Learning Reveals Invisible Structure in Low-Rank RNNs

Yoav Ger
Technion*
yoav.ger@campus.technion.ac.il

Omri Barak
Technion*
omri.barak@gmail.com

Abstract

Learning in neural systems arises from synaptic changes that reshape the representations underlying behavior. While low-rank recurrent neural networks (RNNs) have emerged as a powerful framework for linking connectivity to function, a theoretical understanding of their learning process remains elusive. Here, we extend the low-rank framework from activity to learning by deriving gradient-descent dynamics directly in a reduced overlap space. We formulate a closed-form, low-dimensional system of ODEs that governs learning in this space, exact for linear RNNs and asymptotically exact for nonlinear RNNs in the large-N Gaussian limit. Central to our analysis is a distinction between two classes of overlaps: *loss-visible* overlaps, which fully determine network activity, output, and loss, and *loss-invisible* overlaps, which do not affect function but are required to describe learning. We illustrate the consequences of this decomposition through two phenomena. First, we show that learning can serve as a perturbation that exposes differences in connectivity between functionally equivalent networks. Second, we show that loss-invisible overlaps can act as memory variables that encode training history, and characterize the conditions under which this occurs. Finally, we present several testable predictions for biological learning experiments derived from our theory.

1 Introduction

Learning is a hallmark of intelligent systems, whether biological or artificial [1, 2, 3]. In neuroscience, a central paradigm posits that learning arises from synaptic changes within neural circuits that reshape the internal dynamics (i.e., activity) and representations underlying behavior [4, 5]. However, directly linking microscopic circuit-level plasticity to macroscopic behavioral outcomes remains a fundamental challenge [6, 7]. One possible reason for this difficulty, at least in theory, lies in the disparity of scales [8]. Adaptation occurs in a high-dimensional space of synaptic parameters (analogous to the overparameterized weight space in artificial neural networks), while the resulting functions or behaviors are much lower-dimensional, often by orders of magnitude. This mismatch renders the mapping from function to connectivity intrinsically ill-posed [9], raising fundamental questions about degeneracy [10, 11, 12] and identifiability [13, 14, 15] in neural systems.

A promising framework for addressing these challenges is low-rank recurrent neural networks. In these models, recurrent connectivity is constrained to be low-rank, such that the effective mapping from connectivity to network dynamics and function is fully described by a small set of macroscopic overlap variables [16]. This reduction has made low-rank RNNs a powerful model for studying recurrent computation, including an analysis of the networks' dynamical properties [17, 18], the design of engineered networks that implement prescribed computations [19, 20, 21], and work showing how low-rank structure emerges through training [22, 23, 24]. While recent work has begun to analyze the learning dynamics of RNNs [25, 26], these approaches have largely been developed

*Ruth and Bruce Rappaport Faculty of Medicine and Network Biology Research Laboratory
Technion – Israel Institute of Technology

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

outside the low-rank framework. Consequently, it remains unclear whether the overlap view—so successful in describing network function—can be extended to account for learning while retaining a similar low-dimensional description.

To bridge this gap, we extend the low-rank framework from network activity to learning dynamics. By expressing gradient descent updates directly in terms of scalar overlaps, we obtain a closed-form, low-dimensional description of learning. This derivation reveals that the resulting dynamics are not equivalent to naive gradient descent in overlap space, but are rather shaped by a preconditioning metric that captures the geometry of the high-dimensional parameter space. Interestingly, for low-rank RNNs, this metric can be computed explicitly and depends on additional overlaps beyond those that determine the current function, thereby revealing structural constraints on learning that are invisible at the level of function alone.

Our **contributions** can be summarized as follows:

- **Technical:** We extend the low-rank framework to learning by deriving a closed-form system of ODEs for the overlap dynamics in low-rank RNNs. These are exact in the linear case and asymptotically exact in the Gaussian nonlinear case as $N \to \infty$, providing, to the best of our knowledge, the first analytical description of learning in nonlinear task-trained RNNs.
- **Conceptual:** A key consequence of our technical derivation is a partition of connectivity into two groups: *loss-visible* overlaps, which fully determine the network's activity, output, and loss, and *loss-invisible* overlaps, which are functionally silent yet shape the trajectory of learning. In our setting, the boundary between these groups is determined by the activation function (linear/nonlinear), although it may generally also depend on the architecture, input–output map, and task.
- **Implications:** We illustrate the implications of this partition through two central phenomena of neural learning: (i) degeneracy – networks with identical function can have distinct connectivity, with learning resolving this ambiguity. Thus, observing how a system learns can serve as a non-invasive probe of underlying structure, an idea we term *perturbation-by-learning*. (ii) memory – *loss-invisible* overlaps can serve as memory variables, encoding aspects of past training history without affecting network function. We show that memory is generally unreliable in linear networks, with its presence depending on the learning rule, while in nonlinear networks it emerges more readily.

2 Preliminaries

We study a high-dimensional RNN trained via gradient descent. While our framework is general, we develop it here for RNNs, a canonical model in theoretical neuroscience [19, 27, 28]. Throughout, bold lowercase letters denote vectors (e.g., $\boldsymbol{z}$), bold uppercase letters denote matrices (e.g., $\boldsymbol{W}$), and plain symbols denote scalars. For two vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^N$, we define their scaled overlap by $\sigma_{vu} = \frac{1}{N}\boldsymbol{v}^\top \boldsymbol{u}$ and the squared norm $\|\boldsymbol{v}\|^2 = \frac{1}{N}\boldsymbol{v}^\top \boldsymbol{v}$, so both remain $O(1)$ as $N \to \infty$. Within-episode (trial) time is indexed by t, and learning time (across episodes) by τ. Gradients with respect to parameters $\boldsymbol{\theta}$ are written $\nabla_{\boldsymbol{\theta}}$, and $\dot{\boldsymbol{\theta}} = d\boldsymbol{\theta}/d\tau$ denotes differentiation with respect to learning time.

RNN model We consider a rate-based RNN with N neurons (Fig. 1a, top). Its continuous-time dynamics and readout are

$$\dot{\boldsymbol{h}}(t) = -\boldsymbol{h}(t) + \frac{1}{\sqrt{N}}\boldsymbol{W}\,\phi(\boldsymbol{h}(t)) + \boldsymbol{m}\,x(t), \qquad \hat{y}(t) = \frac{1}{N}\,\boldsymbol{z}^\top \phi(\boldsymbol{h}(t)) \tag{1}$$

where $\boldsymbol{h}(t) \in \mathbb{R}^N$ is the hidden state, $\phi(\cdot)$ is an element-wise activation function, and $\boldsymbol{W} \in \mathbb{R}^{N\times N}$ is the recurrent connectivity matrix. For simplicity, we focus on a single-input, single-output network (extensions to multiple inputs/outputs are straightforward). The scalar input $x(t)$ enters through $\boldsymbol{m} \in \mathbb{R}^N$, and the scalar output $\hat{y}(t)$ is obtained via a linear readout with weights $\boldsymbol{z} \in \mathbb{R}^N$.

Learning setup The trainable parameters are collected as $\boldsymbol{\theta} = \{\boldsymbol{m}, \boldsymbol{W}, \boldsymbol{z}\}$ and updated across episodes to minimize the squared-error loss relative to a target $y^\star(t)$

$$\mathcal{L} = \int_0^T \left[\hat{y}(t) - y^\star(t)\right]^2 dt \tag{2}$$

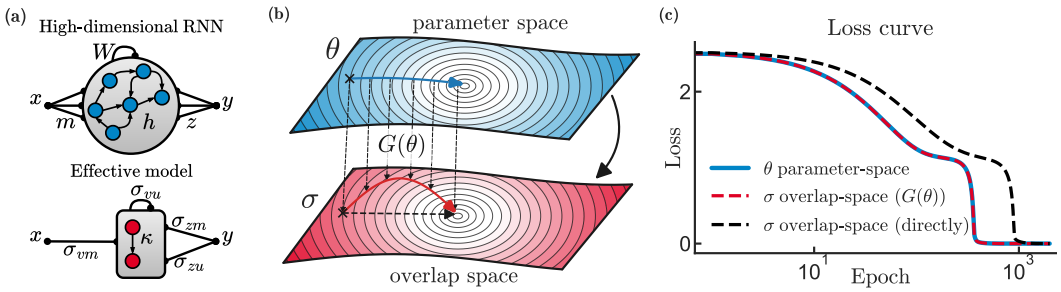

Figure 1: **(a)** High-dimensional RNN in parameter $\boldsymbol{\theta}$-space (top): input x drives activity $\boldsymbol{h}$ through input weights $\boldsymbol{m}$, recurrent connectivity $\boldsymbol{W}$, and readout weights $\boldsymbol{z}$ to produce output y. For a low-rank RNN, the same input–output function is captured by an effective model described by a small set of scalar overlaps $\boldsymbol{\sigma}$ (bottom). **(b)** Schematic illustration of a learning trajectory in the loss landscape over the parameter $\boldsymbol{\theta}$ (top, blue arrow) and its projection onto the loss landscape over the overlap $\boldsymbol{\sigma}$ via $\boldsymbol{G}(\boldsymbol{\theta})$ (bottom, red arrow), highlighting how this mapping can alter the perceived trajectory. Crucially, the projected dynamics can differ from direct optimization in overlap space (black dashed arrow), reflecting structural constraints of the parameterization. **(c)** Concrete example of (b), showing the loss dynamics of training a low-rank linear RNN on a filter task. Optimization in parameter space (blue) and the corresponding overlap dynamics induced by $\boldsymbol{G}$ (red dashed) closely agree, whereas direct optimization in overlap space (black dashed) produces different dynamics.

where T denotes the total episode duration. We analyze learning in the gradient-flow limit $\eta \to 0$, where parameter updates follow

$$\dot{\boldsymbol{\theta}} = -\nabla_{\boldsymbol{\theta}} \mathcal{L} \tag{3}$$

While this formulation provides an exact description of learning in the full parameter space, the resulting dynamics are high-dimensional and difficult to interpret. The key observation we exploit is that, in many cases of interest, the network output—and therefore the loss—depend on $\boldsymbol{\theta}$ only through a reduced set of variables (i.e., redundancy). Thus, although learning occurs in a high-dimensional space, behavior evolves along far fewer effective degrees of freedom. To make this point concrete, we next specialize to low-rank RNNs and derive a reduced description of learning.

3 Low-rank linear RNN

To build intuition, we begin with a simple tractable setting: a rank-1 RNN with linear activation $\phi(\cdot) = \mathrm{id}$. In this case, the dynamics of Eq. (1) simplify to

$$\dot{\boldsymbol{h}}(t) = -\boldsymbol{h}(t) + \frac{1}{N}\boldsymbol{u}\boldsymbol{v}^\top \boldsymbol{h}(t) + \boldsymbol{m}x(t) \tag{4}$$

where $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^N$ define the rank-1 recurrent connectivity. The trainable parameters are four vectors in $\mathbb{R}^N$, $\boldsymbol{\theta} = \{\boldsymbol{m}, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{z}\}$, corresponding to the input, left and right recurrent, and readout vectors. Assuming a zero initial condition $\boldsymbol{h}(0) = \boldsymbol{0}$, known results on low-rank RNNs [16] imply that the hidden-state dynamics are confined to the two-dimensional subspace $\mathrm{span}\{\boldsymbol{m}, \boldsymbol{u}\}$, and can therefore be written as

$$\boldsymbol{h}(t) = \kappa_m(t)\,\boldsymbol{m} + \kappa_u(t)\,\boldsymbol{u}, \qquad \boldsymbol{\kappa}(t) = \begin{bmatrix}\kappa_m(t)\\ \kappa_u(t)\end{bmatrix} \in \mathbb{R}^2 \tag{5}$$

where $\kappa_m(t)$ and $\kappa_u(t)$ denote the coordinates of the activity along the input and recurrent modes. Substituting this ansatz into Eq. (4) yields a 2D effective RNN (Fig.1a, bottom) with activity dynamics and output

$$\dot{\boldsymbol{\kappa}}(t) = -\boldsymbol{\kappa}(t) + \begin{bmatrix}0 & 0\\ \frac{1}{N}\boldsymbol{v}^\top\boldsymbol{m} & \frac{1}{N}\boldsymbol{v}^\top\boldsymbol{u}\end{bmatrix}\boldsymbol{\kappa}(t) + \begin{bmatrix}1\\0\end{bmatrix}x(t), \qquad \hat{y}(t) = \frac{1}{N}\begin{bmatrix}\boldsymbol{z}^\top\boldsymbol{m} & \boldsymbol{z}^\top\boldsymbol{u}\end{bmatrix}\boldsymbol{\kappa}(t) \tag{6}$$

Thus, the N-dimensional RNN reduces exactly to a 2D system whose input–output behavior is fully determined by four scalar overlaps

$$\sigma_{zm} = \tfrac{1}{N}\boldsymbol{z}^\top\boldsymbol{m}, \quad \sigma_{zu} = \tfrac{1}{N}\boldsymbol{z}^\top\boldsymbol{u}, \quad \sigma_{vm} = \tfrac{1}{N}\boldsymbol{v}^\top\boldsymbol{m}, \quad \sigma_{vu} = \tfrac{1}{N}\boldsymbol{v}^\top\boldsymbol{u} \tag{7}$$

We collect these quantities into the vector $\boldsymbol{\sigma} = (\sigma_{zm}, \sigma_{zu}, \sigma_{vm}, \sigma_{vu})$, and refer to them as the *loss-visible* overlaps, since they fully determine the within-episode dynamics and thus the loss. Crucially, although the loss depends only on $\boldsymbol{\sigma}$, optimization is performed in the high-dimensional parameter space $\boldsymbol{\theta}$. As a result, the trajectory of the overlaps induced by learning can differ from the one obtained by directly minimizing $\boldsymbol{\sigma}$ (Fig. 1b). To connect parameter-space learning with the induced dynamics in overlap space, we consider the Jacobian of $\boldsymbol{\sigma}$ with respect to $\boldsymbol{\theta}$

$$\boldsymbol{D}(\boldsymbol{\theta}) = \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\theta}} \in \mathbb{R}^{4\times 4N} \tag{8}$$

Because the loss depends on the parameters only through the overlaps, the chain rule gives $\nabla_{\boldsymbol{\theta}}\mathcal{L} = \boldsymbol{D}(\boldsymbol{\theta})^\top \nabla_{\boldsymbol{\sigma}}\mathcal{L}$, where $\nabla_{\boldsymbol{\sigma}}\mathcal{L} \in \mathbb{R}^4$. Under gradient flow in parameter space, the overlaps evolve as

$$\dot{\boldsymbol{\sigma}} = \boldsymbol{D}(\boldsymbol{\theta})\,\dot{\boldsymbol{\theta}} = -\boldsymbol{D}(\boldsymbol{\theta})\boldsymbol{D}(\boldsymbol{\theta})^\top \nabla_{\boldsymbol{\sigma}}\mathcal{L} \tag{9}$$

where $\boldsymbol{G}(\boldsymbol{\theta}) = \boldsymbol{D}(\boldsymbol{\theta})\boldsymbol{D}(\boldsymbol{\theta})^\top$ is a symmetric, positive semi-definite Gram matrix that defines the effective learning metric on overlap space. Thus, $\boldsymbol{G}(\boldsymbol{\theta})$ acts as a preconditioner, reshaping $\nabla_{\boldsymbol{\sigma}}\mathcal{L}$ according to the geometry inherited from parameter space. Interestingly, for a rank-1 RNN, the matrix $\boldsymbol{G}(\boldsymbol{\theta})$ can be computed in closed form (App. A.3) and is given by

$$\boldsymbol{G}(\boldsymbol{\theta}) = \frac{1}{N}\begin{bmatrix} \|\boldsymbol{m}\|^2 + \|\boldsymbol{z}\|^2 & \sigma_{mu} & \sigma_{zv} & 0 \\ \sigma_{mu} & \|\boldsymbol{u}\|^2 + \|\boldsymbol{z}\|^2 & 0 & \sigma_{zv} \\ \sigma_{zv} & 0 & \|\boldsymbol{m}\|^2 + \|\boldsymbol{v}\|^2 & \sigma_{mu} \\ 0 & \sigma_{zv} & \sigma_{mu} & \|\boldsymbol{u}\|^2 + \|\boldsymbol{v}\|^2 \end{bmatrix} \tag{10}$$

A direct inspection of $\boldsymbol{G}(\boldsymbol{\theta})$ shows six additional quantities beyond the loss-visible overlaps $\boldsymbol{\sigma}$. These include other overlaps $(\sigma_{mu}\,, \sigma_{zv})$ as well as all the squared norms of the parameter vectors, none of which contribute to the loss. We group them into

$$\tilde{\boldsymbol{\sigma}} = (\sigma_{mu}, \sigma_{zv}, \|\boldsymbol{m}\|^2, \|\boldsymbol{u}\|^2, \|\boldsymbol{v}\|^2, \|\boldsymbol{z}\|^2) \tag{11}$$

and refer to them as *loss-invisible* overlaps. Together, $(\boldsymbol{\sigma}, \tilde{\boldsymbol{\sigma}})$ provide a closed, 10-dimensional description of the learning dynamics in overlap space. In other words, initializations with the same ten overlaps yield identical input–output functions and learning trajectories, despite potentially differing along the high-dimensional parameter space. Note that the loss-invisible overlaps evolve analogously as the loss-visible ones, with their dynamics (i.e., $\dot{\tilde{\boldsymbol{\sigma}}}$) derived explicitly in App. A.2.

To verify that this compact low-dimensional description faithfully captures high-dimensional learning, we train a rank-1 linear RNN on a simple filter task. In this task, the network is trained to emulate the output of a first-order exponential filter driven by white noise input [25] (see App. A.4 for full details). The loss obtained from numerical simulation of gradient descent in the full parameter space matches exactly the prediction obtained by integrating our 10D ODE system in overlap space (Fig. 1c; and also see Fig. 5). In contrast, directly optimizing $\mathcal{L}(\boldsymbol{\sigma})$ produces qualitatively different loss dynamics (App. A.5 for a detailed comparison between optimization in both spaces). A complete derivation of this section is provided in App. A.

Before proceeding, we highlight an important point. The matrix $\boldsymbol{G}$ is a sub-block of a larger matrix that arises when considering the full set of ten quadratic overlaps among the four vectors $\boldsymbol{\theta} = \{\boldsymbol{m}, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{z}\}$. Differentiating this set with respect to $\boldsymbol{\theta}$ yields an augmented Jacobian $\bar{\boldsymbol{D}}(\boldsymbol{\theta}) \in \mathbb{R}^{10\times 4N}$ and a corresponding 10×10 Gram matrix $\bar{\boldsymbol{G}}(\boldsymbol{\theta})$ that jointly governs the evolution of both visible and invisible overlaps (see App. C and also App. D for a more general and compact derivation).

4 Implications of visible and invisible overlaps

We now examine two consequences of decomposing connectivity into loss-visible and loss-invisible overlaps in the context of learning.

4.1 Learning reveals hidden degeneracy

A direct consequence of this framework is the separation between two sets of overlaps. The first, loss-visible overlaps, determine the within-episode dynamics, output, and loss. The second, loss-invisible overlaps, leave the input–output function unchanged but shape the effective learning metric

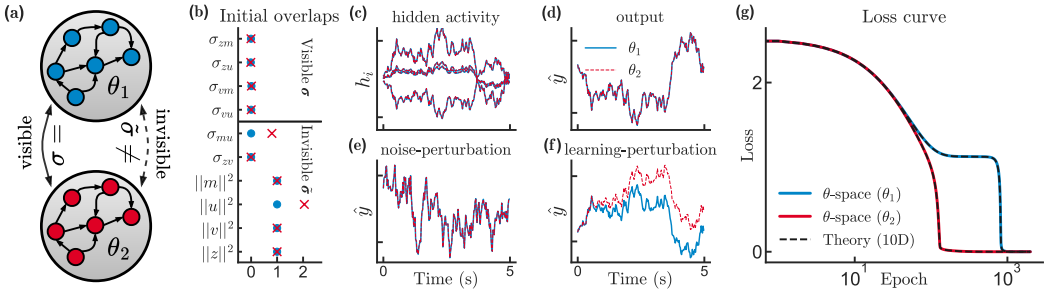

Figure 2: **(a)** Two RNNs parametrized with $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ share identical loss-visible $\boldsymbol{\sigma}$ ($=$) but differ in loss-invisible $\tilde{\boldsymbol{\sigma}}$ ($\neq$) overlaps. **(b)** Initial values of all overlaps (blue: $\boldsymbol{\theta}_1$, red: $\boldsymbol{\theta}_2$) show identical visible components (top) but differences in the invisible components (bottom), including σ_{mu} and $||u||^2$. **(c–d)** Because the input–output function depends only on the visible set, their hidden activity (c) and outputs (d) are indistinguishable. **(e)** Input noise perturbations likewise fail to differentiate the networks. **(f)** Once learning is turned on, differences in connectivity are revealed in the recorded outputs. **(g)** These invisible differences also emerge in the learning trajectories: $\boldsymbol{\theta}_1$ exhibits a transient plateau (blue) absent in $\boldsymbol{\theta}_2$ (red). The theory accurately captures both learning dynamics.

$\boldsymbol{G}(\boldsymbol{\theta})$. As a result, two networks can implement the same function yet differ in their underlying connectivity structures. Although such degeneracies are well documented in modern machine learning theory [15, 29], our framework provides a direct characterization of these hidden degrees of freedom and shows how learning can reveal them. In this sense, learning acts as a *perturbation* that reveals otherwise hidden degeneracies in the connectivity.

To illustrate this effect, consider two rank-1 linear RNNs parameterized by $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ (Fig. 2a), constructed to share identical loss-visible overlaps while differing in their loss-invisible overlaps (Fig. 2b). Because the visible overlaps coincide, the two networks exhibit identical hidden dynamics and outputs (Figs. 2c,d), and remain indistinguishable even under input noise-perturbations (Fig. 2e). However, once learning is initiated (training on the filter task), the difference in loss-invisible overlaps leads to different parameter updates, causing the two networks to produce distinct outputs in response to the same input (Fig. 2f). This divergence is also reflected in the loss dynamics (Fig. 2g), where network-1 exhibits a pronounced plateau that is absent in network-2. Importantly, both networks's learning dynamics are fully captured by our reduced 10D theory (dashed black).

4.2 Memory and its absence in invisible overlaps

Turning our attention to the loss-invisible overlaps. Although they do not affect the network's function, we showed above that they influence *future* learning. This raises the question of whether they can also retain information about the history of *past* learning, that is, serve as memory variables. To investigate this, we employ an A–B–A training protocol [30], in which the RNN is trained sequentially on task A, then task B, and finally retrained on task A. Because the loss constrains only the loss-visible overlaps, each task admits a continuous manifold of equivalent solutions parameterized by the loss-invisible directions (Fig. 3a). Thus, upon returning to task A, learning could either recover the original solution (blue) or find a different solution on the same manifold (red), revealing history dependence.

To determine which scenario is realized, we train a rank-1 linear RNN on a sequential filter task with two interleaved decay rates (Fig. 3b). Surprisingly, under vanilla gradient descent, we observe complete recovery: when retraining to task A, not only the loss-visible overlaps (expected), but also the loss-invisible overlaps (unexpectedly) return exactly to their original values (Fig. 3b, epoch 750 and Fig. 3c, top). This result indicates that the loss-invisible overlaps are not shaped by the training history, but are instead constrained by the task objective and the initialization. Indeed, our low-rank linear RNN falls within a class of matrix-factorized models [31], where the loss depends on the parameters only through a bilinear form involving two disjoint parameter matrices

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}\left(\frac{1}{N}\begin{bmatrix}\boldsymbol{z}^\top \\ \boldsymbol{v}^\top\end{bmatrix}\begin{bmatrix}\boldsymbol{m} & \boldsymbol{u}\end{bmatrix}\right) = \mathcal{L}\left(\begin{pmatrix}\sigma_{zm} & \sigma_{zu} \\ \sigma_{vm} & \sigma_{vu}\end{pmatrix}\right) \tag{12}$$

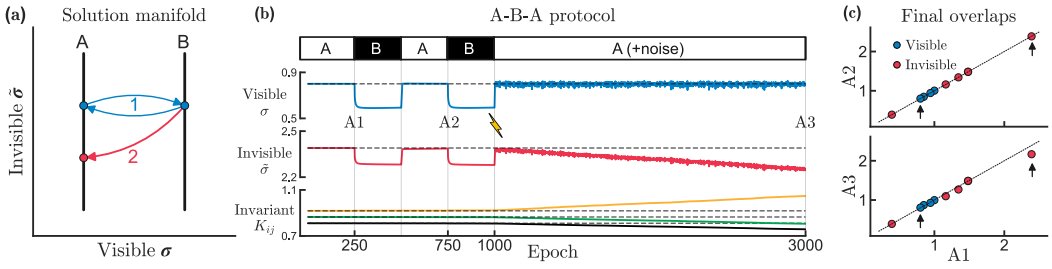

Figure 3: **(a)** Illustration of the solution manifold in overlap space for tasks A and B. For each task, the loss fixes a subset (or all) of the loss-visible overlaps $\boldsymbol{\sigma}$ (x-axis), leaving a continuous manifold (black lines) of equivalent solutions parameterized by *all* the loss-invisible overlaps $\tilde{\boldsymbol{\sigma}}$ (y-axis). Under an A–B–A training protocol, retraining on task A can either (1) recover the original solution (blue) or (2) converge to a different, history-dependent solution on the manifold (red). The panel shows a two-dimensional schematic; in general, the visible and invisible sets contain multiple overlaps. **(b)** Learning trajectories during the A–B–A protocol. Top: representative loss-visible overlap (σ_{vu}, blue). Middle: representative loss-invisible overlap ($\|\boldsymbol{u}\|^2$, red). Bottom: three sampled entries of the conserved matrix $\boldsymbol{K}$. Under vanilla gradient descent, retraining on task A restores both visible and invisible overlaps (compare A2, epoch 750 with A1, epoch 250), while $\boldsymbol{K}$ remains constant. Introducing label noise at epoch 1000 breaks the conservation of $\boldsymbol{K}$, inducing a directed drift in the invisible overlap (middle red) while leaving the visible overlap (top blue) unchanged. **(c)** Comparison of overlaps across task A solutions (A1, A2 and A3). Top: overlaps at A1 (epoch 250; x-axis) versus A2 (epoch 750; y-axis) lie on the identity line, demonstrating exact recovery. Bottom: with label noise A3 (epoch 3000; y-axis), invisible overlaps (red) deviate from the identity, whereas visible overlaps (blue) remain mostly unchanged. Arrows mark the overlaps depicted in (b).

For such models, gradient flow admits exact invariants of the learning dynamics (see App. A.6 for full derivation). In particular, the matrix

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{z} & \boldsymbol{v} \end{bmatrix} \begin{bmatrix} \boldsymbol{z}^\top \\ \boldsymbol{v}^\top \end{bmatrix} - \begin{bmatrix} \boldsymbol{m} & \boldsymbol{u} \end{bmatrix} \begin{bmatrix} \boldsymbol{m}^\top \\ \boldsymbol{u}^\top \end{bmatrix} = \boldsymbol{z}\boldsymbol{z}^\top + \boldsymbol{v}\boldsymbol{v}^\top - \boldsymbol{m}\boldsymbol{m}^\top - \boldsymbol{u}\boldsymbol{u}^\top \in \mathbb{R}^{N\times N} \tag{13}$$

is conserved throughout learning [32] (Fig. 3b, bottom). Consequently, training trajectories are confined to invariant manifolds set by the initialization. Thus, if the visible overlaps return to their original values, so will the invisible ones. This analysis implies that encoding memory in the loss-invisible overlaps requires breaking this invariant, which can be achieved either by modifying the architecture (see nonlinear below) or by altering the learning rule. In the latter case, adding label noise (which deviates from pure gradient flow) breaks the conservation of $\boldsymbol{K}$ (Fig. 3b, bottom, from epoch 1000 onward), causing its entries to drift. Notably, this perturbation induces a directed drift within the loss-invisible subspace while leaving loss-visible overlaps unchanged (Figs. 3b,c). Such behavior is consistent with SGD dynamics, where noise drives solutions toward flatter or lower-norm regions of the solution manifold [33, 34]. In our setting, this corresponds to a reduction in loss-invisible quantities such as $\|\boldsymbol{u}\|^2$. In the Appendix, we present full trajectories for all ten overlaps (Fig. 7) and show that adaptive optimizers, such as Adam [35], similarly break this invariant (Fig. 8).

5 Low-rank nonlinear RNN

The results presented thus far apply to linear RNNs. We now extend the analysis to a nonlinear network, still within the rank-1 setting. Specifically, we consider a network with dynamics

$$\dot{\boldsymbol{h}}(t) = -\boldsymbol{h}(t) + \frac{1}{N}\boldsymbol{u}\boldsymbol{v}^\top \phi\big(\boldsymbol{h}(t)\big) + \boldsymbol{m}x(t), \quad \phi(\boldsymbol{h}) = \operatorname{erf}\Big(\tfrac{\sqrt{\pi}}{2}\boldsymbol{h}\Big) \tag{14}$$

Here, the nonlinear activation is the error function chosen for analytical tractability [21], and the prefactor $\sqrt{\pi}/2$ ensures unit slope at the origin. Otherwise, the model is identical to the linear rank-1 RNN. To analyze the nonlinear case, we consider the limit $N \to \infty$ and assume that the components of the parameter vectors $\boldsymbol{\theta}$ are jointly Gaussian, following standard dynamical mean-field theory (DMFT) [16, 27]. As in the linear case, the hidden state remains confined to the two-dimensional

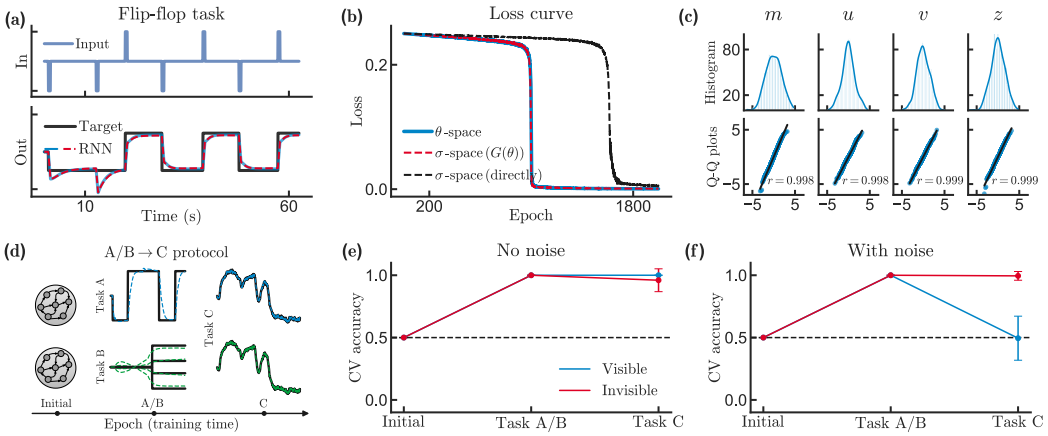

Figure 4: **(a)** Example input sequence of the flip-flop task (top). Target output (black), high-dimensional RNN prediction (blue), and effective RNN prediction (red dashed) show excellent agreement (bottom). **(b)** Training loss of RNN in parameter space $\boldsymbol{\theta}$ (blue) and 10D overlap dynamics using $\boldsymbol{G}(\boldsymbol{\theta})$ (red dashed), which closely match, while direct optimization in $\boldsymbol{\sigma}$ space (black dashed), leads to different dynamics. **(c)** Distributions of parameter vector components $\boldsymbol{\theta}$ at convergence (top), along with corresponding Q–Q plots (bottom), showing that the Gaussian structure is preserved (correlation coefficient $r \geq 0.998$). **(d)** Schematic of the history-dependent training protocol (A/B $\rightarrow$ C), where two identically initialized twin networks are first trained on either task A (flip-flop) or task B (stimulus integration), and then both on the same task C (emulating a teacher network). Solid black lines indicate targets, and dashed colored lines are the network outputs. **(e–f)** Cross-validation accuracy for decoding training history using loss-visible (blue) and loss-invisible (red) overlaps (error bars ± 1 s.d). (e) Without noise, decoding is possible from either set due to infinitesimal differences. (f) With small noise added to the overlaps ($\mathcal{N}(0, 0.1)$), only loss-invisible overlaps remain informative, while loss-visible overlaps drop to chance level.

subspace spanned by $\boldsymbol{m}$ and $\boldsymbol{u}$, and can therefore be written as in Eq. (5). The key difference is that the recurrent input now depends nonlinearly on the state. Using Stein's Lemma, one obtains

$$\frac{1}{N}\boldsymbol{v}^\top \phi(\boldsymbol{h}(t)) = \big(\sigma_{vm}\kappa_m(t) + \sigma_{vu}\kappa_u(t)\big)\,\langle \phi' \rangle \tag{15}$$

For the erf nonlinearity, the average gain $\langle \phi' \rangle$ is analytically tractable and given by

$$\begin{aligned} \langle \phi' \rangle &= \mathbb{E}_{g\sim\mathcal{N}(0,\Delta(t))}[\phi'(g)] = \big(1 + \tfrac{\pi}{2}\Delta(t)\big)^{-1/2} \\ \Delta(t) &= \|\boldsymbol{m}\|^2\kappa_m(t)^2 + \|\boldsymbol{u}\|^2\kappa_u(t)^2 + 2\,\sigma_{mu}\,\kappa_m(t)\kappa_u(t) \end{aligned} \tag{16}$$

Thus, under the Gaussian assumption, the input–output behavior of the nonlinear RNN is described by a finite set of macroscopic overlaps. Crucially, unlike in the linear case, the overlap σ_{mu} as well as $\|\boldsymbol{m}\|^2$ and $\|\boldsymbol{u}\|^2$ now enter the dynamics through $\Delta(t)$ in Eq. (16), and therefore become *loss-visible*, altering the previous visible–invisible separation. Accordingly, the loss no longer admits a disjoint bilinear dependence on the parameters as in Eq. (12), with important consequences described next. A complete mean-field derivation and full learning equation are provided in App. B.

5.1 Flip-flop task

To validate our nonlinear derivation, we consider the 1-bit flip-flop task (see App. B.4 for full details), a widely studied task in theoretical neuroscience [36]. The task requires the network to maintain a stable internal state and update it only in response to brief, signed input pulses (Fig. 4a). As this requires bistability, which is absent in linear systems, it provides a natural setting to assess our nonlinear network. Training a high-dimensional nonlinear RNN on this task and comparing it with the corresponding 10D ODE theory, we observe close agreement in the loss curves (Fig. 4b). Furthermore, as in the linear case, directly optimizing in overlap space yields qualitatively different learning dynamics, underscoring the role of the structural preconditioning $\boldsymbol{G}(\boldsymbol{\theta})$. However, unlike

the linear case, the validity of the theory relies on the components of the weight vectors $\boldsymbol{\theta}$ remaining approximately Gaussian. While gradient descent dynamics alone do not guarantee this condition, we numerically find that it holds throughout training for sufficiently small learning rates (Fig. 4c). In the Appendix, we test the limits of the Gaussian assumption, showing that training with larger learning rates or using alternative optimizers (e.g., Adam) leads to deviations from Gaussianity and discrepancies between theory and simulation (see App. B.4.1 and Fig. 9).

5.2 Memory is encoded in the invisible overlaps

Finally, we revisit whether the invisible overlaps can serve as memory variables encoding past training history. In linear networks trained with vanishingly small learning rate, the invisible overlaps are constrained by an invariant $\boldsymbol{K}$ and thus return to their original values. In contrast, in nonlinear networks, the altered visible–invisible separation breaks the invariant, opening the possibility for memory storage in the invisible set. To demonstrate this, we devise a hypothetical training protocol where identically initialized networks (twin networks) undergo history-dependent training. Each network is first trained on a different task (A/B), then both on the same task (C), yielding identical outputs. We then ask whether training history can be decoded from the overlaps. We conjecture that, since the loss-visible overlaps determine the output, they must converge to the same values (up to task degeneracies) and are thus weakly informative. In contrast, loss-invisible overlaps do not affect the output and can retain distinct values, enabling them to encode the training history.

To test this, we train 20 networks (10 twin pairs) using the 10D overlap ODEs with the preconditioned metric $\boldsymbol{G}$, ensuring the Gaussian assumption is satisfied by construction while also affording a computational speedup. Each network is first trained on either Task A (flip-flop) or Task B (stimulus integration), and then on a common Task C, where it emulates the response of a teacher network (Fig. 4d; see also App. B.5 for full task details). We then train a classifier (logistic regression; similar results are obtained with other classifiers) to predict the training history (A or B) using either loss-visible or loss-invisible overlaps. Performance is evaluated via cross-validation accuracy over 50 random train–test splits and at three training checkpoints (Initial, Task A/B, and Task C). Our analysis reveals that, while both loss-visible and loss-invisible overlaps appear to be informative of the training history in the noiseless setting (Fig. 4e), this reflects infinitesimal differences arising from imperfect convergence (see Fig. 11 for full overlap trajectories). However, consistent with our prediction, adding a small amount of noise to the overlaps (to mimic realistic conditions) reveals that only the loss-invisible overlaps enable robust decoding of the training history, while the accuracy of the loss-visible set drops to chance level (Fig. 4f). This suggests that loss-invisible overlaps can serve as memory variables.

6 Related work

Low-rank RNNs RNNs are widely used in theoretical neuroscience and machine learning [28, 37, 38, 39, 40]. A particularly tractable class, especially in theoretical neuroscience, consists of RNNs with low-rank connectivity, which simplifies the dynamics and enables analysis via a small set of macroscopic overlap variables [16]. Historically, research on low-rank RNNs has followed two main directions. The first constructs networks by hand, designing connectivity to implement specific computations and analyzing the resulting dynamics [16, 17, 18, 19, 21, 41]. The second studies how such structure emerges through learning, either by training low-rank networks directly or by showing that unconstrained networks develop low-rank solutions [22, 23, 24, 42].

Learning dynamics of RNNs More recently, a third line of work has begun to analyze RNN learning dynamics analytically [22, 25, 26, 43]. Two particularly relevant studies are [25, 26], which analyze unconstrained *linear* RNNs in both the lazy and rich regimes. In [25], learning equations are derived for simplified tasks, building on earlier results [22]. In [26], ideas from feedforward networks [32, 44] are extended to RNNs, strongly relying on task decomposition into singular modes. These approaches obtain tractability through assumptions such as timescale separation, frozen parameters, or balanced, aligned, or tied initializations. Our framework instead leverages the low-rank architecture rather than a task-specific decomposition, allowing all parameters to evolve simultaneously under gradient descent. Within this setting, we further show that initialization structure (including the relative magnitudes across layers, beyond their overall scale [45]), particularly within the loss-invisible set, can substantially alter learning trajectories. Finally, we extend the

analysis to *nonlinear* RNNs, providing, to our knowledge, the first analytical treatment of their full deterministic gradient-descent trajectories. We note that two recent works, which appeared during the preparation and review of our manuscript [46, 47], provide complementary insights into learning in nonlinear RNNs through Bayesian mean-field descriptions. These works characterize the equilibrium distribution of trained weights under Langevin dynamics rather than the full learning trajectory.

Parameter and function space duality Our work analyzes learning dynamics in a low-dimensional overlap space rather than the high-dimensional parameter space, focusing on the function implemented by the network. This perspective is related to the Neural Tangent Kernel (NTK) framework [48, 49], where learning in infinitely wide networks (mostly feedforward) is described at the level of functions rather than parameters. However, while NTK analyses typically focus on the output layer, our approach tracks additional quantities beyond it. Our framework is also related to natural-gradient methods [50, 51]. In particular, the matrix $\boldsymbol{G}$ maps parameter-space gradient descent to induced dynamics in overlap space, endowing these coordinates with a non-Euclidean geometry. Conceptually, this plays a similar role to a metric or preconditioner in natural-gradient methods, but here it arises directly from the low-rank parameterization instead of defining a distance metric.

Invariants, symmetries, and drift Finally, our analysis connects to work on invariants and symmetries in neural network learning dynamics [31, 52, 53]. We interpret loss-invisible components of connectivity as potential memory variables. In linear networks under gradient flow, conserved quantities constrain these directions and prevent them from storing training history. Breaking these symmetries—via modified learning rules or nonlinearities—removes this constraint, enabling loss-invisible directions to encode past training. We further show that label noise induces a slow drift along these directions, consistent with results that noisy optimization (e.g., SGD) biases solutions toward balanced, minimal-norm, or flatter regions of the solution manifold [33, 34, 54].

7 Discussion

Learning is a fundamental property of both biological and artificial neural networks, yet linking structural changes during learning (e.g., synaptic plasticity or weight updates) to changes in network function remains challenging. Here, we leverage the low-rank RNN framework to derive a low-dimensional description of learning under gradient descent. Our central insight is that learning can be captured through a small set of overlaps that separate connectivity into *loss-visible* directions, which determine network function, and *loss-invisible* directions, which shape how learning unfolds. We demonstrate this perspective in both linear and nonlinear RNNs.

Biological implications This perspective yields several testable predictions for biological learning experiments. First, we introduce the concept of *perturbation-by-learning*, whereby observing how a system learns can reveal structural differences that remain hidden in static recordings of behavior. In this sense, learning trajectories act as probes of circuit structure. Second, we interpret loss-invisible directions as candidate memory variables encoding a circuit's training history (and potentially also future learning capacity). This suggests that uncovering learning history requires focusing not on synapses that determine current behavior, but on those that are functionally silent yet central to learning. More broadly, this perspective parallels findings that residual neural activity, though behaviorally silent, may carry valuable information [55, 56], and extends this principle from neural activity to learning dynamics. Finally, inferring the matrix $\boldsymbol{G}$ could guide the design of efficient shaping protocols tailored to the microscopic parameters whose plasticity mediates learning.

Limitations We note several limitations of our work. First, for analytical tractability, we focus on rank-1 connectivity. While extensions to higher-rank networks are possible (App. D), the number of overlaps grows quadratically with rank, leading to increasingly complex dynamics. Second, gradients with respect to the overlaps are not always analytically tractable; although closed-form expressions exist for some tasks (e.g., the filter task; App. A.4.3), they may be unavailable in more complex ones. Nevertheless, simulating the resulting low-dimensional overlap dynamics is far more efficient than the full N-dimensional ones. Third, our nonlinear analysis assumes that the parameter-vector components remain approximately Gaussian during training, an assumption not strictly preserved (App. B.4.1). Finally, we consider purely low-rank connectivity, enforced by training the low-rank factors directly

rather than the full recurrent matrix $\boldsymbol{W}$. Many RNN studies instead train $\boldsymbol{W}$ directly or add a full-rank random bulk to the low-rank connectivity, whose incorporation remains an important future direction.

In summary, our work extends the low-rank RNN framework to incorporate learning dynamics within the same low-dimensional description. This perspective provides a tractable link between changes in connectivity and the evolution of network function, offering a principled framework for studying degeneracy, memory, and drift in recurrent networks, with implications for both neuroscience and machine learning theory.

Acknowledgments

We thank an anonymous reviewer for the suggestion of the compact derivation in Appendix D. This work was supported by the Israel Science Foundation (grant No. 1442/21 to OB) and Human Frontiers Science Program (HFSP) research grant (RGP0017/2021 to OB). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Code Availability

All code was implemented in Python using PyTorch [57] and is available on GitHub: `https://github.com/yoavger/learning_reveals_invisible_structure_lr_rnns`

References

[1] Andrew B Barron, Eileen A Hebets, Thomas A Cleland, Courtney L Fitzpatrick, Mark E Hauber, and Jeffrey R Stevens. Embracing multiple definitions of learning. *Trends in neurosciences*, 38(7):405–407, 2015.

[2] Jay A Hennig, Emily R Oby, Darby M Losey, Aaron P Batista, Byron M Yu, and Steven M Chase. How learning unfolds in the brain: toward an optimization view. *Neuron*, 109(23):3720–3735, 2021.

[3] Andrew Saxe, Stephanie Nelli, and Christopher Summerfield. If deep learning is the answer, what is the question? *Nature Reviews Neuroscience*, 22(1):55–67, 2021.

[4] Richard F Thompson. The neurobiology of learning and memory. *Science*, 233(4767):941–947, 1986.

[5] Jeffrey C Magee and Christine Grienberger. Synaptic plasticity forms and functions. *Annual review of neuroscience*, 43(1):95–117, 2020.

[6] Patrick T Sadtler, Kristin M Quick, Matthew D Golub, Steven M Chase, Stephen I Ryu, Elizabeth C Tyler-Kabara, Byron M Yu, and Aaron P Batista. Neural constraints on learning. *Nature*, 512(7515):423–426, 2014.

[7] Yann Humeau and Daniel Choquet. The next generation of approaches to investigate the link between synaptic plasticity and learning. *Nature neuroscience*, 22(10):1536–1543, 2019.

[8] Peiran Gao and Surya Ganguli. On simplicity and complexity in the brave new world of large-scale neuroscience. *Current opinion in neurobiology*, 32:148–155, 2015.

[9] Abhranil Das and Ila R Fiete. Systematic errors in connectivity inferred from activity in strongly recurrent networks. *Nature Neuroscience*, 23(10):1286–1296, 2020.

[10] Gerald M Edelman and Joseph A Gally. Degeneracy and complexity in biological systems. *Proceedings of the national academy of sciences*, 98(24):13763–13768, 2001.

[11] Astrid A Prinz, Dirk Bucher, and Eve Marder. Similar network activity from disparate circuit parameters. *Nature neuroscience*, 7(12):1345–1352, 2004.

[12] Larissa Albantakis, Christophe Bernard, Naama Brenner, Eve Marder, and Rishikesh Narayanan. The brain's best kept secret is its degenerate structure. *Journal of Neuroscience*, 44(40), 2024.

[13] Francesca Albertini and Eduardo D Sontag. For neural networks, function determines form. *Neural networks*, 6(7):975–990, 1993.

[14] Geoffrey Roeder, Luke Metz, and Durk Kingma. On linear identifiability of learned representations. In *International Conference on Machine Learning*, pages 9030–9039. PMLR, 2021.

[15] Lukas Braun, Erin Grant, and Andrew M Saxe. Not all solutions are created equal: An analytical dissociation of functional and representational similarity in deep linear neural networks. In *Forty-second International Conference on Machine Learning*, 2025.

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

[17] 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.

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

[19] 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.

[20] Chris Eliasmith and Charles H Anderson. *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT press, 2003.

[21] Owen Marschall, David G Clark, and Ashok Litwin-Kumar. A theory of multi-task computation and task selection. *bioRxiv*, pages 2025–12, 2025.

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

[23] Alexis Dubreuil, Adrian Valente, Manuel Beiran, Francesca Mastrogiuseppe, and Srdjan Ostojic. The role of population structure in computations through neural dynamics. *Nature neuroscience*, 25(6):783–794, 2022.

[24] Adrian Valente, Jonathan W Pillow, and Srdjan Ostojic. Extracting computational mechanisms from neural data using low-rank rnns. *Advances in Neural Information Processing Systems*, 35:24072–24086, 2022.

[25] Blake Bordelon, Jordan Cotler, Cengiz Pehlevan, and Jacob A Zavatone-Veth. Dynamically learning to integrate in recurrent neural networks. *arXiv preprint arXiv:2503.18754*, 2025.

[26] Alexandra Maria Proca, Clémentine Carla Juliette Dominé, Murray Shanahan, and Pedro AM Mediano. Learning dynamics in linear recurrent neural networks. In *Forty-second International Conference on Machine Learning*, 2025.

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

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

[29] Ann Huang, Satpreet Harcharan Singh, Flavio Martinelli, and Kanaka Rajan. Measuring and controlling solution degeneracy across task-trained recurrent neural networks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[30] Basile Confavreux, Will Dorrell, Nishil Patel, and Andrew M Saxe. Memory by accident: a theory of learning as a byproduct of network stabilization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[31] Simon S Du, Wei Hu, and Jason D Lee. Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced. *Advances in neural information processing systems*, 31, 2018.

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

[33] Guy Blanc, Neha Gupta, Gregory Valiant, and Paul Valiant. Implicit regularization for deep neural networks driven by an ornstein-uhlenbeck like process. In *Conference on learning theory*, pages 483–513. PMLR, 2020.

[34] Aviv Ratzon, Dori Derdikman, and Omri Barak. Representational drift as a result of implicit regularization. *Elife*, 12:RP90069, 2024.

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

[36] 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.

[37] Niru Maheswaranathan, Alex Williams, Matthew Golub, Surya Ganguli, and David Sussillo. Reverse engineering recurrent networks for sentiment classification reveals line attractor dynamics. *Advances in neural information processing systems*, 32, 2019.

[38] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. Ieee, 2013.

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

[40] Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In *International Conference on Machine Learning*, pages 26670–26698. PMLR, 2023.

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

[42] Matthijs Pals, Jakob H Macke, and Omri Barak. Trained recurrent neural networks develop phase-locked limit cycles in a working memory task. *PLOS Computational Biology*, 20(2):e1011852, 2024.

[43] Yoav Ger and Omri Barak. Learning dynamics of rnns in closed-loop environments. *Advances in Neural Information Processing Systems*, 38:96353–96392, 2026.

[44] 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.

[45] Shahar Azulay, Edward Moroshko, Mor Shpigel Nacson, Blake E Woodworth, Nathan Srebro, Amir Globerson, and Daniel Soudry. On the implicit bias of initialization shape: Beyond infinitesimal mirror descent. In *International Conference on Machine Learning*, pages 468–477. PMLR, 2021.

[46] Jan P Bauer, Kirsten Fischer, Moritz Helias, and Agostina Palmigiano. A unified theory of feature learning in rnns and dnns. *arXiv preprint arXiv:2602.15593*, 2026.

[47] David G Clark, Blake Bordelon, Jacob A Zavatone-Veth, and Cengiz Pehlevan. Structure, disorder, and dynamics in task-trained recurrent neural circuits. *bioRxiv*, pages 2026–03, 2026.

[48] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[49] Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in neural information processing systems*, 32, 2019.

[50] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[51] Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*, 2013.

[52] Daniel Kunin, Javier Sagastuy-Brena, Surya Ganguli, Daniel LK Yamins, and Hidenori Tanaka. Neural mechanics: Symmetry and broken conservation laws in deep learning dynamics. *arXiv preprint arXiv:2012.04728*, 2020.

[53] Hidenori Tanaka and Daniel Kunin. Noether's learning dynamics: Role of symmetry breaking in neural networks. *Advances in Neural Information Processing Systems*, 34:25646–25660, 2021.

[54] Zhiyuan Li, Tianhao Wang, and Sanjeev Arora. What happens after sgd reaches zero loss?–a mathematical framework. *arXiv preprint arXiv:2110.06914*, 2021.

[55] Aniruddh R Galgali, Maneesh Sahani, and Valerio Mante. Residual dynamics resolves recurrent contributions to neural computation. *Nature Neuroscience*, 26(2):326–338, 2023.

[56] Ulises Pereira-Obilinovic, Kayvon Daie, Susu Chen, Karel Svoboda, and Ran Darshan. Neural dynamics outside task-coding dimensions drive decision trajectories through transient amplification. *bioRxiv*, pages 2025–11, 2025.

[57] A Paszke. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

Appendix

The appendix is organized as follows:

- **Section A** – Full derivation of the linear rank-1 RNN: reduced activity dynamics, overlap learning dynamics, filter task and training details, a detailed comparison of parameter-space and overlap-space learning, gradient-flow invariants, and experiments using alternative learning rules.
- **Section B** – Full derivation of the nonlinear rank-1 RNN: mean-field reduced activity dynamics, overlap learning dynamics, flip-flop task and training details, experiments testing the limits of the Gaussian assumptions, and full details of the history-dependent memory protocol (A/B $\rightarrow$ C).
- **Section C** – Derivation of the augmented 10×10 Gram matrix $\bar{\boldsymbol{G}}(\boldsymbol{\theta})$, and detailed comparison between linear and nonlinear mode
- **Section D** – Extension to a linear rank-2 RNN in compact matrix form, and including how overlaps scale with rank.

A Linear rank-1 RNN

We provide here a complete derivation of the linear rank-1 RNN, including both the within-episode dynamics and the across-episode learning dynamics.

A.1 Within-episode dynamics

With rank-1 connectivity $\boldsymbol{W} = \frac{1}{N}\boldsymbol{u}\boldsymbol{v}^\top$ and linear activation $\phi = \mathrm{id}$, the state dynamics becomes

$$\dot{\boldsymbol{h}}(t) = -\boldsymbol{h}(t) + \frac{1}{N}\boldsymbol{u}\boldsymbol{v}^\top \boldsymbol{h}(t) + \boldsymbol{m}\, x(t) \tag{A.1}$$

Assuming $\boldsymbol{h}(0) = \boldsymbol{0}$, the right-hand side of Eq. (A.1) always lies in $\mathrm{span}\{\boldsymbol{m}, \boldsymbol{u}\}$, since

$$\boldsymbol{u}\boldsymbol{v}^\top \boldsymbol{h}(t) \in \mathrm{span}\{\boldsymbol{u}\} \quad \text{and} \quad \boldsymbol{m}\, x(t) \in \mathrm{span}\{\boldsymbol{m}\}$$

Therefore $\boldsymbol{h}(t) \in \mathrm{span}\{\boldsymbol{m}, \boldsymbol{u}\}$ for all t, and we may write

$$\boldsymbol{h}(t) = \kappa_m(t)\, \boldsymbol{m} + \kappa_u(t)\, \boldsymbol{u}, \qquad \boldsymbol{\kappa}(t) = \begin{bmatrix} \kappa_m(t) \\ \kappa_u(t) \end{bmatrix} \in \mathbb{R}^2 \tag{A.2}$$

Projecting Eq. (A.1) onto this subspace yields the effective dynamics

$$\dot{\boldsymbol{\kappa}}(t) = -\boldsymbol{\kappa}(t) + \begin{bmatrix} 0 & 0 \\ \frac{1}{N}\boldsymbol{v}^\top \boldsymbol{m} & \frac{1}{N}\boldsymbol{v}^\top \boldsymbol{u} \end{bmatrix} \boldsymbol{\kappa}(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} x(t) \tag{A.3}$$

and readout

$$\hat{y}(t) = \frac{1}{N}\boldsymbol{z}^\top \boldsymbol{h}(t) = \frac{1}{N}\begin{bmatrix} \boldsymbol{z}^\top \boldsymbol{m} & \boldsymbol{z}^\top \boldsymbol{u} \end{bmatrix} \boldsymbol{\kappa}(t) \tag{A.4}$$

A.2 Across-episode learning dynamics

The episode loss depends on the parameters only through the *loss-visible* overlaps

$$\mathcal{L} = \mathcal{L}\big(\sigma_{zm}, \sigma_{zu}, \sigma_{vm}, \sigma_{vu}\big) \tag{A.5}$$

By the chain rule, the gradients with respect to the parameter vectors can be written as

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{m}} = \frac{\partial \mathcal{L}}{\partial \sigma_{zm}}\frac{\partial \sigma_{zm}}{\partial \boldsymbol{m}} + \frac{\partial \mathcal{L}}{\partial \sigma_{vm}}\frac{\partial \sigma_{vm}}{\partial \boldsymbol{m}} = \frac{1}{N}\left(\nabla_{zm}\boldsymbol{z} + \nabla_{vm}\boldsymbol{v}\right) \tag{A.6}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}} = \frac{\partial \mathcal{L}}{\partial \sigma_{zu}}\frac{\partial \sigma_{zu}}{\partial \boldsymbol{u}} + \frac{\partial \mathcal{L}}{\partial \sigma_{vu}}\frac{\partial \sigma_{vu}}{\partial \boldsymbol{u}} = \frac{1}{N}\left(\nabla_{zu}\boldsymbol{z} + \nabla_{vu}\boldsymbol{v}\right) \tag{A.7}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{v}} = \frac{\partial \mathcal{L}}{\partial \sigma_{vm}}\frac{\partial \sigma_{vm}}{\partial \boldsymbol{v}} + \frac{\partial \mathcal{L}}{\partial \sigma_{vu}}\frac{\partial \sigma_{vu}}{\partial \boldsymbol{v}} = \frac{1}{N}\left(\nabla_{vm}\boldsymbol{m} + \nabla_{vu}\boldsymbol{u}\right) \tag{A.8}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{z}} = \frac{\partial \mathcal{L}}{\partial \sigma_{zm}}\frac{\partial \sigma_{zm}}{\partial \boldsymbol{z}} + \frac{\partial \mathcal{L}}{\partial \sigma_{zu}}\frac{\partial \sigma_{zu}}{\partial \boldsymbol{z}} = \frac{1}{N}\left(\nabla_{zm}\boldsymbol{m} + \nabla_{zu}\boldsymbol{u}\right) \tag{A.9}$$

Under gradient flow, $\dot{\boldsymbol{\theta}} = -\nabla_{\boldsymbol{\theta}}\mathcal{L}$, the parameter dynamics become

$$\begin{aligned} \dot{\boldsymbol{m}} &= -N^{-1}(\nabla_{zm}\,\boldsymbol{z} + \nabla_{vm}\,\boldsymbol{v}) \qquad & \dot{\boldsymbol{u}} &= -N^{-1}(\nabla_{zu}\,\boldsymbol{z} + \nabla_{vu}\,\boldsymbol{v}) \\ \dot{\boldsymbol{v}} &= -N^{-1}(\nabla_{vm}\,\boldsymbol{m} + \nabla_{vu}\,\boldsymbol{u}) \qquad & \dot{\boldsymbol{z}} &= -N^{-1}(\nabla_{zm}\,\boldsymbol{m} + \nabla_{zu}\,\boldsymbol{u}) \end{aligned} \tag{A.10}$$

By the product rule, for any two vectors $\boldsymbol{v}$ and $\boldsymbol{u}$, the derivative of their inner product is

$$\frac{d}{d\tau}(\boldsymbol{v}^\top \boldsymbol{u}) = \dot{\boldsymbol{v}}^\top \boldsymbol{u} + \boldsymbol{v}^\top \dot{\boldsymbol{u}} \tag{A.11}$$

Applying this to the loss-visible overlaps together with Eq. (A.10) yields the induced dynamics

$$\begin{aligned} \dot{\sigma}_{zm} &= -(\|\boldsymbol{m}\|^2 + \|\boldsymbol{z}\|^2)\nabla_{zm} - \sigma_{mu}\nabla_{zu} - \sigma_{zv}\nabla_{vm} \\ \dot{\sigma}_{zu} &= -\sigma_{mu}\nabla_{zm} - (\|\boldsymbol{u}\|^2 + \|\boldsymbol{z}\|^2)\nabla_{zu} - \sigma_{zv}\nabla_{vu} \\ \dot{\sigma}_{vm} &= -(\|\boldsymbol{m}\|^2 + \|\boldsymbol{v}\|^2)\nabla_{vm} - \sigma_{mu}\nabla_{vu} - \sigma_{zv}\nabla_{zm} \\ \dot{\sigma}_{vu} &= -\sigma_{mu}\nabla_{vm} - (\|\boldsymbol{u}\|^2 + \|\boldsymbol{v}\|^2)\nabla_{vu} - \sigma_{zv}\nabla_{zu} \end{aligned} \tag{A.12}$$

which revealed six additional *loss-invisible* overlaps that are needed to close the learning dynamics

$$\sigma_{mu}, \quad \sigma_{zv}, \quad \|\boldsymbol{m}\|^2, \quad \|\boldsymbol{u}\|^2, \quad \|\boldsymbol{v}\|^2, \quad \|\boldsymbol{z}\|^2 \tag{A.13}$$

Similarly, their evolution under gradient flow is

$$\begin{aligned}
\dot{\sigma}_{mu} &= -\sigma_{zu}\nabla_{zm} - \sigma_{zm}\nabla_{zu} - \sigma_{vu}\nabla_{vm} - \sigma_{vm}\nabla_{vu} \\
\dot{\sigma}_{zv} &= -\sigma_{vm}\nabla_{zm} - \sigma_{vu}\nabla_{zu} - \sigma_{zm}\nabla_{vm} - \sigma_{zu}\nabla_{vu} \\
\dot{\|\boldsymbol{m}\|^2} &= -2(\sigma_{zm}\nabla_{zm} + \sigma_{vm}\nabla_{vm}) \\
\dot{\|\boldsymbol{u}\|^2} &= -2(\sigma_{zu}\nabla_{zu} + \sigma_{vu}\nabla_{vu}) \\
\dot{\|\boldsymbol{v}\|^2} &= -2(\sigma_{vm}\nabla_{vm} + \sigma_{vu}\nabla_{vu}) \\
\dot{\|\boldsymbol{z}\|^2} &= -2(\sigma_{zm}\nabla_{zm} + \sigma_{zu}\nabla_{zu})
\end{aligned} \tag{A.14}$$

These equations form a closed 10-dimensional ODE system in scalar variables, which fully describes the learning trajectory in overlap space.

A.3 Learning dynamics in matrix form

The learning dynamics can be written compactly in matrix form by defining

$$\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{m} \\ \boldsymbol{u} \\ \boldsymbol{v} \\ \boldsymbol{z} \end{bmatrix} \in \mathbb{R}^{4N}, \qquad \boldsymbol{\sigma} = \begin{bmatrix} \sigma_{zm} \\ \sigma_{zu} \\ \sigma_{vm} \\ \sigma_{vu} \end{bmatrix} \in \mathbb{R}^4, \qquad \nabla_{\boldsymbol{\sigma}}\mathcal{L} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{\sigma}} = \begin{bmatrix} \nabla_{zm} \\ \nabla_{zu} \\ \nabla_{vm} \\ \nabla_{vu} \end{bmatrix} \in \mathbb{R}^4 \tag{A.15}$$

Let $\boldsymbol{D}$ denote the Jacobian of the overlap map with respect to the parameters

$$\boldsymbol{D}(\boldsymbol{\theta}) = \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\theta}} = \frac{1}{N} \begin{bmatrix} \boldsymbol{z}^\top & 0 & 0 & \boldsymbol{m}^\top \\ 0 & \boldsymbol{z}^\top & 0 & \boldsymbol{u}^\top \\ \boldsymbol{v}^\top & 0 & \boldsymbol{m}^\top & 0 \\ 0 & \boldsymbol{v}^\top & \boldsymbol{u}^\top & 0 \end{bmatrix} \in \mathbb{R}^{4\times 4N} \tag{A.16}$$

Under gradient flow, $\dot{\boldsymbol{\theta}} = -\boldsymbol{D}(\boldsymbol{\theta})^\top \nabla_{\boldsymbol{\sigma}}\mathcal{L}$, and the induced overlap dynamics follow as

$$\dot{\boldsymbol{\sigma}} = \boldsymbol{D}(\boldsymbol{\theta})\,\dot{\boldsymbol{\theta}} = -\boldsymbol{D}(\boldsymbol{\theta})\boldsymbol{D}(\boldsymbol{\theta})^\top \nabla_{\boldsymbol{\sigma}}\mathcal{L} = -\boldsymbol{G}(\boldsymbol{\theta})\,\nabla_{\boldsymbol{\sigma}}\mathcal{L}, \qquad \boldsymbol{G}(\boldsymbol{\theta}) = \boldsymbol{D}(\boldsymbol{\theta})\boldsymbol{D}(\boldsymbol{\theta})^\top \tag{A.17}$$

The matrix $\boldsymbol{G}(\boldsymbol{\theta})$ is given explicitly by

$$\boldsymbol{G}(\boldsymbol{\theta}) = \frac{1}{N} \begin{bmatrix} \|\boldsymbol{m}\|^2 + \|\boldsymbol{z}\|^2 & \sigma_{mu} & \sigma_{zv} & 0 \\ \sigma_{mu} & \|\boldsymbol{u}\|^2 + \|\boldsymbol{z}\|^2 & 0 & \sigma_{zv} \\ \sigma_{zv} & 0 & \|\boldsymbol{m}\|^2 + \|\boldsymbol{v}\|^2 & \sigma_{mu} \\ 0 & \sigma_{zv} & \sigma_{mu} & \|\boldsymbol{u}\|^2 + \|\boldsymbol{v}\|^2 \end{bmatrix} \in \mathbb{R}^{4\times 4} \tag{A.18}$$

The dynamics of the loss-invisible overlaps can be written in an analogous matrix form. Define

$$\tilde{\boldsymbol{\sigma}} = \left(\sigma_{mu},\, \sigma_{zv},\, \|\boldsymbol{m}\|^2,\, \|\boldsymbol{u}\|^2,\, \|\boldsymbol{v}\|^2,\, \|\boldsymbol{z}\|^2\right)^\top, \quad \tilde{\boldsymbol{D}(\theta)} = \frac{\partial \tilde{\boldsymbol{\sigma}}}{\partial \boldsymbol{\theta}} \tag{A.19}$$

Using $\dot{\boldsymbol{\theta}} = -\boldsymbol{D}^\top(\boldsymbol{\theta})\nabla_{\boldsymbol{\sigma}}\mathcal{L}$, the induced dynamics of $\tilde{\boldsymbol{\sigma}}$ follow as

$$\dot{\tilde{\boldsymbol{\sigma}}} = -\tilde{\boldsymbol{D}}(\boldsymbol{\theta})\boldsymbol{D}(\boldsymbol{\theta})^\top \nabla_{\boldsymbol{\sigma}}\mathcal{L} = -\tilde{\boldsymbol{G}}(\boldsymbol{\theta})\,\nabla_{\boldsymbol{\sigma}}\mathcal{L}, \qquad \tilde{\boldsymbol{G}}(\boldsymbol{\theta}) = \tilde{\boldsymbol{D}}(\boldsymbol{\theta})\boldsymbol{D}^\top(\boldsymbol{\theta}) \tag{A.20}$$

With $\tilde{\boldsymbol{G}(\boldsymbol{\theta})}$ given explicitly as

$$\tilde{\boldsymbol{G}}(\boldsymbol{\theta}) = \frac{1}{N} \begin{bmatrix} \sigma_{zu} & \sigma_{zm} & \sigma_{vu} & \sigma_{vm} \\ \sigma_{vm} & \sigma_{vu} & \sigma_{zm} & \sigma_{zu} \\ 2\sigma_{zm} & 0 & 2\sigma_{vm} & 0 \\ 0 & 2\sigma_{zu} & 0 & 2\sigma_{vu} \\ 0 & 0 & 2\sigma_{vm} & 2\sigma_{vu} \\ 2\sigma_{zm} & 2\sigma_{zu} & 0 & 0 \end{bmatrix} \in \mathbb{R}^{6\times 4} \tag{A.21}$$

That is, the learning dynamics reduce to a closed 10D coupled ODE system

$$\dot{\boldsymbol{\sigma}} = -\boldsymbol{G}(\boldsymbol{\theta})\,\nabla_{\boldsymbol{\sigma}}\mathcal{L}, \qquad \dot{\tilde{\boldsymbol{\sigma}}} = -\tilde{\boldsymbol{G}}(\boldsymbol{\theta})\,\nabla_{\boldsymbol{\sigma}}\mathcal{L} \tag{A.22}$$

which translates parameter-space learning dynamics into overlap-space learning. For a complete treatment, we derive the augmented 10×10 Gram matrix $\bar{\boldsymbol{G}}$ in App. C.

A.4 Filter task

As a concrete example for the rank-1 linear RNN, we consider the simple task of reproducing a first-order exponential filter driven by white-noise input [25] (Fig. 5). The transfer function defines the target response

$$y^\star(t) = a^\star\, e^{-c^\star t} * x(t) \qquad \Longleftrightarrow \qquad H^\star(s) = \frac{a^\star}{s + c^\star} \tag{A.23}$$

where $x(t)$ denotes a white-noise input signal, $a^\star$ is the filter gain, $c^\star > 0$ is the decay rate, and $y^\star(t)$ is the target.

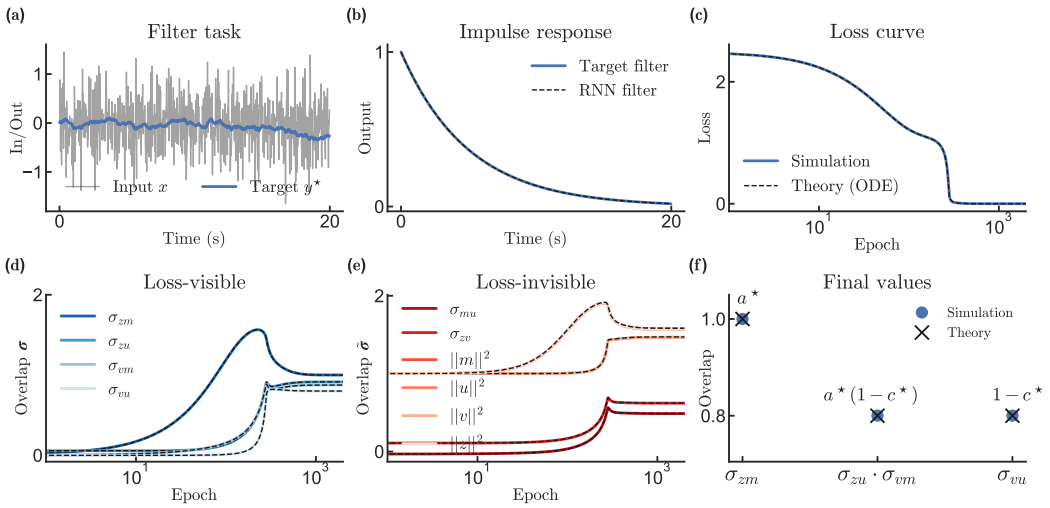

Figure 5: **(a)** Filter task with white-noise input x (gray) and target output $y^\star$ (blue). **(b)** Impulse responses of the target filter (solid blue), with gain $a^\star = 1$ and decay rate $c^\star = 0.2$, and final learned RNN function (dashed black). **(c)** Training loss of the RNN: numerical simulation of the high-dimensional network (solid blue) and the corresponding ODE theory (dashed black), showing excellent agreement. **(d,e)** Dynamics of loss-visible and loss-invisible overlaps, comparing numerical simulations (solid) with ODE predictions (dashed), again in close agreement. **(f)** Final overlap values from simulation (blue circles) compared with theoretical predictions (black crosses).

A.4.1 Training details and RNN initialization

Numerical simulations were performed by training a continuous-time rank-1 RNN discretized via the Euler method ($\Delta t = 0.025$) and size $N = 500$. Every element of the trainable vectors $\boldsymbol{\theta} = \{\boldsymbol{m}, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{z}\} \subset \mathbb{R}^N$ is initialized i.i.d. from a standard normal distribution $\mathcal{N}(0, 1)$. Training is performed by gradient descent with learning rate $\eta = 5 \times 10^{-3}$ to match the filter target with gain $a^\star = 1$ and decay rate $c^\star = 0.2$, using the impulse (delta) response as input, and for an episode of length $T = 20\,\mathrm{s}$. The loss is the time-integrated squared error between the network output and the target. For the A–B–A training protocol, task A filter parameters are as described above, while task B uses the same gain $a^\star = 1$ but a different decay rate $c^\star = 0.4$. In the same protocol, during the noisy training phase, Gaussian noise $\mathcal{N}(0, 0.01)$ is added to the target labels (Fig. 7).

A.4.2 Network impulse response and exact solution

For the effective 2D RNN described in Eqs. (A.3), (A.4), the impulse response can be obtained in closed form. Define

$$A = \begin{bmatrix} -1 & 0 \\ \sigma_{vm} & -1 + \sigma_{vu} \end{bmatrix}, \qquad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad C = \begin{bmatrix} \sigma_{zm} & \sigma_{zu} \end{bmatrix} \tag{A.24}$$

The Laplace transform of the transfer function from the input x to the output $\hat{y}$ reads

$$H(s) = C(sI - A)^{-1}B = \frac{s\,\sigma_{zm} + \sigma_{vm}\sigma_{zu} - \sigma_{vu}\sigma_{zm} + \sigma_{zm}}{(s+1)(s+1-\sigma_{vu})} \tag{A.25}$$

and the corresponding impulse response (kernel) is a sum of two exponentials

$$\Big(\sigma_{zm} - \frac{\sigma_{zu}\sigma_{vm}}{\sigma_{vu}}\Big)e^{-t} \; + \; \frac{\sigma_{zu}\sigma_{vm}}{\sigma_{vu}}\, e^{-(1-\sigma_{vu})t} \tag{A.26}$$

To match the target one–pole filter $a^\star e^{-c^\star t}$ exactly, the unwanted e^{-t} mode must vanish, and the remaining pole and amplitude must be set to $c^\star$ and $a^\star$, respectively. This yields the following constraints

$$\sigma_{vu} = 1 - c^\star, \qquad \frac{\sigma_{zu}\sigma_{vm}}{\sigma_{vu}} = a^\star, \qquad \sigma_{zm} = a^\star \tag{A.27}$$

Notably, this expression depends on σ_{zu} and σ_{vm} only through their product, implying a continuous rescaling symmetry $\sigma_{zu} \to \alpha\, \sigma_{zu},\ \sigma_{vm} \to \alpha^{-1}\sigma_{vm}$ under which the network function is invariant. This further mean, that any pair $(\sigma_{zu}, \sigma_{vm})$ satisfying the product constraint $\sigma_{zu}\sigma_{vm} = a^\star(1 - c^\star)$ provides a valid solution. These constraints reveal a 1D degenerate manifold of global minima in the visible overlap space. Within this manifold, we identify the **balanced solution** as the unique symmetric point where

$$\sigma_{zu} = \sigma_{vm} = \sqrt{a^\star(1 - c^\star)} \tag{A.28}$$

This point is relevant to our drift results, where training under noisy conditions tends to converge toward this solution.

A.4.3 Exact gradient calculation

For the filter task considered here, the gradients with respect to the loss-visible overlaps (i.e., $\nabla_\sigma \mathcal{L}$) can be derived in closed form.

Let $e(t) = \hat{y}(t) - y^\star(t)$ denote the output error. Writing the impulse-response coefficients as

$$A = \sigma_{zm} - \frac{\sigma_{zu}\sigma_{vm}}{\sigma_{vu}}, \qquad B = \frac{\sigma_{zu}\sigma_{vm}}{\sigma_{vu}} \tag{A.29}$$

the output takes the form

$$\hat{y}(t) = A\, e^{-t} + B\, e^{-(1-\sigma_{vu})t} \tag{A.30}$$

Define the difference

$$\delta(t) = e^{-(1-\sigma_{vu})t} - e^{-t} \tag{A.31}$$

The sensitivities of the output with respect to the loss-visible overlaps are

$$\begin{aligned}
\partial_{\sigma_{zm}} \hat{y}(t) &= e^{-t} \\
\partial_{\sigma_{zu}} \hat{y}(t) &= \frac{\sigma_{vm}}{\sigma_{vu}}\, \delta(t) \\
\partial_{\sigma_{vm}} \hat{y}(t) &= \frac{\sigma_{zu}}{\sigma_{vu}}\, \delta(t) \\
\partial_{\sigma_{vu}} \hat{y}(t) &= B\left[-\frac{1}{\sigma_{vu}}\delta(t) + t\, e^{-(1-\sigma_{vu})t}\right]
\end{aligned} \tag{A.32}$$

For the episode loss $\mathcal{L} = \int_0^T \big[\hat{y}(t) - y^\star(t)\big]^2\, dt = \int_0^T e(t)^2\, dt$, the corresponding gradients are

$$\begin{aligned}
\nabla_{zm} &= \partial_{\sigma_{zm}}\mathcal{L} = 2\int_0^T e(t)\, e^{-t}\, dt \\
\nabla_{zu} &= \partial_{\sigma_{zu}}\mathcal{L} = 2\int_0^T e(t)\, \frac{\sigma_{vm}}{\sigma_{vu}}\delta(t)\, dt \\
\nabla_{vm} &= \partial_{\sigma_{vm}}\mathcal{L} = 2\int_0^T e(t)\, \frac{\sigma_{zu}}{\sigma_{vu}}\delta(t)\, dt \\
\nabla_{vu} &= \partial_{\sigma_{vu}}\mathcal{L} = 2\int_0^T e(t)\, B\left[-\frac{1}{\sigma_{vu}}\delta(t) + t\, e^{-(1-\sigma_{vu})t}\right] dt
\end{aligned} \tag{A.33}$$

A.5 Differences between parameter-space and overlap-space learning

As noted in the main text, parameter-space gradient flow induces in overlap space the following dynamics $\dot{\boldsymbol{\sigma}} = -\boldsymbol{G}(\boldsymbol{\theta})\nabla_{\boldsymbol{\sigma}}\mathcal{L}$, whereas direct overlap-space optimization follows $\dot{\boldsymbol{\sigma}} = -\nabla_{\boldsymbol{\sigma}}\mathcal{L}$. The two dynamics follow the same path up to a rescaling of time whenever $\boldsymbol{G}(\boldsymbol{\theta})\nabla_{\boldsymbol{\sigma}}\mathcal{L} = \alpha(t)\nabla_{\boldsymbol{\sigma}}\mathcal{L}$ for some $\alpha(t) > 0$, with $\boldsymbol{G} = \alpha\boldsymbol{I}$ as a sufficient condition. More generally, however, anisotropic diagonal entries or off-diagonal mixing in $\boldsymbol{G}(\theta)$ can alter the trajectory and the converged solution. To illustrate these possibilities, Fig. 6 compares parameter-induced and direct overlap-space learning dynamics on the linear filter task (see A.4) across three initialization regimes. For a balanced i.i.d. initialization, the metric is effectively isotropic, and the dynamics differ only by a time rescaling (blue). Introducing initial correlations through $\sigma_{mu} > 0$ changes the trajectory but not the final solution (green), whereas an unbalanced initialization with $\|\boldsymbol{u}\| \gg \|\boldsymbol{v}\|$ changes both the trajectory and solution selected on the degenerate manifold (red).

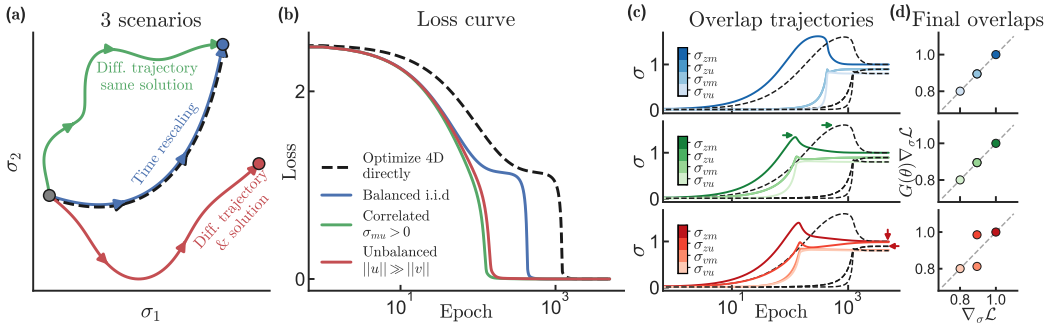

Figure 6: **(a)** Schematic of three possible regimes. Relative to direct overlap-space optimization (black dashed), the induce parameter-space learning may follow the same trajectory up to a rescaling of time (blue), a different trajectory to the same solution (green), or a different trajectory to a different solution (red). **(b)** Concrete examples of these regimes. Loss curves for the filter task shown for direct optimization in overlap space (black dashed) and for parameter-space induced dynamics under balanced i.i.d. (blue), correlated $\sigma_{mu} > 0$ (green), and unbalanced $\|\boldsymbol{u}\| \gg \|\boldsymbol{v}\|$ (red) initializations. **(c)** Corresponding trajectories of the four loss-visible overlaps for the balanced (top), correlated (middle), and unbalanced (bottom) initializations, shown alongside direct overlap-space optimization (black dashed). Arrows highlight differences between the induced and direct trajectories. **(d)** Final overlap values under direct optimization (x-axis) and parameter-space induced dynamics (y-axis). Each point denotes one loss-visible overlap, and the dashed diagonal denotes the identity. The balanced and correlated initializations converge to the same solution as direct optimization, whereas the unbalanced initialization selects a different solution, as indicated by the off-diagonal values.

A.6 An invariant under gradient flow

Define the $N \times 2$ matrices

$$\boldsymbol{A} = [\boldsymbol{z} \quad \boldsymbol{v}], \qquad \boldsymbol{B} = [\boldsymbol{m} \quad \boldsymbol{u}] \tag{A.34}$$

Note that $\boldsymbol{A}$ and $\boldsymbol{B}$ are disjoint, which will be important for the derivation below (and does not hold in the nonlinear case App. B). The four loss-visible overlaps assemble into the 2×2 matrix

$$\frac{1}{N}\boldsymbol{A}^\top \boldsymbol{B} = \frac{1}{N}\begin{bmatrix} \boldsymbol{z}^\top \boldsymbol{m} & \boldsymbol{z}^\top \boldsymbol{u} \\ \boldsymbol{v}^\top \boldsymbol{m} & \boldsymbol{v}^\top \boldsymbol{u} \end{bmatrix} = \begin{bmatrix} \sigma_{zm} & \sigma_{zu} \\ \sigma_{vm} & \sigma_{vu} \end{bmatrix} \tag{A.35}$$

Since the within-episode dynamics, readout, and loss depend on the parameters only through $\frac{1}{N}\boldsymbol{A}^\top \boldsymbol{B}$, the loss can be written as

$$\mathcal{L} = \mathcal{L}(\tfrac{1}{N}\boldsymbol{A}^\top \boldsymbol{B}) \tag{A.36}$$

Defining the 2×2 gradient matrix

$$\boldsymbol{J} = \frac{\partial \mathcal{L}}{\partial(\boldsymbol{A}^\top \boldsymbol{B})} = \begin{bmatrix} \nabla_{zm} & \nabla_{zu} \\ \nabla_{vm} & \nabla_{vu} \end{bmatrix} \tag{A.37}$$

gradient flow in parameter space takes the form

$$\dot{\boldsymbol{A}} = -\boldsymbol{B}\,\boldsymbol{J}^\top, \qquad \dot{\boldsymbol{B}} = -\boldsymbol{A}\,\boldsymbol{J} \tag{A.38}$$

which reproduces Eq. (A.10). Now, consider the matrix difference $\boldsymbol{A}\boldsymbol{A}^\top - \boldsymbol{B}\boldsymbol{B}^\top \in \mathbb{R}^{N \times N}$

$$\frac{d}{d\tau}(\boldsymbol{A}\boldsymbol{A}^\top) = -\boldsymbol{B}\boldsymbol{J}^\top \boldsymbol{A}^\top - \boldsymbol{A}\boldsymbol{J}\boldsymbol{B}^\top \tag{A.39}$$

and similarly

$$\frac{d}{d\tau}(\boldsymbol{B}\boldsymbol{B}^\top) = -\boldsymbol{A}\boldsymbol{J}\boldsymbol{B}^\top - \boldsymbol{B}\boldsymbol{J}^\top \boldsymbol{A}^\top \tag{A.40}$$

Since the RHS of Eqs. (A.39), (A.40) are identical we have that

$$\frac{d}{d\tau}\big(\boldsymbol{A}\boldsymbol{A}^\top - \boldsymbol{B}\boldsymbol{B}^\top\big) = 0 \tag{A.41}$$

Therefore the matrix

$$\boldsymbol{K} = \boldsymbol{A}\boldsymbol{A}^\top - \boldsymbol{B}\boldsymbol{B}^\top = \boldsymbol{z}\boldsymbol{z}^\top + \boldsymbol{v}\boldsymbol{v}^\top - \boldsymbol{m}\boldsymbol{m}^\top - \boldsymbol{u}\boldsymbol{u}^\top \tag{A.42}$$

is conserved under gradient flow and is fixed entirely by the initialization $\big(\boldsymbol{A}(0), \boldsymbol{B}(0)\big)$.

Scalar Invariants While $\boldsymbol{K} \in \mathbb{R}^{N \times N}$ is a high-dimensional matrix, it is constructed from only four vectors, implying $\mathrm{rank}(\boldsymbol{K}) \leq 4$. Consequently, its conservation provides exactly four independent scalar constraints, given by the traces of its powers

$$\mathcal{C}_k = \mathrm{Tr}(\boldsymbol{K}^k), \qquad k = 1, \ldots, 4$$

Using the cyclic property of the trace, these invariants can be expressed directly in terms of the vector norms and pairwise overlaps. The first invariant ($k = 1$) is given by the squared norms of all connectivity vectors

$$\mathcal{C}_1 = \mathrm{Tr}(\boldsymbol{K}) = \|\boldsymbol{z}\|^2 + \|\boldsymbol{v}\|^2 - \|\boldsymbol{m}\|^2 - \|\boldsymbol{u}\|^2 \tag{A.43}$$

The second invariant ($k = 2$) couples the norms and all overlaps

$$\mathcal{C}_2 = \big(\|\boldsymbol{z}\|^4 + \|\boldsymbol{v}\|^4 + 2\sigma_{zv}^2\big) + \big(\|\boldsymbol{m}\|^4 + \|\boldsymbol{u}\|^4 + 2\sigma_{mu}^2\big) - 2\left(\sigma_{zm}^2 + \sigma_{zu}^2 + \sigma_{vm}^2 + \sigma_{vu}^2\right) \tag{A.44}$$

Similarly, $\mathcal{C}_3$ and $\mathcal{C}_4$ expand into increasingly complex overlap combinations. Collectively, these four scalar invariants constrain the 10-dimensional overlap system to a 6-dimensional manifold. From the gradient flow Eq. (A.14), the symmetry in the updates further implies that the differences $\|\boldsymbol{z}\|^2 - \|\boldsymbol{m}\|^2$ and $\|\boldsymbol{v}\|^2 - \|\boldsymbol{u}\|^2$ remain constant whenever $\sigma_{vm}\nabla_{vm} = \sigma_{zu}\nabla_{zu}$, a condition satisfied for the filter task, due to the symmetric dependence of the loss on the product $\sigma_{zu}\sigma_{vm}$ (see Eq. (A.27)). Under these combined structural and task-specific constraints, there are insufficient degrees of freedom for the six loss-invisible overlaps to drift independently. Their evolution is thus tied to the initialization $\boldsymbol{K}$, leading to an exact recovery observed in the A–B–A protocol.

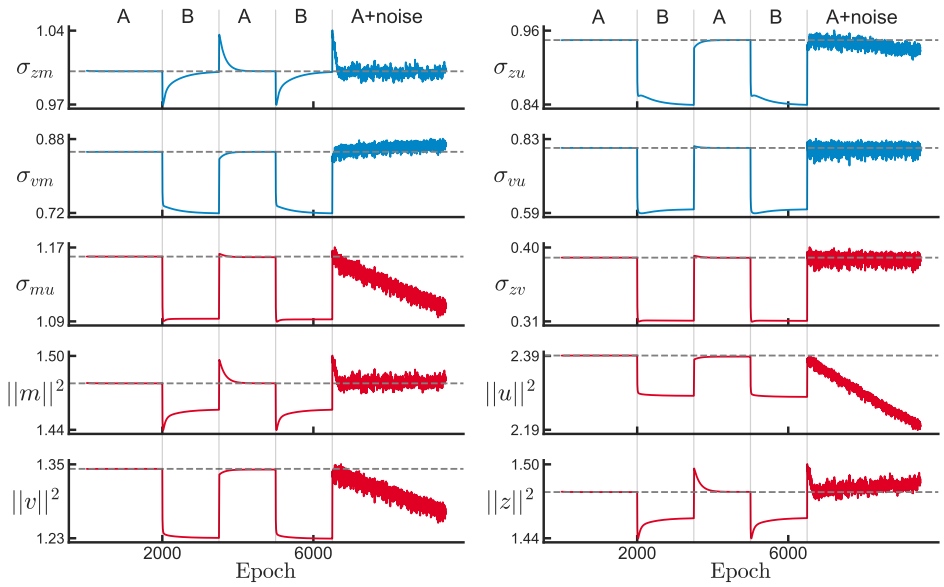

Figure 7: Trajectories of all ten overlaps in the A-B-A protocol, supplementing Fig. 3 of the main text. Blue traces represent loss-visible overlaps, and red traces depict loss-invisible ones. During the initial ABAB phases, all overlaps return to their previous values, demonstrating a lack of memory. However, when noise is added to the training process, we observe a distinct drift, which is most pronounced in the loss-invisible overlaps. However, also note that the loss-visible overlaps can exhibit drift under noisy conditions (σ_{vm} and σ_{zu}); this is understood through the degenerate task condition, where the filter task only constrains the product $\sigma_{vm}\sigma_{zu}$, allowing noise to drive the system toward a balanced solution where these individual overlaps equalize (see App. A.4.2).

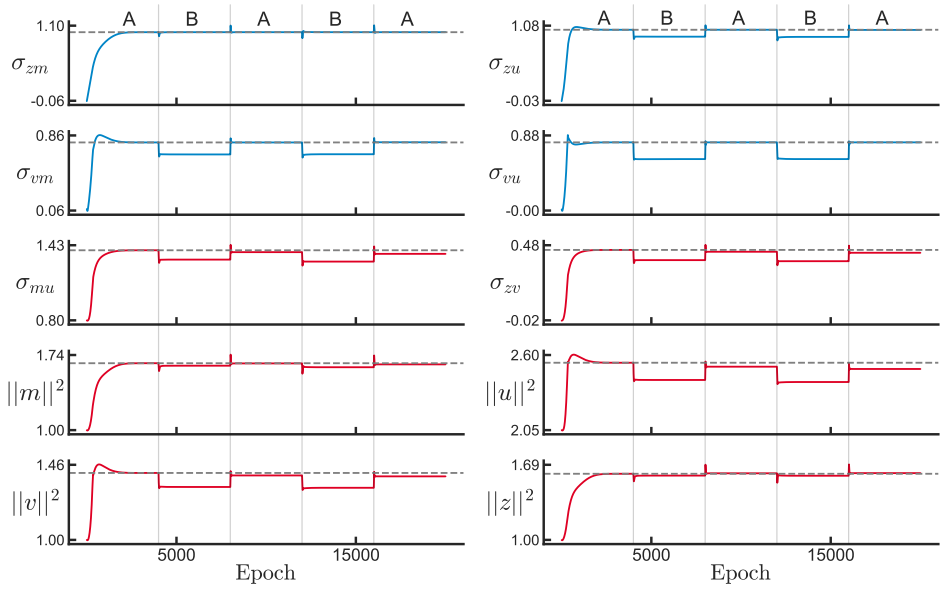

Figure 8: Trajectories of all ten overlaps in the A-B-A protocol using the Adam optimizer ($\eta = 10^{-3}$). Blue traces represent loss-visible overlaps, and red traces depict loss-invisible ones. Note that, unlike vanilla gradient descent, the adaptive optimizer violates the learning invariant, allowing the loss-invisible overlaps to settle to new values upon retraining, demonstrating history-dependent (red traces do not return to the gray dashed baseline).

B Nonlinear rank-1 RNN

We provide here a complete derivation of the learning dynamics of the nonlinear rank-1 RNN.

B.1 Within-episode dynamics

We consider a rank-1 recurrent network with nonlinear activation

$$\dot{\boldsymbol{h}}(t) = -\boldsymbol{h}(t) + \frac{1}{N}\boldsymbol{u}\boldsymbol{v}^\top \phi(\alpha \boldsymbol{h}(t)) + \boldsymbol{m}\, x(t), \qquad \phi(x) = \operatorname{erf}(x), \qquad \alpha = \sqrt{\pi}/2 \tag{B.1}$$

where the choice $\alpha = \sqrt{\pi}/2$ ensures that the activation is locally linear with unit slope at the origin. As in the linear case, the recurrent term lies in $\operatorname{span}\{\boldsymbol{u}\}$ and the input term lies in $\operatorname{span}\{\boldsymbol{m}\}$. Assuming $\boldsymbol{h}(0) = \mathbf{0}$, the state remains confined to

$$\boldsymbol{h}(t) = \kappa_m(t)\boldsymbol{m} + \kappa_u(t)\boldsymbol{u}, \qquad \boldsymbol{\kappa}(t) = \begin{bmatrix}\kappa_m(t)\\ \kappa_u(t)\end{bmatrix} \in \mathbb{R}^2 \tag{B.2}$$

Assuming that the parameter vectors have zero-mean, the activity also has zero mean, with empirical variance

$$\Delta(t) = \frac{1}{N}\boldsymbol{h}(t)^\top \boldsymbol{h}(t) = \kappa_m^2\|\boldsymbol{m}\|^2 + \kappa_u^2\|\boldsymbol{u}\|^2 + 2\kappa_m\kappa_u\sigma_{mu} \tag{B.3}$$

In the large-N limit, neuron averages concentrate to expectations

$$\frac{1}{N}\boldsymbol{v}^\top \phi(\alpha\boldsymbol{h}) = \frac{1}{N}\sum_i v_i\phi(\alpha h_i) \xrightarrow{N\to\infty} \mathbb{E}[v_i\phi(\alpha h_i)] \tag{B.4}$$

Following the dynamical mean-field theory (DMFT) framework [27, 16], we assume that the components of the parameter vectors are jointly Gaussian. Since $h_i(t) = \kappa_m(t)m_i + \kappa_u(t)u_i$, it follows that $h_i(t) \sim \mathcal{N}(0, \Delta(t))$. Applying Stein's lemma to (v_i, h_i) gives

$$\mathbb{E}[v_i\phi(\alpha h_i)] = \operatorname{Cov}(v_i, h_i)\,\langle\phi'\rangle \tag{B.5}$$

where, for the error-function nonlinearity, that gain is given in closed-form

$$\langle\phi'\rangle = \mathbb{E}_{g\sim\mathcal{N}(0,\Delta(t))}\left[\alpha\phi'(\alpha g)\right] = \left(1 + \frac{\pi}{2}\Delta(t)\right)^{-1/2} \tag{B.6}$$

Using $h_i = \kappa_m m_i + \kappa_u u_i$ and the zero-mean assumption

$$\operatorname{Cov}(v_i, h_i) = \kappa_m \operatorname{Cov}(v_i, m_i) + \kappa_u \operatorname{Cov}(v_i, u_i) = \sigma_{vm}\kappa_m + \sigma_{vu}\kappa_u \tag{B.7}$$

Combining all together yields

$$\frac{1}{N}\boldsymbol{v}^\top \phi(\alpha\boldsymbol{h}) = (\sigma_{vm}\kappa_m + \sigma_{vu}\kappa_u)\,\langle\phi'\rangle \tag{B.8}$$

Substituting into the dynamics, the N-dimensional system reduces to

$$\begin{aligned}\dot{\kappa}_m(t) &= -\kappa_m(t) + x(t)\\ \dot{\kappa}_u(t) &= -\kappa_u(t) + (\sigma_{vm}\kappa_m(t) + \sigma_{vu}\kappa_u(t))\,\langle\phi'\rangle\end{aligned} \tag{B.9}$$

Applying the same argument to (z_i, h_i) gives

$$\hat{y}(t) = (\sigma_{zm}\kappa_m(t) + \sigma_{zu}\kappa_u(t))\,\langle\phi'\rangle \tag{B.10}$$

Note that, unlike in the linear case, the variance $\Delta(t)$ depends on the overlap σ_{mu} as well as the norms $\|\boldsymbol{m}\|^2$ and $\|\boldsymbol{u}\|^2$. As a result, these quantities directly influence the network dynamics and output, and therefore become *loss-visible*.

B.2 Across-episode learning dynamics

The episode loss depends on the parameters only through the new set of loss-visible overlaps

$$\mathcal{L} = \mathcal{L}\big(\sigma_{zm}, \sigma_{zu}, \sigma_{vm}, \sigma_{vu}, \sigma_{mu}, \|\boldsymbol{m}\|^2, \|\boldsymbol{u}\|^2\big) \tag{B.11}$$

Using $\nabla_{mm} = \partial\mathcal{L}/\partial\|\boldsymbol{m}\|^2$ and $\nabla_{uu} = \partial\mathcal{L}/\partial\|\boldsymbol{u}\|^2$, together with $\partial\|\boldsymbol{m}\|^2/\partial\boldsymbol{m} = 2\boldsymbol{m}/N$ and $\partial\|\boldsymbol{u}\|^2/\partial\boldsymbol{u} = 2\boldsymbol{u}/N$, the chain rule gives

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{m}} = \frac{1}{N}(\nabla_{zm}\,\boldsymbol{z} + \nabla_{vm}\,\boldsymbol{v} + \nabla_{mu}\,\boldsymbol{u} + 2\nabla_{mm}\,\boldsymbol{m}) \tag{B.12}$$

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{u}} = \frac{1}{N}(\nabla_{zu}\,\boldsymbol{z} + \nabla_{vu}\,\boldsymbol{v} + \nabla_{mu}\,\boldsymbol{m} + 2\nabla_{uu}\,\boldsymbol{u}) \tag{B.13}$$

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{z}} = \frac{1}{N}(\nabla_{zm}\,\boldsymbol{m} + \nabla_{zu}\,\boldsymbol{u}) \tag{B.14}$$

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{v}} = \frac{1}{N}(\nabla_{vm}\,\boldsymbol{m} + \nabla_{vu}\,\boldsymbol{u})$$

Under gradient flow $\dot{\boldsymbol{\theta}} = -\nabla_{\boldsymbol{\theta}}\mathcal{L}$ we obtain

$$\begin{aligned}
\dot{\boldsymbol{m}} &= -\frac{1}{N}(\nabla_{zm}\boldsymbol{z} + \nabla_{vm}\boldsymbol{v} + \nabla_{mu}\boldsymbol{u} + 2\nabla_{mm}\boldsymbol{m}) \\
\dot{\boldsymbol{u}} &= -\frac{1}{N}(\nabla_{zu}\boldsymbol{z} + \nabla_{vu}\boldsymbol{v} + \nabla_{mu}\boldsymbol{m} + 2\nabla_{uu}\boldsymbol{u}) \\
\dot{\boldsymbol{v}} &= -\frac{1}{N}(\nabla_{vm}\boldsymbol{m} + \nabla_{vu}\boldsymbol{u}) \\
\dot{\boldsymbol{z}} &= -\frac{1}{N}(\nabla_{zm}\boldsymbol{m} + \nabla_{zu}\boldsymbol{u})
\end{aligned} \tag{B.15}$$

By the product rule of the overlaps, we obtain the dynamics for the loss-visible overlaps

$$\begin{aligned}
\dot{\sigma}_{zm} &= -(\|\boldsymbol{m}\|^2 + \|\boldsymbol{z}\|^2)\nabla_{zm} - \sigma_{mu}\nabla_{zu} - \sigma_{zv}\nabla_{vm} - \sigma_{zu}\nabla_{mu} - 2\nabla_{mm}\sigma_{zm} \\
\dot{\sigma}_{zu} &= -\sigma_{mu}\nabla_{zm} - (\|\boldsymbol{u}\|^2 + \|\boldsymbol{z}\|^2)\nabla_{zu} - \sigma_{zv}\nabla_{vu} - \sigma_{zm}\nabla_{mu} - 2\nabla_{uu}\sigma_{zu} \\
\dot{\sigma}_{vm} &= -(\|\boldsymbol{m}\|^2 + \|\boldsymbol{v}\|^2)\nabla_{vm} - \sigma_{mu}\nabla_{vu} - \sigma_{zv}\nabla_{zm} - \sigma_{vu}\nabla_{mu} - 2\nabla_{mm}\sigma_{vm} \\
\dot{\sigma}_{vu} &= -\sigma_{mu}\nabla_{vm} - (\|\boldsymbol{u}\|^2 + \|\boldsymbol{v}\|^2)\nabla_{vu} - \sigma_{zv}\nabla_{zu} - \sigma_{vm}\nabla_{mu} - 2\nabla_{uu}\sigma_{vu} \\
\dot{\sigma}_{mu} &= -\sigma_{zu}\nabla_{zm} - \sigma_{zm}\nabla_{zu} - \sigma_{vu}\nabla_{vm} - \sigma_{vm}\nabla_{vu} - (\|\boldsymbol{m}\|^2 + \|\boldsymbol{u}\|^2)\nabla_{mu} - 2(\nabla_{mm} + \nabla_{uu})\sigma_{mu} \\
\dot{\|\boldsymbol{m}\|^2} &= -2\Big(\sigma_{zm}\nabla_{zm} + \sigma_{vm}\nabla_{vm} + \sigma_{mu}\nabla_{mu} + 2\nabla_{mm}\|\boldsymbol{m}\|^2\Big) \\
\dot{\|\boldsymbol{u}\|^2} &= -2\Big(\sigma_{zu}\nabla_{zu} + \sigma_{vu}\nabla_{vu} + \sigma_{mu}\nabla_{mu} + 2\nabla_{uu}\|\boldsymbol{u}\|^2\Big)
\end{aligned} \tag{B.16}$$

as well as the loss-invisible dynamics

$$\begin{aligned}
\dot{\sigma}_{zv} &= -\sigma_{vm}\nabla_{zm} - \sigma_{vu}\nabla_{zu} - \sigma_{zm}\nabla_{vm} - \sigma_{zu}\nabla_{vu} \\
\dot{\|\boldsymbol{v}\|^2} &= -2\Big(\sigma_{vm}\nabla_{vm} + \sigma_{vu}\nabla_{vu}\Big) \\
\dot{\|\boldsymbol{z}\|^2} &= -2\Big(\sigma_{zm}\nabla_{zm} + \sigma_{zu}\nabla_{zu}\Big)
\end{aligned} \tag{B.17}$$

B.3 Learning dynamics in matrix form

Define

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{zm} \\ \sigma_{zu} \\ \sigma_{vm} \\ \sigma_{vu} \\ \sigma_{mu} \\ \|\boldsymbol{m}\|^2 \\ \|\boldsymbol{u}\|^2 \end{bmatrix}, \qquad \tilde{\boldsymbol{\sigma}} = \begin{bmatrix} \sigma_{zv} \\ \|\boldsymbol{v}\|^2 \\ \|\boldsymbol{z}\|^2 \end{bmatrix}, \qquad \nabla_{\boldsymbol{\sigma}}\mathcal{L} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{\sigma}} = \begin{bmatrix} \nabla_{zm} \\ \nabla_{zu} \\ \nabla_{vm} \\ \nabla_{vu} \\ \nabla_{mu} \\ \nabla_{\|\boldsymbol{m}\|^2} \\ \nabla_{\|\boldsymbol{u}\|^2} \end{bmatrix} \tag{B.18}$$

The loss-visible Gram matrix $\boldsymbol{G} \in \mathbb{R}^{7\times 7}$ is

$$\boldsymbol{G}(\boldsymbol{\theta}) = \frac{1}{N}\begin{bmatrix} \|\boldsymbol{m}\|^2 + \|\boldsymbol{z}\|^2 & \sigma_{mu} & \sigma_{zv} & 0 & \sigma_{zu} & 2\sigma_{zm} & 0 \\ \sigma_{mu} & \|\boldsymbol{u}\|^2 + \|\boldsymbol{z}\|^2 & 0 & \sigma_{zv} & \sigma_{zm} & 0 & 2\sigma_{zu} \\ \sigma_{zv} & 0 & \|\boldsymbol{m}\|^2 + \|\boldsymbol{v}\|^2 & \sigma_{mu} & \sigma_{vu} & 2\sigma_{vm} & 0 \\ 0 & \sigma_{zv} & \sigma_{mu} & \|\boldsymbol{u}\|^2 + \|\boldsymbol{v}\|^2 & \sigma_{vm} & 0 & 2\sigma_{vu} \\ \sigma_{zu} & \sigma_{zm} & \sigma_{vu} & \sigma_{vm} & \|\boldsymbol{m}\|^2 + \|\boldsymbol{u}\|^2 & 2\sigma_{mu} & 2\sigma_{mu} \\ 2\sigma_{zm} & 0 & 2\sigma_{vm} & 0 & 2\sigma_{mu} & 4\|\boldsymbol{m}\|^2 & 0 \\ 0 & 2\sigma_{zu} & 0 & 2\sigma_{vu} & 2\sigma_{mu} & 0 & 4\|\boldsymbol{u}\|^2 \end{bmatrix} \tag{B.19}$$

and the loss-invisible matrix $\tilde{\boldsymbol{G}} \in \mathbb{R}^{3\times 7}$ is

$$\tilde{\boldsymbol{G}}(\boldsymbol{\theta}) = \frac{1}{N}\begin{bmatrix} \sigma_{vm} & \sigma_{vu} & \sigma_{zm} & \sigma_{zu} & 0 & 0 & 0 \\ 0 & 0 & 2\sigma_{vm} & 2\sigma_{vu} & 0 & 0 & 0 \\ 2\sigma_{zm} & 2\sigma_{zu} & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{B.20}$$

Using these definitions, the learning dynamics can be written compactly as

$$\dot{\boldsymbol{\sigma}} = -\boldsymbol{G}(\boldsymbol{\theta})\,\nabla_{\boldsymbol{\sigma}}\mathcal{L}, \qquad \dot{\tilde{\boldsymbol{\sigma}}} = -\tilde{\boldsymbol{G}}(\boldsymbol{\theta})\,\nabla_{\boldsymbol{\sigma}}\mathcal{L} \tag{B.21}$$

B.4 Flip-flop task

For our nonlinear presentation, we consider the 1-bit flip-flop task [36] (Fig. 10d). The task requires the network to act as a bistable memory: it must maintain a constant output and only "flip" its state upon receiving a brief, signed input pulse. This behavior requires the creation of stable fixed points separated by a nonlinear boundary. During each trial, the network receives a sequence of short input pulses, each of duration t_{stim}. During a pulse, the input channel is set to $x(t) = s\,x_{\mathrm{amp}}$, where $x_{\mathrm{amp}} = 1$, and the sign $s \in \{\pm 1\}$ is chosen at random. Each pulse is followed by a delay period of duration t_{delay}, after which a decision period begins. During this decision period, the loss is activated (i.e., a mask is set to 1), and the target value is defined as $y(t) = s\,y_{\mathrm{amp}}$, with $y_{\mathrm{amp}} = 0.5$. The decision period ends when the next pulse begins. The inter-stimulus delays t_{isd} are drawn randomly.

B.4.1 Training details, RNN initialization and Gaussian assumption

Numerical simulations were performed by training a continuous-time rank-1 RNN discretized Euler method with time step $\Delta t = 0.025$ and network size $N = 1000$. Every element of the trainable vectors $\boldsymbol{\theta} = \{\boldsymbol{m}, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{z}\} \subset \mathbb{R}^N$ is initialized i.i.d. from a standard normal distribution $\mathcal{N}(0, 1)$. Training is performed using gradient descent with learning rate $\eta = 0.05$ over episodes of length $T = 20\,\mathrm{s}$ and batch size of 10. We use a masked mean-squared error (MSE) loss that ignores the output during input pulses and short transients, thereby focusing learning on maintaining stable fixed-point outputs.

Furthermore, unlike the linear case, the nonlinear theory relies on the components of the high-dimensional parameter vectors $\boldsymbol{\theta}$ remaining approximately Gaussian during training. Here, we complement the main text by (i) verifying that the weights remain approximately Gaussian under a small learning rate ($\eta = 0.05$; Fig. 9a), and (ii) showing that when this assumption breaks (e.g., with larger learning rates ($\eta = 0.5$) or Adam ($\eta = 0.001$)), the theory is no longer valid (Fig 9b,c).

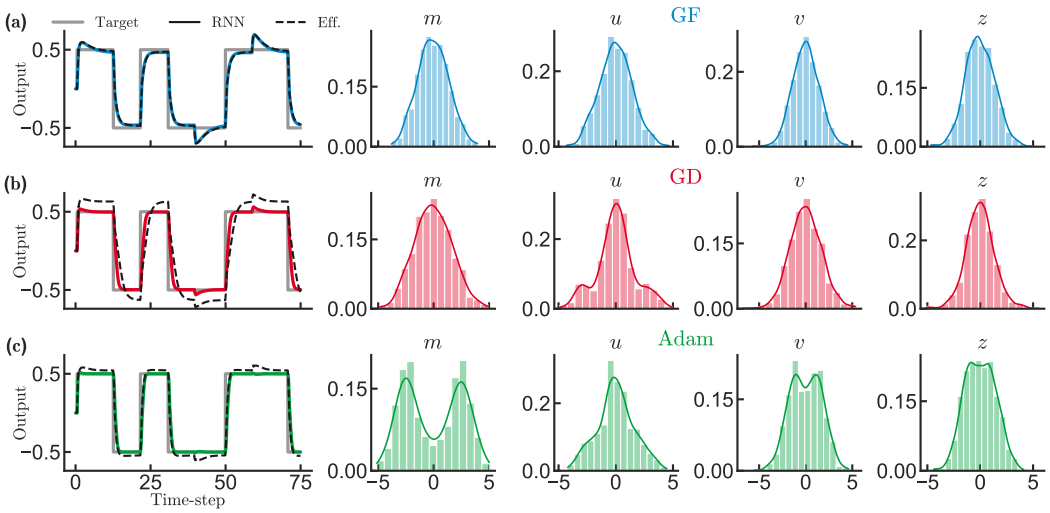

Figure 9: Training on flip-flop task with different optimizers. **(a)** gradient flow (GF; small η), **(b)** gradient descent (GD), and **(c)** Adam. Left column: target signals (solid gray), high-dimensional RNN outputs (solid colors), and effective RNN models (dashed black). Right columns: empirical distributions of the components of the parameter vectors $\boldsymbol{\theta} \in \{\boldsymbol{m}, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{z}\}$ at convergence. Under GF (top), these distributions remain approximately Gaussian, and the effective model accurately captures the simulated outputs, consistent with theory. In contrast, under GD (middle) and Adam (bottom), the Gaussian assumption breaks down, and the effective theory fails to capture simulated output.

B.5 History-dependent training protocol

We consider a history-dependent training protocol consisting of three tasks. Task A is identical to the flip-flop task described above (B.4). Task B is a stimulus-integration decision-making task (see below). And Task C is a teacher-student task, where both networks are trained to reproduce the output of a pre-defined teacher network in response to white noise input (Fig. 10f). The teacher network overlaps are defined as

$$\big(\sigma_{zm}, \sigma_{zu}, \sigma_{vm}, \sigma_{vu}, \sigma_{mu}, \|\boldsymbol{m}\|^2, \|\boldsymbol{u}\|^2\big) = \big(0.5,\ 2.3,\ 2.0,\ 1.5,\ 1.6,\ 1.8,\ 2.2\big)$$

Training proceeds in two phases: Phase A/B and Phase C. In Phase A/B, networks are trained on either Task A or Task B for 30,000 epochs, using an initial learning rate of $\eta = 0.01$. The learning rate is reduced by a factor of 5 whenever the loss falls below 0.015. In Phase C, training continues for an additional 30,000 epochs with a fixed learning rate of $\eta = 0.001$. To ensure consistency with the Gaussian assumptions underlying the low-dimensional nonlinear theory, training is performed directly in the overlap space using the corresponding preconditioned $\boldsymbol{G}$. The initial overlap values are sampled as follows: cross-overlaps $\sigma_{zm}, \sigma_{zu}, \sigma_{vm}, \sigma_{vu}, \sigma_{mu}, \sigma_{zv}$ are drawn from $|\mathcal{N}(0, 0.4)|$, and the squared norms $\|\boldsymbol{m}\|^2, \|\boldsymbol{u}\|^2, \|\boldsymbol{v}\|^2, \|\boldsymbol{z}\|^2$ are drawn uniformly from $\mathcal{U}[0.5, 2.0]$. Within-episode dynamics are integrated with time step $\Delta t = 0.05$ over episodes of length $T = 20\,\mathrm{s}$. For both phases, we used a batch size of 128.

Decision-making task This task requires the network to integrate a noisy evidence input to produce a continuous output proportional to the stimulus strength (Fig. 10e). Each trial consists of a stimulus period of duration t_{stim}, a brief delay t_{delay}, and a response period. During the stimulus period, the input channel receives a noisy signal $x(t) = c + \xi(t)$, where c is a coherence level chosen from a discrete set $\mathcal{C} \in \{\pm 2, \pm 8, \pm 16\}$ and $\xi(t) \sim \mathcal{N}(0, 0.05^2)$ is zero-mean Gaussian noise. During the response period the target value is defined as $y(t) = y_{\mathrm{amp}}(c/c_{\max})$, with $y_{\mathrm{amp}} = 1.0$ and $c_{\max}$ is the maximum absolute coherence. The network is trained using a masked mean squared error (MSE) loss that is active only during the decision period.

Classification To test whether task history can be decoded from the overlaps, we trained a logistic-regression classifier (the precise choice of classifier is not critical; similar results are obtained using SVM) to classify networks first trained on Task A from those first trained on Task B. Overlap vectors

were extracted from 10 independent runs (10 $\times$ 2 twins network) at three stages: initialization, after Phase A/B, and after Phase C. Classification was performed separately using either the loss-visible or loss-invisible overlaps. At each stage, the 20 samples (10 per class) were split into training and test sets (16/4), and evaluation was repeated over 50 random splits. Performance is reported as the mean and standard deviation of test accuracy across splits. Importantly, before classification, white noise was added to the overlap features to both reflect realistic variability and prevent the classifier from exploiting infinitesimal differences arising from imperfect convergence (see Fig. 11; the results are largely insensitive to the precise choice of noise level).

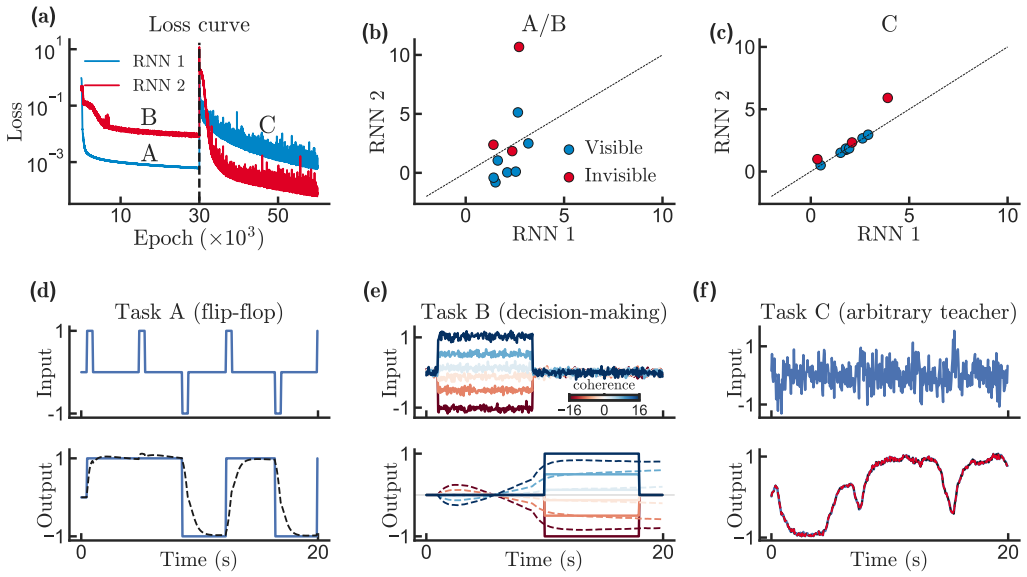

Figure 10: **(a)** Training loss for the A/B $\rightarrow$ C protocol, for an example run of network 1 (A$\rightarrow$C; blue) and network 2 (B$\rightarrow$C; red). **(b)** Overlaps at the end of phase 1 (A/B; epoch 30,000), showing that both loss-visible (blue) and loss-invisible (red) overlaps settle to distinct values. **(c)** After training on task C (epoch 60,000), loss-visible overlaps converge to the same values, while loss-invisible overlaps remain distinct. **(d–f)** Example inputs (top) and target vs. predicted outputs (bottom) for the three tasks: (d) flip-flop, (e) stimulus integration (showing 6 different coherence levels), and (f) arbitrary teacher signal. Solid and dashed lines denote targets and network predictions, respectively.

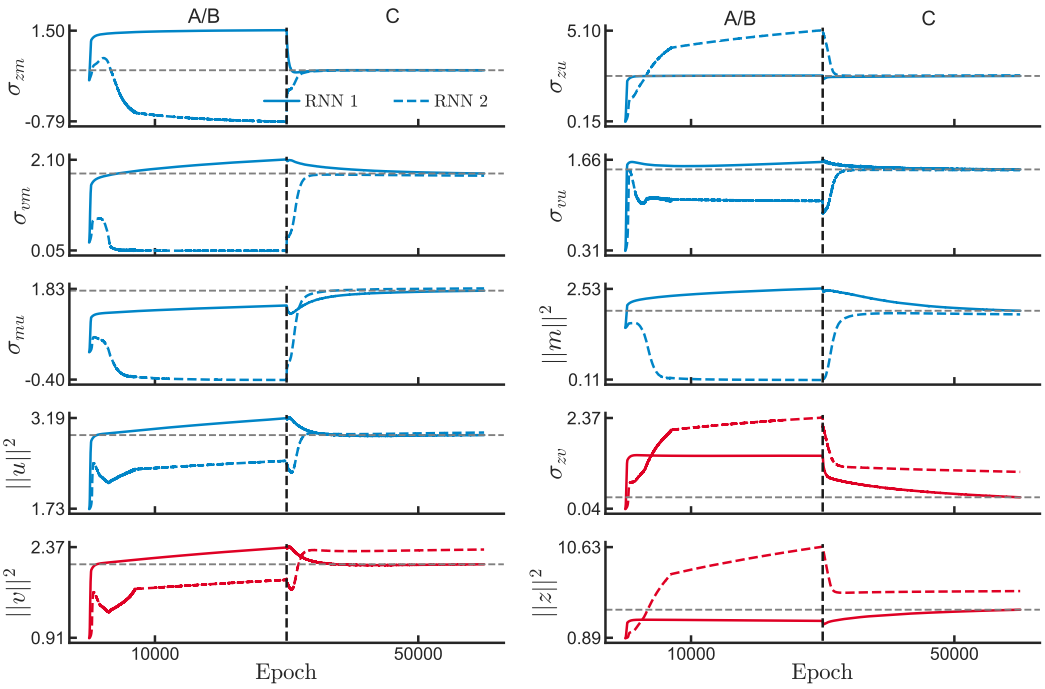

Figure 11: Trajectories of all ten overlaps in the A/B $\rightarrow$ C for an example run. Blue traces denote loss-visible overlaps, red traces denote loss-invisible overlaps, with solid and dashed lines corresponding to networks 1 and 2, respectively. After retraining on task C, loss-visible overlaps converge to the same values (up to infinitesimal differences due to imperfect convergence), while loss-invisible overlaps retain distinct values, reflecting history-dependent memory. Gray dashed lines denote the converged overlap values of network 1 for Task C.

C Derivation of the augmented Gram matrix $\bar{G}$

To derive the augmented $\bar{\boldsymbol{G}}$ matrix, we define $\bar{\boldsymbol{\sigma}}(\boldsymbol{\theta})$ as the exhaustive collection of all quadratic scalars formable from the four vectors $\{\boldsymbol{m}, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{z}\}$. This set comprises the six pairwise overlaps and the four squared norms, encompassing both loss-visible and loss-invisible quantities

$$\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{m} \\ \boldsymbol{u} \\ \boldsymbol{v} \\ \boldsymbol{z} \end{bmatrix} \in \mathbb{R}^{4N}, \qquad \bar{\boldsymbol{\sigma}}(\boldsymbol{\theta}) = \begin{bmatrix} \sigma_{zm} \\ \sigma_{zu} \\ \sigma_{vm} \\ \sigma_{vu} \\ \sigma_{mu} \\ \sigma_{zv} \\ \|\boldsymbol{m}\|^2 \\ \|\boldsymbol{u}\|^2 \\ \|\boldsymbol{v}\|^2 \\ \|\boldsymbol{z}\|^2 \end{bmatrix} \in \mathbb{R}^{10} \tag{C.1}$$

The full Jacobian is given by

$$\bar{\boldsymbol{D}}(\boldsymbol{\theta}) = \frac{\partial \bar{\boldsymbol{\sigma}}}{\partial \boldsymbol{\theta}} = \frac{1}{N} \begin{bmatrix} \boldsymbol{z}^\top & 0 & 0 & \boldsymbol{m}^\top \\ 0 & \boldsymbol{z}^\top & 0 & \boldsymbol{u}^\top \\ \boldsymbol{v}^\top & 0 & \boldsymbol{m}^\top & 0 \\ 0 & \boldsymbol{v}^\top & \boldsymbol{u}^\top & 0 \\ \boldsymbol{u}^\top & \boldsymbol{m}^\top & 0 & 0 \\ 0 & 0 & \boldsymbol{z}^\top & \boldsymbol{v}^\top \\ 2\boldsymbol{m}^\top & 0 & 0 & 0 \\ 0 & 2\boldsymbol{u}^\top & 0 & 0 \\ 0 & 0 & 2\boldsymbol{v}^\top & 0 \\ 0 & 0 & 0 & 2\boldsymbol{z}^\top \end{bmatrix} \in \mathbb{R}^{10 \times 4N} \tag{C.2}$$

and the associated 10×10 Gram matrix, defined as $\bar{\boldsymbol{G}}(\boldsymbol{\theta}) = \bar{\boldsymbol{D}}(\boldsymbol{\theta})\bar{\boldsymbol{D}}(\boldsymbol{\theta})^\top$

$$\frac{1}{N} \begin{bmatrix} \|\boldsymbol{z}\|^2 + \|\boldsymbol{m}\|^2 & \sigma_{mu} & \sigma_{zv} & 0 & \sigma_{zu} & \sigma_{vm} & 2\sigma_{zm} & 0 & 0 & 2\sigma_{zm} \\ \sigma_{mu} & \|\boldsymbol{z}\|^2 + \|\boldsymbol{u}\|^2 & 0 & \sigma_{zv} & \sigma_{zm} & \sigma_{vu} & 0 & 2\sigma_{zu} & 0 & 2\sigma_{zu} \\ \sigma_{zv} & 0 & \|\boldsymbol{v}\|^2 + \|\boldsymbol{m}\|^2 & \sigma_{mu} & \sigma_{vu} & \sigma_{zm} & 2\sigma_{vm} & 0 & 2\sigma_{vm} & 0 \\ 0 & \sigma_{zv} & \sigma_{mu} & \|\boldsymbol{v}\|^2 + \|\boldsymbol{u}\|^2 & \sigma_{vm} & \sigma_{zu} & 0 & 2\sigma_{vu} & 2\sigma_{vu} & 0 \\ \sigma_{zu} & \sigma_{zm} & \sigma_{vu} & \sigma_{vm} & \|\boldsymbol{m}\|^2 + \|\boldsymbol{u}\|^2 & 0 & 2\sigma_{mu} & 2\sigma_{mu} & 0 & 0 \\ \sigma_{vm} & \sigma_{vu} & \sigma_{zm} & \sigma_{zu} & 0 & \|\boldsymbol{z}\|^2 + \|\boldsymbol{v}\|^2 & 0 & 0 & 2\sigma_{zv} & 2\sigma_{zv} \\ 2\sigma_{zm} & 0 & 2\sigma_{vm} & 0 & 2\sigma_{mu} & 0 & 4\|\boldsymbol{m}\|^2 & 0 & 0 & 0 \\ 0 & 2\sigma_{zu} & 0 & 2\sigma_{vu} & 2\sigma_{mu} & 0 & 0 & 4\|\boldsymbol{u}\|^2 & 0 & 0 \\ 0 & 0 & 2\sigma_{vm} & 2\sigma_{vu} & 0 & 2\sigma_{zv} & 0 & 0 & 4\|\boldsymbol{v}\|^2 & 0 \\ 2\sigma_{zm} & 2\sigma_{zu} & 0 & 0 & 0 & 2\sigma_{zv} & 0 & 0 & 0 & 4\|\boldsymbol{z}\|^2 \end{bmatrix} \tag{C.3}$$

Crucially, these derivations hold for any RNN where the connectivity is formed by these four vectors. While the network's nonlinearity changes the functional form of the loss, it does not change the Jacobian $\bar{\boldsymbol{D}}$ or the Gram matrix $\bar{\boldsymbol{G}}$. The network linearity simply sets the boundary between visible and invisible overlaps, shaping how gradient flow evolves in overlap space (Fig. 12).

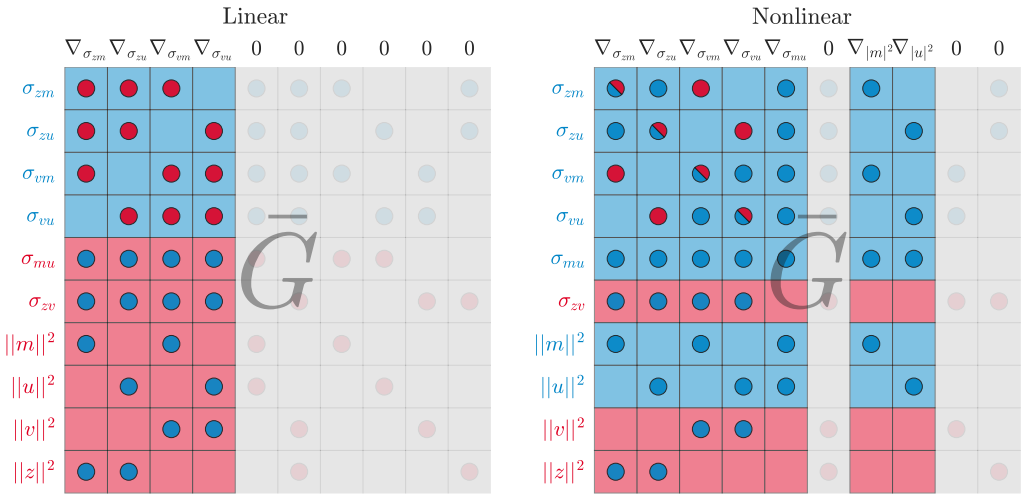

Figure 12: Augmented 10×10 Gram matrix $\bar{\boldsymbol{G}}$ for the linear (left) and nonlinear (right) rank-1 RNNs. Rows correspond to overlaps being updated, and columns to the associated loss gradients. Blue letter/dots denote loss-visible overlaps, while red letter/dots denote loss-invisible overlaps. Colored circles indicate the coefficient of $\bar{\boldsymbol{G}}$, for which the gradient is nonzero. In the linear case, the structure is cleanly separated: updates of loss-visible overlaps depend only on the coefficients of loss-invisible quantities, and vice versa. Note that the top-left blue block of the linear matrix corresponds exactly to the $\boldsymbol{G}(\boldsymbol{\theta})$ matrix derived in Eq. 10 of the main text. In the nonlinear case, this separation is broken. Blue–red mixed circles highlight entries where visible and invisible quantities are coupled, implying that the learning dynamics depend jointly on both sets of overlaps. This mixing reflects the fact that quantities that are invisible in the linear model become visible in the nonlinear model.

D Rank-2 linear RNN

We provide a brief extension of our analysis to a rank-2 linear RNN with recurrent connectivity

$$\boldsymbol{W} = \frac{1}{N}\sum_{j=1}^{2} \boldsymbol{u}_j \boldsymbol{v}_j^\top \tag{D.1}$$

The parameter set consists of six vectors $\{\boldsymbol{m}, \boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{v}_2, \boldsymbol{v}_2, \boldsymbol{z}\} \subset \mathbb{R}^N$. With zero initial condition $\boldsymbol{h}(0) = \boldsymbol{0}$, the dynamics remain confined to the 3-dimensional subspace $\mathrm{span}\{\boldsymbol{m}, \boldsymbol{u}_1, \boldsymbol{u}_2\}$. We write

$$\boldsymbol{h}(t) = \kappa_m(t)\boldsymbol{m} + \kappa_{u_1}(t)\boldsymbol{u}_1 + \kappa_{u_2}(t)\boldsymbol{u}_2, \qquad \boldsymbol{\kappa}(t) = \begin{bmatrix} \kappa_m \\ \kappa_{u_1} \\ \kappa_{u_2} \end{bmatrix} \tag{D.2}$$

The effective within-episode dynamics and readout are

$$\begin{aligned} \dot{\boldsymbol{\kappa}}(t) &= -\boldsymbol{\kappa}(t) + \begin{bmatrix} 0 & 0 & 0 \\ \sigma_{v_1 m} & \sigma_{v_1 u_1} & \sigma_{v_1 u_2} \\ \sigma_{v_2 m} & \sigma_{v_2 u_1} & \sigma_{v_2 u_2} \end{bmatrix} \boldsymbol{\kappa}(t) + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} x(t) \\ \hat{y}(t) &= \begin{bmatrix} \sigma_{zm} & \sigma_{zu_1} & \sigma_{zu_2} \end{bmatrix} \boldsymbol{\kappa}(t) \end{aligned} \tag{D.3}$$

Thus, the input–output map, and hence the loss, is fully determined by 9 loss-visible overlaps. To derive the resulting overlap dynamics, we introduce a compact matrix formulation (alternative to that presented in the main text) that scales naturally to higher-rank networks. To this end, we collect the six parameter vectors into

$$\boldsymbol{\Theta} = \begin{bmatrix} \boldsymbol{m} & \boldsymbol{u}_1 & \boldsymbol{u}_2 & \boldsymbol{v}_1 & \boldsymbol{v}_2 & \boldsymbol{z} \end{bmatrix} \in \mathbb{R}^{N \times 6} \tag{D.4}$$

and define the Gram matrix $\boldsymbol{Q}$, which contains all 21 pairwise overlaps

$$\boldsymbol{Q} = \frac{1}{N}\boldsymbol{\Theta}^\top \boldsymbol{\Theta} = \begin{bmatrix} \|\boldsymbol{m}\|^2 & \sigma_{mu_1} & \sigma_{mu_2} & \sigma_{v_1 m} & \sigma_{v_2 m} & \sigma_{zm} \\ \sigma_{mu_1} & \|\boldsymbol{u}_1\|^2 & \sigma_{u_1 u_2} & \sigma_{v_1 u_1} & \sigma_{v_2 u_1} & \sigma_{zu_1} \\ \sigma_{mu_2} & \sigma_{u_1 u_2} & \|\boldsymbol{u}_2\|^2 & \sigma_{v_1 u_2} & \sigma_{v_2 u_2} & \sigma_{zu_2} \\ \sigma_{v_1 m} & \sigma_{v_1 u_1} & \sigma_{v_1 u_2} & \|\boldsymbol{v}_1\|^2 & \sigma_{v_1 v_2} & \sigma_{zv_1} \\ \sigma_{v_2 m} & \sigma_{v_2 u_1} & \sigma_{v_2 u_2} & \sigma_{v_1 v_2} & \|\boldsymbol{v}_2\|^2 & \sigma_{zv_2} \\ \sigma_{zm} & \sigma_{zu_1} & \sigma_{zu_2} & \sigma_{zv_1} & \sigma_{zv_2} & \|\boldsymbol{z}\|^2 \end{bmatrix} \tag{D.5}$$

Note that $\boldsymbol{Q}$ is distinct from the preconditioning Gram matrix $\boldsymbol{G} = \boldsymbol{D}\boldsymbol{D}^\top$ introduced in the main text. The nine cross-block entries of $\boldsymbol{Q}$ are loss-visible, whereas the remaining 12 entries are loss-invisible. Let $\nabla_{ab} = \partial\mathcal{L}/\partial\sigma_{ab}$, and define the symmetric gradient matrix

$$\boldsymbol{M} = \begin{bmatrix} 0 & 0 & 0 & \nabla_{v_1 m} & \nabla_{v_2 m} & \nabla_{zm} \\ 0 & 0 & 0 & \nabla_{v_1 u_1} & \nabla_{v_2 u_1} & \nabla_{zu_1} \\ 0 & 0 & 0 & \nabla_{v_1 u_2} & \nabla_{v_2 u_2} & \nabla_{zu_2} \\ \nabla_{v_1 m} & \nabla_{v_1 u_1} & \nabla_{v_1 u_2} & 0 & 0 & 0 \\ \nabla_{v_2 m} & \nabla_{v_2 u_1} & \nabla_{v_2 u_2} & 0 & 0 & 0 \\ \nabla_{zm} & \nabla_{zu_1} & \nabla_{zu_2} & 0 & 0 & 0 \end{bmatrix} \tag{D.6}$$

Because the loss depends only on the visible overlaps, gradient flow in parameter space is given by

$$\dot{\boldsymbol{\Theta}} = -\frac{1}{N}\boldsymbol{\Theta}\boldsymbol{M} \tag{D.7}$$

Using $\boldsymbol{Q} = \boldsymbol{\Theta}^\top\boldsymbol{\Theta}/N$ and the product rule, we obtain

$$\dot{\boldsymbol{Q}} = \frac{1}{N}\left(\dot{\boldsymbol{\Theta}}^\top\boldsymbol{\Theta} + \boldsymbol{\Theta}^\top\dot{\boldsymbol{\Theta}}\right) = -\frac{1}{N}\left(\boldsymbol{M}\boldsymbol{Q} + \boldsymbol{Q}\boldsymbol{M}\right) \tag{D.8}$$

This single matrix equation defines a closed 21-dimensional system that fully characterizes learning in the rank-2 linear RNN and makes explicit how the loss-invisible entries of $\boldsymbol{Q}$ shape the evolution

of its loss-visible entries. To validate this derivation, we train the rank-2 RNN to emulate the response of a second-order filter (damped sinusoidal [25]):

$$y^\star(t) = e^{-c^\star t}\cos(\omega^\star t) \tag{D.9}$$

This target filter represents an oscillatory dynamics with frequency $\omega^\star$ and damping rate $c^\star$ (set to 2 and 0.3, respectively). Aside from initializing the RNN with rank-2 connectivity, all simulation details remain identical to those used in the rank-1 analysis App. A.4.1. Numerically training a high-dimensional RNN on this task, we find that both the loss and overlap trajectories align perfectly with the predictions of our scalar ODE system. This demonstrates the generality of our derivation beyond the rank-1 case (Fig. 13).

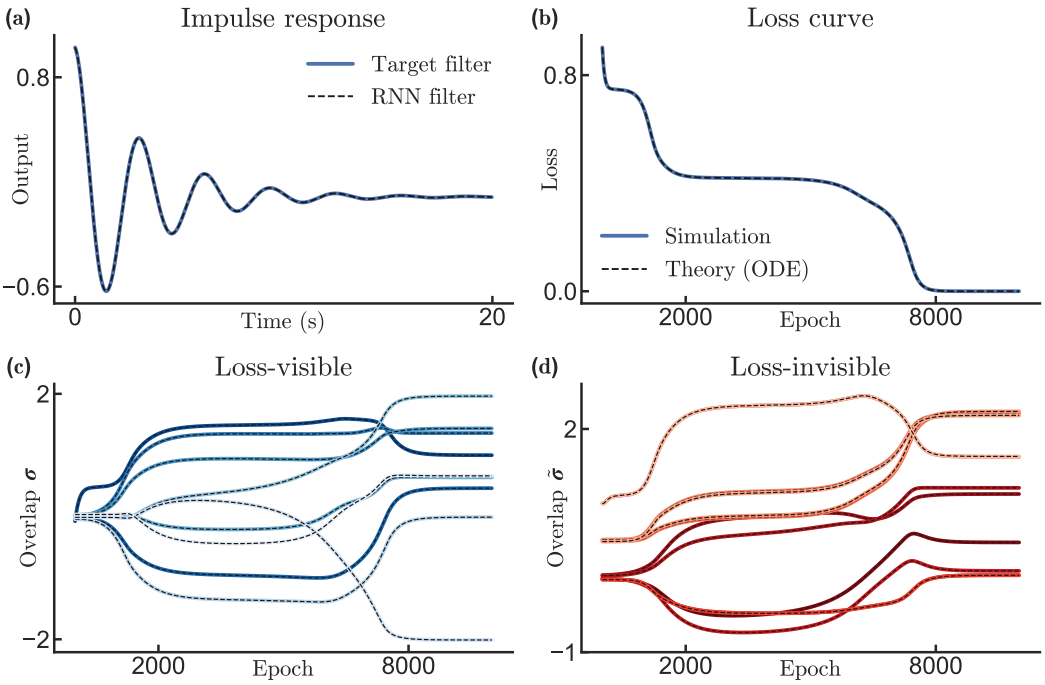

Figure 13: **(a)** Impulse response of the target damped-oscillatory filter (solid blue) and the final learned RNN response (dashed black). **(b)** Training loss for the full high-dimensional simulation (solid blue) and the overlap-based ODE theory (dashed black). **(c)** Dynamics of the 9 loss-visible and **(d)** 12 invisible overlaps, comparing numerical simulations (solid) with theoretical predictions (dashed).

Note For a general rank-r architecture with multiple inputs $\boldsymbol{m}_{\text{in}}$ and outputs $\boldsymbol{z}_{\text{out}}$, the derivation follows the same logic: define the full set of pairwise overlaps and apply the chain rule to obtain their induced dynamics. In this general case, the total number of overlaps—both loss-visible and loss-invisible—scales as $\mathcal{O}((2r + \boldsymbol{m}_{\text{in}} + \boldsymbol{z}_{\text{out}})^2)$. While this expression grows quadratically with the rank and the number of input/output channels, it remains strictly independent of the network size N. In this sense, the resulting dynamics remain tractable.

1. **Claims**

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

   Answer: [Yes]

   Justification: Our main contribution is the derivation of a mathematical framework to track the learning dynamics of low-rank RNNs. These results are developed analytically and validated through numerical simulations for both linear and nonlinear models. We then show how key phenomena such as degeneracy and memory can be studied using our mathematical framework, supported by exact equations and/or numerical simulations.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The limitations of our work are explicitly addressed in the Discussion section 7.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: We highlight key theoretical assumptions in the main text and provide complete and thorough mathematical derivations of our framework for both linear and nonlinear models in Appendices A and B, respectively.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: All implementation and simulation details, including task specifications, RNN parameters, and hyperparameters, are provided in the main text or the appendix.

Guidelines:

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

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

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

   Answer: [Yes]

   Justification: The main contribution is theoretical, with full derivations and simulation details provided in the main text and/or appendix to enable reproduction. Python code for both linear and nonlinear models is provided in the supplementary material.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: All details are either provided in the main text or the appendix.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: In Sec. 5, we report two statistical analyses. First, Fig. 4c verifies the Gaussian assumption via empirical histograms and Q–Q plots (see also App. B.4.1 for detailed discussion and breakdown cases). Second, in Figs. 4e,f, decoding accuracy is evaluated over 50 random train–test splits, with means and ± 1 s.d. error bars. Results show a substantial gap between loss-invisible and loss-visible overlaps that far exceeds the corresponding error bars. All additional experiments serve to verify the precise mathematical derivations.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: All simulations are lightweight and can be run on an off-the-shelf laptop. No specialized hardware or extensive computational resources are required.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: All authors have reviewed the NeurIPS Code of Ethics and confirm that the work complies with its principles.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [N/A]

Justification: This is a theoretical study that does not involve the development or deployment of new technology. As such, we do not anticipate direct societal impacts, either positive or negative.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

Answer: [N/A]

Justification: This is a theoretical study and does not involve the release of data or models with potential for misuse. Therefore, safeguards are not applicable in this context.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [N/A]

Justification: Prior work is appropriately cited where relevant.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [N/A]

    Justification: This is a theory-focused paper. No new assets are introduced.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: This is a theory-focused paper. No human subjects were involved, and no data were collected.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: This is a theory-focused paper. No human subjects were involved, and no data were collected.

Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: LLM is used only for writing and editing purposes.

    Guidelines:

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