\ y & 1-x \end{bmatrix} = \begin{bmatrix} 1+x & y \\ y & 1 \end{bmatrix} + \mathcal{O}(\tau^4)\,. \tag{60}$$

To complete the evaluation of z, Eq. (54), we further compute $\mathbf{w}^T BU$ and $V^T B\mathbf{m}$:

$$\mathbf{w}^T B \mathbf{u}_3 = \mathbf{v}_3^T B \mathbf{m} = a_3 \beta + c_3 \gamma\,, \tag{61}$$

$$\mathbf{w}^T B \hat{\mathbf{u}}_3 = \hat{\mathbf{v}}_3^T B \mathbf{m} = 0\,. \tag{62}$$

Hence,

$$\begin{aligned} z &= \begin{bmatrix} \mathbf{w}^T B \mathbf{u}_3 & \mathbf{w}^T B \hat{\mathbf{u}}_3 \end{bmatrix} \begin{bmatrix} 1+x & y \\ y & 1 \end{bmatrix} \begin{bmatrix} \mathbf{v}_3^T B \mathbf{m} \\ \hat{\mathbf{v}}_3^T B \mathbf{m} \end{bmatrix} + \mathcal{O}(\tau^4) \\ &= (1+x) \mathbf{w}^T B \mathbf{u}_3 \, \mathbf{v}_3^T B \mathbf{m} + \mathcal{O}(\tau^4) \\ &= (1 + 2a_3 b_3 \gamma) \left(a_3 \beta + c_3 \gamma\right)^2 + \mathcal{O}(\tau^4) \\ &= \Big(1 + \underbrace{2a_3 b_3 \gamma}_{\mathcal{O}(\tau^2)}\Big) \Big(\underbrace{a_3^2 \beta^2}_{\mathcal{O}(\tau)} + \underbrace{2a_3 c_3 \beta \gamma}_{\mathcal{O}(\tau^3)} + \underbrace{c_3^2 \gamma^2}_{\mathcal{O}(\tau^4)}\Big) + \mathcal{O}(\tau^4) \\ &= a_3^2 \beta^2 + 2a_3 c_3 \beta \gamma + 2a_3^2 a_3 b_3 \beta^2 \gamma + \mathcal{O}(\tau^4) \\ &= \hat{z} \left[\beta^2 \tau - \frac{(\beta^2 \tau)^2}{2} + (1 + 8\hat{z}^2 \beta) \frac{(\beta^2 \tau)^3}{6}\right] + \mathcal{O}(\tau^4)\,. \end{aligned} \tag{63}$$

The last lines are based on the implicit definitions of the coefficients a_3, b_3, and c_3 in Eqs. (48) to (50) and $\gamma = \beta^4$.

We end this section with looking at the special case $g = 0$. With $B = I$ and $\beta = 1$, the weight changes Eq. (43) simplify to

$$\begin{aligned} \Delta W &= \left(\tau \hat{z} - \frac{\tau^2}{2} \hat{z} + \frac{\tau^3}{2} \hat{z}\right) \mathbf{w}\mathbf{m}^T + \left(\frac{\tau^2}{2} \hat{z}^2 - \frac{\tau^3}{2} \hat{z}^2\right) \left(\mathbf{w}\mathbf{w}^T + \mathbf{m}\mathbf{m}^T\right) + \frac{\tau^3}{3} \hat{z}^3 \mathbf{m}\mathbf{w}^T + \mathcal{O}(\tau^4) \\ &= \begin{bmatrix} \mathbf{w} & \mathbf{m} \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} \mathbf{w}^T \\ \mathbf{m}^T \end{bmatrix}\,, \end{aligned} \tag{64}$$

with

$$A_{11} = \frac{\hat{z}^2}{2} \left(\tau^2 - \tau^3\right) + \mathcal{O}(\tau^4)\,, \tag{65}$$

$$A_{12} = \hat{z} \left(\tau - \frac{\tau^2}{2} + \frac{\tau^3}{6}(1 + 2\hat{z}^2)\right) + \mathcal{O}(\tau^4)\,, \tag{66}$$

$$A_{21} = \frac{\hat{z}^3 \tau^3}{3} + \mathcal{O}(\tau^4)\,, \tag{67}$$

and $A_{22} = A_{11}$. Note that for $g = 0$, one can write the entire gradient descent dynamics in terms of the matrix 2×2 matrix A:

$$\frac{\mathrm{d}A}{\mathrm{d}\tau} = (\hat{z} - z)\left[I + C^T\right]\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}0 & 1\end{bmatrix}\left[I + C^T\right], \tag{68}$$

with

$$z = \begin{bmatrix}1 & 0\end{bmatrix}[I + C]\begin{bmatrix}0\\1\end{bmatrix} = C_{12}\,, \tag{69}$$

and

$$C = A(I - A)^{-1}\,. \tag{70}$$

With the symmetry $A_{11} = A_{22}$, this equation still has three degrees of freedom, and we were not able to find a closed form solution.

S4.4 Singular values of weight changes

The singular values of ΔW are determined by the eigenvalues of $\Delta W^T \Delta W$ up to order $\mathcal{O}(\tau^3)$. For the rank-two matrix $\Delta W = UV^T$, these are the eigenvalues of the matrix

$$P = V^T V U^T U = \begin{bmatrix}p & q\\ q & r\end{bmatrix}^2 = \begin{bmatrix}p^2 + q^2 & q(p+r)\\ q(p+r) & q^2 + r^2\end{bmatrix}^2\,. \tag{71}$$

As before, we compute the coefficients up to order $\mathcal{O}(\tau^3)$:

$$p = \mathbf{u}_3^T\mathbf{u}_3 = \mathbf{v}_3^T\mathbf{v}_3 = a^2\beta + (b^2 + 2ac)\gamma\,, \tag{72}$$

$$q = \mathbf{u}_3^T\hat{\mathbf{u}}_3 = \mathbf{v}_3^T\hat{\mathbf{v}}_3 = b\hat{b}\gamma\,, \tag{73}$$

$$r = \hat{\mathbf{u}}_3^T\hat{\mathbf{u}}_3 = \hat{\mathbf{v}}_3^T\hat{\mathbf{v}}_3 = \hat{b}^2\gamma\,. \tag{74}$$

The squared singular values are therefore

$$s_\pm^2 = \frac{1}{2}\left(\mathrm{Tr}P \pm \sqrt{(\mathrm{Tr}P)^2 - 4|P|}\right)\,. \tag{75}$$

The terms are of order $p = \mathcal{O}(\tau)$ and $q, r = \mathcal{O}(\tau^3)$, so that

$$\mathrm{Tr} = p^2 + 2q^2 + r^2 = \mathcal{O}(\tau^2)\,, \tag{76}$$

$$|P| = (pr - q^2)^2 = \mathcal{O}(\tau^8)\,. \tag{77}$$

This means that the solutions have different orders:

$$s_+^2 = \mathrm{Tr}P - \frac{|P|}{\mathrm{Tr}P}\,, \tag{78}$$

$$s_-^2 = \frac{|P|}{\mathrm{Tr}P}\,. \tag{79}$$

Taking the square roots and sorting out the orders yields a linear first singular value,

$$s_+ = \frac{\hat{z}}{\beta}\left[\beta^2\tau - \frac{(\beta^2\tau)^2}{2} + \left(1 + \frac{7}{2}\hat{z}^2\beta\right)\frac{(\beta^2\tau)^3}{6}\right]\,. \tag{80}$$

The second singular value is cubic in learning time:

$$s_- = \hat{b}_{(3)}^2\gamma = \hat{z}^3\frac{(\beta^2\tau)^3}{12}\,. \tag{81}$$

S4.5 Effect of shuffling

Shuffling W_0 at the end of training destroys any correlation between W_0 and W_1, while keeping the same statistics. We denote that shuffled W_0 by W_0^s, and the corresponding inverse by $B^s = (1 - W_0^s)^{-1}$.

At first order, the shuffled readout is

$$\begin{aligned} z^s(\tau) &= \mathbf{w}^T (I - W_0^s - \tau_1^* W_1)^{-1} \mathbf{m} \\ &= \mathbf{w}^T \Big[B^s + \frac{1}{1 - \underbrace{\mathbf{v}_1^T B^s \mathbf{u}_1}_{=0}} B^s \mathbf{u}_1 \mathbf{v}_1^T B^s \Big] \mathbf{m} \\ &= \underbrace{\mathbf{w}^T B^s \mathbf{m}}_{=0} + \mathbf{w}^T B^s \mathbf{u}_1 \, \mathbf{v}_1^T B^s \mathbf{m} \\ &= \tau \hat{z} \, \mathbf{w}^T B^s B^T \mathbf{w} \, \mathbf{m}^T B^T B^s \mathbf{m} \\ &= \tau \hat{z} + \mathcal{O}(\tau^2) \, . \end{aligned} \tag{82}$$

The factor β vanishes because

$$\mathbb{E}\left[\mathbf{w}^T B^s B^T \mathbf{w}\right] = \sum_{i=1}^N \sum_{j=1}^N \sum_{k=1}^N \underbrace{\mathbb{E}[w_i w_k]}_{=\delta_{ik}/N} \mathbb{E}[B_{ij}^s] \mathbb{E}[B_{jk}^T] = \frac{1}{N} \sum_{i=1}^N \sum_{j=1}^N \underbrace{\mathbb{E}[B_{ij}^s]}_{=\delta_{ij}\left(1+\frac{1}{N}\right)} \underbrace{\mathbb{E}[B_{ji}^T]}_{=\delta_{ji}\left(1+\frac{1}{N}\right)} = 1 + \mathcal{O}(1/N) \, . \tag{83}$$

Inserting $\tau_1^* = 1/\beta^2$ into Eq. (82) yields $z^s(\tau_1^*) = \hat{z}/\beta^2$. The corresponding loss is

$$L^s = \frac{1}{2}(\hat{z} - z^s(\tau_1^*))^2 = \frac{1}{2}\hat{z}^2 \left(1 - \frac{1}{\beta^2}\right)^2 = L_0 g^4 (2 - g^2)^2 \, , \tag{84}$$

with initial loss $L_0 = \hat{z}^2/2$.

For the third order, not all amplification is lost: Replacing B with B^s in the evaluation of z, Eq. (54) yields

$$\begin{aligned} z^s &= \mathbf{w}^T \left(I - W_0^s - UV^T\right)^{-1} \mathbf{m} + \mathcal{O}(\tau^4) \\ &= \mathbf{w}^T B^s U \left(I_2 - V^T B^s U\right)^{-1} V^T B^s \mathbf{m} + \mathcal{O}(\tau^4) \, . \end{aligned} \tag{85}$$

We compute

$$x^s = \mathbf{v}_3^T B^s \mathbf{u}_3 = a_3 b_3 \left(\mathbf{m}^T B^T B^s B^T B \mathbf{m} + \mathbf{w}^T B B^T B^s B^T \mathbf{w}\right) = 2 a_3 b_3 \beta^2 \, . \tag{86}$$

This is based on

$$\begin{aligned} \mathbb{E}\left[\mathbf{w}^T B B^T B^s B^T \mathbf{w}\right] &= \sum_{i=1}^N \sum_{j=1}^N \underbrace{\mathbb{E}[w_i w_j]}_{=\delta_{ij}/N} \mathbb{E}[(B B^T B^s B^T)_{ij}] \\ &= \frac{1}{N} \sum_{i,j,k,l} \mathbb{E}[B_{ij} B_{jk}^T B_{li}^T] \underbrace{\mathbb{E}[B_{kl}^s]}_{=\delta_{kl}\left(1+\frac{1}{N}\right)} \\ &= \frac{1}{N} \sum_{i,j,k,l} \mathbb{E}[B_{ij} B_{jk}^T B_{ki}^T] \underbrace{\mathbb{E}[B_{kl}^s]}_{=\delta_{kl}\left(1+\frac{1}{N}\right)} \\ &= \frac{1}{N} \mathbb{E}[\mathrm{Tr}(B B^T B^T)] = \beta^2 \, . \end{aligned} \tag{87}$$

Similarly,

$$y^s = \mathbf{v}_3^T B^s \hat{\mathbf{u}}_3 = \hat{\mathbf{v}}_3^T B^s \mathbf{u}_3 = a_3 \hat{b}_3 \beta^2 \, , \qquad \hat{\mathbf{v}}_3^T B^s \hat{\mathbf{u}}_3 = 0 \, , \tag{88}$$

and

$$\mathbf{w}^T B^s \mathbf{u}_3 = \mathbf{v}_3^T B^s \mathbf{m} = a_3 + c_3 \beta^2 \, , \tag{89}$$

$$\mathbf{w}^T B^s \hat{\mathbf{u}}_3 = \hat{\mathbf{v}}_3^T B^s \mathbf{m} = 0 \, . \tag{90}$$

The remaining parts of the calculation of z are similar to the case without shuffling, and the corresponding result to Eq. (63) is:

$$\begin{aligned}
z^s &= \begin{bmatrix}\mathbf{w}^T B^s \mathbf{u}_3 & \mathbf{w}^T B^s \hat{\mathbf{u}}_3\end{bmatrix} \begin{bmatrix}1+x^s & y^s \\ y^s & 1\end{bmatrix} \begin{bmatrix}\mathbf{v}_3^T B^s \mathbf{m} \\ \hat{\mathbf{v}}_3^T B^s \mathbf{m}\end{bmatrix} + \mathcal{O}(\tau^4) \\
&= (1+x^s)\mathbf{w}^T B^s \mathbf{u}_3\, \mathbf{v}_3^T B^s \mathbf{m} + \mathcal{O}(\tau^4) \\
&= (1+2a_3 b_3 \beta^2)\,(a_3 + c_3\beta^2)^2 + \mathcal{O}(\tau^4) \\
&= \Big(1 + \underbrace{2a_3 b_3 \beta^2}_{\mathcal{O}(\tau^2)}\Big)\Big(\underbrace{a_3^2}_{\mathcal{O}(\tau)} + \underbrace{2a_3 c_3 \beta^2}_{\mathcal{O}(\tau^3)} + \underbrace{c_3^2 \beta^4}_{\mathcal{O}(\tau^4)}\Big) + \mathcal{O}(\tau^4) \\
&= a_3^2 + 2a_3 c_3 \beta^2 + 2a_3^2 a_3 b_3 \beta^2 + \mathcal{O}(\tau^4) \\
&= \frac{\hat{z}}{\beta^2}\left[\beta^2\tau - \frac{(\beta^2\tau)^2}{2} + \left(1 + 2\hat{z}^2\left(1+\frac{3}{\beta}\right)\right)\frac{(\beta^2\tau)^3}{6}\right] + \mathcal{O}(\tau^4)\,.
\end{aligned} \tag{91}$$

A comparison with Eq. (63) shows that the first and second order terms are decreased by $1/\beta^2$. However, the third order term has a correction to this, similar to the learning time τ^*.

S5 Traces

Here we compute traces appearing in our learning problem:

$$\frac{1}{N}\mathrm{Tr}(B) = 1\,, \tag{92}$$

$$\frac{1}{N}\mathrm{Tr}(BB^T) = \beta\,, \tag{93}$$

$$\frac{1}{N}\mathrm{Tr}(BBB^T) = \beta^2\,, \tag{94}$$

$$\frac{1}{N}\mathrm{Tr}(BB^TBB^T) = \gamma = \beta^4\,, \tag{95}$$

with $B = (I-J)^{-1}$ and $\beta = \frac{1}{1-g^2}$. The matrix J is a Gaussian random matrix whose entries are drawn independently from $\mathcal{N}(0, g^2/N)$. We denote $W_0 = J$ in order to avoid the extra index.

The traces generally stem from scalar products of the form $\mathbf{a}^T M \mathbf{a}$, where the entries of the random vector $\mathbf{a}$ are drawn from $\mathcal{N}(0, 1/N)$, and the matrix M is independent of $\mathbf{a}$. In particular, any combinations of the matrices B are independent of $\mathbf{a}$, since they only contain the random matrix J. Because of this independence, we have

$$\mathbb{E}\left[\mathbf{a}^T M \mathbf{a}\right] = \sum_{i,j=1}^N \mathbb{E}[a_i M_{ij} a_j] = \sum_{i,j=1}^N \underbrace{\mathbb{E}[a_i a_j]}_{=\delta_{ij}/N}\mathbb{E}[M_{ij}] = \mathbb{E}\left[\frac{\mathrm{Tr}M}{N}\right]\,. \tag{96}$$

Computing the traces above and showing the self-averaging quality of the terms is a matter of counting the number of contributing combinations of J and J^T. Our results are based on expanding B into a geometric series

$$B = I + \sum_{K=1}^{\infty} J^K\,. \tag{97}$$

S5.1 $\mathrm{Tr}(B)$

We start with the trace of B alone:

$$\mathbb{E}\left[\frac{\mathrm{Tr}B}{N}\right] = 1 + \sum_{K=1}^{\infty}\frac{1}{N}\sum_{i=1}^N \mathbb{E}\left[(J^K)_{ii}\right] = 1 + \mathcal{O}\left(\frac{1}{N}\right)\,. \tag{98}$$

We show why the sum vanishes with N. For $K = 1$, the entries J_{ii} have expectation 0. For $K = 2$, the independence of elements of J yields

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}\left[(J^2)_{ii}\right] = \frac{1}{N}\sum_{i,j=1}^{N}\mathbb{E}\left[J_{ij}J_{ji}\right] = \frac{1}{N}\sum_{i\neq j}\underbrace{\mathbb{E}\left[J_{ij}\right]\mathbb{E}\left[J_{ji}\right]}_{=0} + \frac{1}{N}\sum_{i}\underbrace{\mathbb{E}\left[J_{ii}^2\right]}_{=g^2/N} = \mathcal{O}\left(\frac{1}{N}\right). \tag{99}$$

The second term vanishes because there are only N terms, but the factor $1/N$ before the sum and the contribution g^2/N together yield $1/N^2$. This observation can be generalized to higher K:

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}\left[(J^K)_{ii}\right] = \frac{1}{N}\sum_{i_1,i_2,\ldots,i_K}\mathbb{E}\left[J_{i_1i_2}J_{i_2i_3}\ldots J_{i_Ki_1}\right] = \frac{1}{N}\sum_{i}\underbrace{\mathbb{E}\left[J_{ii}^{K/2}\right]}_{=\mathcal{O}\left(N^{K/2}\right)} = \mathcal{O}\left(\frac{1}{N^{K/2-1}}\right). \tag{100}$$

There are K different indices. Because each index appears once as a first and once as a second index, the attempt to form pairs directly results in setting all indices equal.

S5.2 $\mathrm{Tr}(BB^T)$

The situation changes when introducing B^T. We can write

$$BB^T = \sum_{K,L=0}^{\infty} J^K J^{TL}, \tag{101}$$

where the transpose T and power L commute. We compute the trace again term by term, starting at $K = L = 1$:

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}\left[(JJ^T)_{ii}\right] = \frac{1}{N}\sum_{i,j}\mathbb{E}\left[J_{ij}J_{ji}^T\right] = \frac{1}{N}\sum_{i,j}\underbrace{\mathbb{E}\left[J_{ij}^2\right]}_{=g^2/N} = g^2. \tag{102}$$

For general $K, L \geq 1$, we have

$$\begin{aligned}\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}\left[(J^KJ^{TL})_{ii}\right] &= \frac{1}{N}\sum_{i_1,\ldots i_K}\sum_{j_1,\ldots j_L}\mathbb{E}\left[J_{i_1i_2}J_{i_2i_3}\ldots J_{i_Kj_1}J_{j_1j_2}^TJ_{j_2j_3}^T\ldots J_{j_Li_1}^T\right] \\ &= \frac{1}{N}\sum_{i_1,\ldots i_K}\sum_{j_1,\ldots j_L}\mathbb{E}\left[J_{i_1i_2}J_{i_2i_3}\ldots J_{i_Kj_1}J_{j_2j_1}J_{j_3j_2}\ldots J_{j_1i_L}^T\right].\end{aligned} \tag{103}$$

We need to form pairs of indices. To simplify the discussion, we write the sequence of index pairs alone:

$$\begin{bmatrix}i_1\\i_2\end{bmatrix}\begin{bmatrix}i_2\\i_3\end{bmatrix}\ldots\begin{bmatrix}i_{K-1}\\i_K\end{bmatrix}\begin{bmatrix}i_K\\j_1\end{bmatrix}\begin{bmatrix}j_2\\j_1\end{bmatrix}\begin{bmatrix}j_3\\j_2\end{bmatrix}\ldots\begin{bmatrix}i_1\\j_L\end{bmatrix}. \tag{104}$$

There are $K + L$ indices, and we need to form $(K + L)/2$ distinct pairs of index pairs. Each index constraint reduces the entire term by a factor of $1/N$. Because of the additional factor $1/N$ in front of the sum, we can have only $(K + L)/2 - 1$ index constraints. The question becomes one of counting the number of possible combinations.

The expression above indicates that the only relevant term needs to constrain $i_K = j_2$. Under this condition, we have

$$\begin{bmatrix}i_1\\i_2\end{bmatrix}\begin{bmatrix}i_2\\i_3\end{bmatrix}\ldots\begin{bmatrix}i_{K-1}\\i_K\end{bmatrix}\begin{bmatrix}i_K\\j_1\end{bmatrix}\begin{bmatrix}i_K\\j_1\end{bmatrix}\begin{bmatrix}j_3\\i_K\end{bmatrix}\ldots\begin{bmatrix}i_1\\j_L\end{bmatrix}. \tag{105}$$

The two middle terms drop and the new middle pairs show the same configuration. One can proceed iteratively with this scheme until reaching the right or left end (depending on $min(K, L)$). In fact, if $L > K$, then

$$\underbrace{\begin{bmatrix}i_1\\i_2\end{bmatrix}\begin{bmatrix}i_2\\i_3\end{bmatrix}\ldots\begin{bmatrix}i_{K-1}\\i_K\end{bmatrix}\begin{bmatrix}i_K\\j_1\end{bmatrix}\begin{bmatrix}i_K\\j_1\end{bmatrix}\begin{bmatrix}i_{K-1}\\i_K\end{bmatrix}\ldots\begin{bmatrix}i_1\\i_2\end{bmatrix}}_{\text{paired with } K-1 \text{ constraints}}\begin{bmatrix}j_{K+1}\\i_1\end{bmatrix}\begin{bmatrix}j_{K+2}\\j_{K+1}\end{bmatrix}\ldots\begin{bmatrix}i_1\\j_L\end{bmatrix}. \tag{106}$$

The non-paired terms need $L-K$ additional constraints, so that the entire term only gives a contribution of $\mathcal{O}(1/N^{(L-K)/2-1})$. This and a similar argument for $K>L$ shows that we need $K=L$. In that case, there are $K-1=(K+L)/2-1$ constraints and the term contributes at order $\mathcal{O}(1)$. We summarize with

$$\mathbb{E}\left[\frac{\mathrm{Tr}(J^K J^{TL})}{N}\right] = g^{2K}\delta_{KL} + \mathcal{O}\left(\frac{1}{N}\right) . \tag{107}$$

For the entire matrix BB^T, this leads to

$$\mathbb{E}\left[\frac{\mathrm{Tr}(BB^T)}{N}\right] = \sum_{K,L=1}^{\infty} \mathbb{E}\left[\frac{\mathrm{Tr}(J^K J^{TL})}{N}\right] = \sum_{K=1}^{\infty} g^{2K} + \mathcal{O}\left(\frac{1}{N}\right) = \frac{1}{1-g^2} + \mathcal{O}\left(\frac{1}{N}\right) . \tag{108}$$

Note that the correction terms remain finite under the infinite sums for K and L because they scale with g^{K+L} and we chose $g<1$.

S5.3 $\mathrm{Tr}(BBB^T)$

For $\mathrm{Tr}(BBB^T)$, the arguments go in parallel to the previous discussion. Indeed, we have

$$\begin{aligned}
\mathbb{E}\left[\frac{\mathrm{Tr}(BBB^T)}{N}\right] &= \sum_{K,L,M=1}^{\infty} \mathbb{E}\left[\frac{\mathrm{Tr}(J^K J^L J^{TM})}{N}\right] \\
&= \sum_{K,L,M=1}^{\infty} g^{2M}\delta_{K+L,M} \\
&= \sum_{M=0}^{\infty} g^{2M} \underbrace{\sum_{K=0}^{\infty}\sum_{L=0}^{\infty} \delta_{K+L,M}}_{=\sum_{K=0}^{M} 1} \\
&= \sum_{M=0}^{\infty} g^{2M}(M+1) \\
&= \frac{1}{(1-g^2)^2} .
\end{aligned} \tag{109}$$

plus an order $\mathcal{O}(1/N)$ correction.

S5.4 $\mathrm{Tr}(BB^TBB^T)$

For $\mathbb{E}[\mathrm{Tr}(BB^TBB^T)/N]$, we first compute trace of the components $J^i J^{Tj} J^k J^{Tl}$. Similar to the cases discussed before, we need to constrain indices to create equal index pairs. The index pairs before any constraints can be written as

$$\begin{bmatrix} i_1 & i_2 & \dots & i_i & j_2 & j_3 & \dots & k_1 & k_1 & k_2 & \dots & k_k & l_2 & l_3 & \dots & i_1 \\ i_2 & i_3 & \dots & j_1 & j_1 & j_2 & \dots & j_j & k_2 & k_3 & \dots & l_1 & l_1 & l_2 & \dots & l_l \end{bmatrix} . \tag{110}$$

There are $n=i+j+k+l$ summation indices, and each pair contributes with a factor g^2/N. Together with the additional factor $1/N$, we can thus have at most $n/2-1$ constraints. Note that like before, the number of transposed matrices must equal that of the non-transposed, $i+k=j+l$, so that n is even. A smaller number of constraints is not sufficient, so that the question becomes: How many different sets of $n/2-1$ constraints lead to $n/2$ pairs of index pairs?

We start with $i=j=k=l=1$. The corresponding index pairs are

$$\begin{bmatrix} i_1 & k_1 & k_1 & i_1 \\ j_1 & j_1 & l_1 & l_1 \end{bmatrix} . \tag{111}$$

One can see that there are two possible combinations to create two pairs: $i_1=k_1$ and $j_1=l_1$, which yield

$$\begin{bmatrix} i_1 & i_1 & i_1 & i_1 \\ j_1 & j_1 & l_1 & l_1 \end{bmatrix} , \qquad \begin{bmatrix} i_1 & k_1 & k_1 & i_1 \\ j_1 & j_1 & j_1 & j_1 \end{bmatrix} . \tag{112}$$

Therefore, there are 2 combinations. An index-counting argument like before generalizes this result, showing that the number of combinations is equal to

$$c_{ijkl} = 1 + \min(i, j, k, l)\,. \tag{113}$$

We prove this statement by induction: Let $i = \min(i, j, k, l)$ without loss of generality (since the trace is cyclic). We rewrite the index pairs Eq. (110) and color cases were two upper or lower indices are equal without any constraints:

$$\begin{bmatrix} i_1 & i_2 & \dots & i_i & j_2 & j_3 & \dots & k_1 & k_1 & k_2 & \dots & k_k & l_2 & l_3 & \dots & i_1 \\ i_2 & i_3 & \dots & j_1 & j_1 & j_2 & \dots & j_j & k_2 & k_3 & \dots & l_1 & l_1 & l_2 & \dots & l_l \end{bmatrix}. \tag{114}$$

We next separate two cases: Case 1, $i_i = j_2$, and Case 2, $i_i \neq j_2$. In Case 1, the index pairs with the blue js above become equal:

$$\begin{bmatrix} i_1 & i_2 & \dots & i_{i-1} & j_2 & j_2 & j_3 & \dots & k_1 & k_1 & k_2 & \dots & k_k & l_2 & l_3 & \dots & i_1 \\ i_2 & i_3 & \dots & j_2 & j_1 & j_1 & j_2 & \dots & j_j & k_2 & k_3 & \dots & l_1 & l_1 & l_2 & \dots & l_l \end{bmatrix}. \tag{115}$$

We can take these pairs out, and the remaining indices read

$$\begin{bmatrix} i_1 & i_2 & \dots & i_{i-1} & j_3 & j_4 & \dots & k_1 & k_1 & k_2 & \dots & k_k & l_2 & l_3 & \dots & i_1 \\ i_2 & i_3 & \dots & j_2 & j_2 & j_3 & \dots & j_j & k_2 & k_3 & \dots & l_1 & l_1 & l_2 & \dots & l_l \end{bmatrix}, \tag{116}$$

where we colored the j_2 blue again. We now have $(i', j', k', l') = (i-1, j-1, k, l)$ indices, with $\min(i', j', k', l') = i - 1$. According to our induction hypothesis, there are $c_{i'j'k'l'} = 1 + i - 1 = i$ different sets of $n/2 - 2$ constraints. Adding the constraint of Case 1, $i_i = j_2$ yields the expected number of $n/2 - 1$ constraints.

It remains to show that Case 2 allows for exactly one set of $n/2 - 1$ constraints. Because $i_i \neq j_2$ in Eq. (114), we need to have a pair at the red i_1; otherwise, one needs $n/2$ constraints. The pair at i_1 requires $l_l = i_2$, and dropping the newly formed pair yields

$$\begin{bmatrix} i_2 & i_3 & \dots & i_i & j_2 & j_3 & \dots & k_1 & k_1 & k_2 & \dots & k_k & l_2 & l_3 & \dots & i_2 \\ i_2 & i_3 & \dots & j_1 & j_1 & j_2 & \dots & j_j & k_2 & k_3 & \dots & l_1 & l_1 & l_2 & \dots & l_{l-1} \end{bmatrix}. \tag{117}$$

We follow the same argumentation, constraining $l_{l-1} = i_3, \dots, l_{2+l-i} = i_i$. We arrive at

$$\begin{bmatrix} i_i & j_2 & j_3 & \dots & k_1 & k_1 & k_2 & \dots & k_k & l_2 & \dots & i_i \\ j_1 & j_1 & j_2 & \dots & j_j & k_2 & k_3 & \dots & l_1 & l_1 & \dots & l_{1+l-i} \end{bmatrix}. \tag{118}$$

Further setting $l_{1+l-i} = j_1$ and dropping the induced pair leads to

$$\begin{bmatrix} j_2 & j_3 & \dots & k_1 & k_1 & k_2 & \dots & k_k & l_2 & \dots & j_1 \\ j_1 & j_2 & \dots & j_j & k_2 & k_3 & \dots & l_1 & l_1 & \dots & l_{l-i} \end{bmatrix}. \tag{119}$$

This is equal to the case $J^{Tj} J^k J^{T(l-i)}$. By the cyclic nature of the trace, this is equal to the case $J^{T(j+l-i)} J^k$. As discussed above, Section S5.2, only one set of $(j + l - i + k)/2 - 1$ constraints leads to a full separation into pairs. Note that if $i = l$, the last set of indices, Eq. (119), looks slightly different, but yields the same result.

Counting the number of constraints in Case 2 yields $1 + i - 2 + 1 + (j + l - i + k)/2 - 1 = n/2 - 1$. Since there is no other combination for Case 2, the total number of constraint combinations is precisely $i + 1 = 1 + \min(i, j, k, l) = c_{ijkl}$.

We return to the trace, which contains the factors g^2:

$$\mathbb{E}\left[\frac{\mathrm{Tr}(J^i J^{Tj} J^k J^{Tl})}{N}\right] = g^{2(i+k)}\, \delta_{i+k,j+l}\, c_{ijkl}\,. \tag{120}$$

We now evaluate the sums over i, j, k, l, starting with fixed i:

$$\sum_{j,k,l=1}^{\infty} \mathbb{E}\left[\frac{\mathrm{Tr}(J^i J^{Tj} J^k J^{Tl})}{N}\right] = \sum_{j,k,l=1}^{\infty} g^{2(i+k)}\, \delta_{i+k,j+l}\, c_{ijkl}\,. \tag{121}$$

We split the summation into different regimes:

$$\begin{aligned}\sum_{j,k,l=1}^{\infty} g^{2(i+k)}\,\delta_{i+k,j+l}\,c_{ijkl} &= \sum_{\substack{j,l\\ j+l\geq i}}\sum_{k=1}^{\infty} g^{2(i+k)}\,\delta_{k,j+l-i}\,c_{ijkl}\\ &= \sum_{\substack{j,l\\ j\geq i\\ l\geq i}} g^{2(j+l)}\,(i+1) + \sum_{\substack{j,l\\ j+l\geq i\\ \min(j,l)<i}} g^{2(j+l)}\,c_{ij(j+l-i)l} \qquad (122)\\ &= a+b+c+d\,,\end{aligned}$$

where we split the second summand of the second-last line into two parts. The parts are:

$$a = \sum_{\substack{j,l\\ j\geq i\\ l\geq i}} g^{2(j+l)}\,(i+1) = (i+1)\sum_{j=i}^{\infty}(i+1)\left(\sum_{j=i}^{\infty} g^{2j}\right)^2 = \frac{(i+1)g^{4i}}{(1-g^2)^2}\,, \qquad (123)$$

$$b = \sum_{\substack{j,l\\ j<i\\ l<i\\ j+l\geq i}} g^{2(j+l)}\,(j+l-i+1) = \frac{g^{2i}}{(1-g^2)^3}\left[i(1+g^{2i})(1-g^2) - (1-g^{2i})(1+g^2)\right]\,, \qquad (124)$$

$$c = \sum_{\substack{j,l\\ j\geq i\\ l<i}} g^{2(j+l)}\,(l+1) = \sum_{j=i}^{\infty}\sum_{l=0}^{i-1} g^{2l}\,(l+1) = \frac{g^{2i}}{(1-g^2)^3}\left[1-g^{2i}-g^{2i}i(1-g^2)\right]\,, \qquad (125)$$

$$d = \sum_{\substack{j,l\\ j<i\\ l\geq i}} g^{2(j+l)}\,(j+1) = c\,. \qquad (126)$$

Joining all terms yields

$$\sum_{j,k,l=1}^{\infty}\mathbb{E}\left[\frac{\mathrm{Tr}(J^iJ^{Tj}J^kJ^{Tl})}{N}\right] = \frac{(i+1)g^{2i}}{(1-g^2)^2}\,. \qquad (127)$$

Finally, we sum over i:

$$\mathbb{E}\left[\frac{\mathrm{Tr}(BB^TBB^T)}{N}\right] = \sum_{i,j,k,l=1}^{\infty}\mathbb{E}\left[\frac{\mathrm{Tr}(J^iJ^{Tj}J^kJ^{Tl})}{N}\right] = \sum_{i=0}^{\infty}\frac{(i+1)g^{2i}}{(1-g^2)^2} = \frac{1}{(1-g^2)^4}\,. \qquad (128)$$

We return to the trace, which is therefore

$$\mathbb{E}\left[\frac{\mathrm{Tr}(J^iJ^{Tj}J^kJ^{Tl})}{N}\right] = g^{2(i+k)}\,\delta_{i+k,j+l}\,c_{ijkl}\,. \qquad (129)$$

We now evaluate the sums over i,j,k,l, starting with fixed i:

$$\sum_{j,k,l=1}^{\infty}\mathbb{E}\left[\frac{\mathrm{Tr}(J^iJ^{Tj}J^kJ^{Tl})}{N}\right] = \sum_{j,k,l=1}^{\infty} g^{2(i+k)}\,\delta_{i+k,j+l}\,c_{ijkl}\,. \qquad (130)$$

We split the summation into different regimes:

$$\begin{aligned}\sum_{j,k,l=1}^{\infty} g^{2(i+k)}\,\delta_{i+k,j+l}\,c_{ijkl} &= \sum_{\substack{j,l\\ j+l\geq i}}\sum_{k=1}^{\infty} g^{2(i+k)}\,\delta_{k,j+l-i}\,c_{ijkl}\\ &= \sum_{\substack{j,l\\ j\geq i\\ l\geq i}} g^{2(j+l)}\,(i+1) + \sum_{\substack{j,l\\ j+l\geq i\\ \min(j,l)<i}} g^{2(j+l)}\,c_{ij(j+l-i)l} \qquad (131)\\ &= a+b+c+d\,,\end{aligned}$$

where we split the second summand of the second-last line into two parts. The parts are:

$$a = \sum_{\substack{j,l \\ j\geq i \\ l\geq i}} g^{2(j+l)} (i+1) = (i+1) \sum_{j=i}^{\infty} (i+1) \left(\sum_{j=i}^{\infty} g^{2j} \right)^2 = \frac{(i+1)g^{4i}}{(1-g^2)^2} \,, \tag{132}$$

$$b = \sum_{\substack{j,l \\ j<i \\ l<i \\ j+l\geq i}} g^{2(j+l)} (j+l-i+1) = \frac{g^{2i}}{(1-g^2)^3} \left[i(1+g^{2i})(1-g^2) - (1-g^{2i})(1+g^2) \right] \,, \tag{133}$$

$$c = \sum_{\substack{j,l \\ j\geq i \\ l<i}} g^{2(j+l)} (l+1) = \sum_{j=i}^{\infty} \sum_{l=0}^{i-1} g^{2l} (l+1) = \frac{g^{2i}}{(1-g^2)^3} \left[1 - g^{2i} - g^{2i} i (1-g^2) \right] \,, \tag{134}$$

$$d = \sum_{\substack{j,l \\ j<i \\ l\geq i}} g^{2(j+l)} (j+1) = c \,. \tag{135}$$

Joining all terms yields

$$\sum_{j,k,l=1}^{\infty} \mathbb{E} \left[\frac{\mathrm{Tr}(J^i J^{Tj} J^k J^{Tl})}{N} \right] = \frac{(i+1)g^{2i}}{(1-g^2)^2} \,. \tag{136}$$

Finally, we sum over i:

$$\mathbb{E} \left[\frac{\mathrm{Tr}(BB^TBB^T)}{N} \right] = \sum_{i,j,k,l=1}^{\infty} \mathbb{E} \left[\frac{\mathrm{Tr}(J^i J^{Tj} J^k J^{Tl})}{N} \right] = \sum_{i=0}^{\infty} \frac{(i+1)g^{2i}}{(1-g^2)^2} = \frac{1}{(1-g^2)^4} \,, \tag{137}$$

which is the statement we wanted to prove.

S6 Details of sentiment analysis task

For the sentiment analysis task in the results section, we trained a 2-layer LSTM model on the Standford Sentiment Treebank with binary labels (SST-2) [31]. The dataset consists of sentences from movie reviews which are labeled positive or negative. Sentences have on average 20 words, and there are 6920 training and 872 validation examples. We tokenized the sentences with the scaCy tokenizer [44]. We further used a pretrained word embedding (GloVe, [47]) with dimension $N_{\mathrm{in}} = 100$. The word embedding was kept fixed during training.

Each LSTM layer had $N = 256$ units. All weights and biases were initialized from the uniform distribution $\mathcal{U}(-a, a)$, where $a = \sqrt{1/N}$, except for input weights of layer 1, where $a = \sqrt{1/N_{\mathrm{in}}}$. During training, all weights and biases were updated with Adam on a binary cross entropy loss, as implemented in PyTorch [46]. We set the learning rate to $0.01/N$, and all other parameters at their default values. We additionally applied dropout with probability 0.5 to all hidden states. We trained the model for 500 epochs, each epoch iterating over the entire data set with batches of 64 sentences.

To evaluate the performance after truncation, we separated the weights into recurrent and input weights. Because the LSTM for the four different gates are concatenated, the input weights of layer 1 have shape $4N \times N_{\mathrm{in}}$, all other weights have shape $4N \times N$. We simultaneously truncated the recurrent weights of both layers and the input weights of layer 2, i.e., all blocks with shape $4N \times N$ This specific choice did not alter the qualitative result, namely that truncating the changes ΔW and ΔU at a given rank produces a much smaller decrease in performance than truncating the full weights $W = W_0 + \Delta W$ and $U = U_0 + \Delta U$.

Note that we chose the learning rate to be sufficiently small so that learning dynamics were smooth. With higher learning rates and rugged loss curves, we observed that changes ΔW would replace the initial connectivity, and the effective rank was much higher. Further note that other hyperparameters, such as L2 regularization on the weights, may also change the picture.

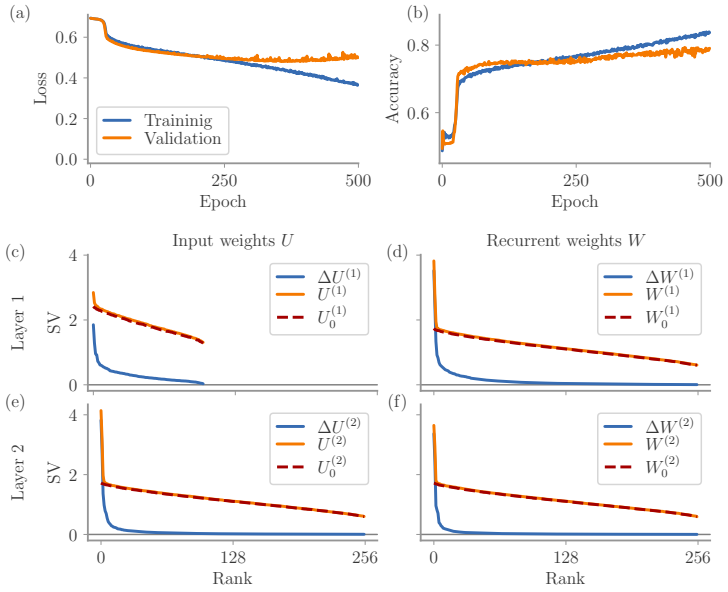

Fig. S6: Details for 2-layer LSTM model trained on a sentiment analysis task. **(a, b)** Training and validation loss and accuracy over epochs. **(c-f)** Singular values (SVs) of the input and recurrent weights in both layers.

Supplementary References

[43] Jean Ginibre. Statistical ensembles of complex, quaternion, and real matrices. *Journal of Mathematical Physics*, 6(3):440–449, 1965.

[44] Matthew Honnibal and Mark Johnson. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1373–1378, 2015.

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

[46] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[47] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

[31] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.