Supplementary Material

A Low tensor rank recurrent neural networks

A.1 Architecture

Low tensor rank weights. In order to probe for the tensor rank of learning in neural data, we introduce an RNN architecture that captures the evolution of neural activity over slow timescales. We first recall the description of the architecture of the main text, $\mathbf{W} \in \mathbb{R}^{n \times n \times K}$,

$$\mathbf{W} = \sum_{j=1}^{R} \mathbf{a}_j \otimes \mathbf{b}_j \otimes \mathbf{c}_j.$$

In particular, at trial k, the dynamics of the RNN can be described as,

$$\tau \dot{\mathbf{x}} = W^{(k)} \phi(\mathbf{x}) - \mathbf{x} + B\mathbf{u}^{(k)}(t) = \sum_{j=1}^{R} c_j^{(k)} (\mathbf{a}_j \otimes \mathbf{b}_j) \phi(\mathbf{x}) - \mathbf{x} + B\mathbf{u}^{(k)}(t)$$

for $B \in \mathbb{R}^{n \times m}$, $\mathbf{u}^{(k)}(t) \in \mathbb{R}^m$, so that the RNN is a low rank RNN [14]. When training, we initialize $\mathbf{a}_j \sim \mathcal{N}(\mathbf{0}, I)$, $\mathbf{b}_j = \mathbf{a}_j$. Furthermore, the weights are parameterized such that $||\mathbf{a}_j|| = ||\mathbf{b}_j|| = 1$, so that the magnitude of a component is captured by $\mathbf{c}_j$.

Smoothness constraint over trials. We further constrain the initial covariance in trial of the trial factors by parameterizing them as $\mathbf{c}_j = (L + \sigma I)\bar{\mathbf{c}}_j$ where $LL^T = A$ is the Cholesky decomposition of the smooth covariance matrix A, and $\bar{\mathbf{c}}_j$ is initialized as $\bar{\mathbf{c}}_j \sim \mathcal{N}(\mathbf{0}, I)$. In particular, we use a rational quadratic kernel $s^2(1+(2l)^{-1}(k_i - k_j)^2)^{-1}$, where k_i is the ith trial index. This is equivalent to performing a 3-mode matrix-tensor product on the weight tensor itself $(L + \sigma I) \times_3 \mathbf{W}$ so that its entries over trials are linear combinations of smooth functions up to observation noise.

This parameterization is similar to that of Gaussian process regression, except no probabilistic objective is set. In particular, given that $(L + \sigma I)$ is invertible, any possible $\mathbf{c}_j$ can in theory be obtained upon optimization of the $\bar{\mathbf{c}}_j$. By additionally setting a regularization on $\bar{\mathbf{c}}_j$, we penalize the smoothness of $\mathbf{c}_j$ as non-smooth solutions have diverging $\bar{\mathbf{c}}_j$. In this way, we bias the optimization process towards smoother $\mathbf{c}_j$'s. As illustrated in Fig. 3b, the cross-validated loss remains similar to the non-smooth, full-rank, case.

A key advantage of having smooth trial factors is that missing trials can be easily be accounted for. Indeed, in most large-scale neural datasets, such as the one explored in the present work, potentially many trials may have been discarded, for example due to behavioural performance being outside the range set by the experimentalist. The assumption being made here is that the across-trial covariance is preserved when such trials occur. That is, we assume that a failure of the animal to perform a given trial does not imply that a trial wasn't informative, or that learning did not occur.

Condition-wise inputs. We parameterize the condition-wise inputs to the network with neural ordinary differential equations [37], i.e. as a dynamical system whose right hand side is parameterized by a deep neural network (DNN)

$$\dot{\mathbf{v}}^{(i)} = DNN(\mathbf{v}^{(i)}) \quad \mathbf{u}^{(i)}(t) = \phi(D\mathbf{v}^{(i)}(t)),$$

where $\mathbf{v}^{(i)}(t) \in \mathbb{R}^l$ and $D \in \mathbb{R}^{m \times l}$. Throughout this work, we used a fully-connected 3-layer DNN with layers of size 150 and ReLU nonlinearities. This provides inputs whose dynamics are considerably less constrained than those generated by low-rank RNNs. Thus, we do not make any assumption on the activity of upstream brain regions which drive the activity of the brain region being recorded. While here we chose to model the inputs using autonomous neural ODEs, one might imagine feeding the neural ODE with behavioural or task covariates to relax the condition specificity assumption, or fitting residual inputs to capture trial-by-trial variability arising from unmeasured variables [20]. This could account for some of the variability that can neither be explained by the task condition, nor by changes in the dynamics due to learning.

Loss. In sections 4 and 5 we focus on optimizing for the mean squared error (MSE)

$$L(\mathbf{a}, \mathbf{b}, \mathbf{c}, B, M) = \sum_{k}^{K} \sum_{q}^{T} \left|\left| M\phi(\mathbf{x}^{(k)}(t_q)) - \mathbf{y}^{(k)}(t_q) \right|\right|^2, \tag{A.1.1}$$

where $\mathbf{y}$ are firing rate estimates data and $M \in \mathbb{R}^{n \times N}$. In section B we present supplementary results on optimizing the Poisson log-likelihood with respect to spike data,

$$L(\mathbf{a}, \mathbf{b}, \mathbf{c}, B, M) = -\sum_{k}^{K} \sum_{q}^{T} \log(\text{Poisson}(\mathbf{y}^{(k)}(t_q) | M\phi(\mathbf{x}^{(k)}(t_q)))) \tag{A.1.2}$$

where $\mathbf{y}$ are binned spike data.

Code availability. A package implemented in Pytorch and TorchSDE [39] for applying ltrRNN to neural data will be released upon publication.

Table 1: **Hyperparameters of the ltrRNN models**. Bold indicates values specific to section 4 and 5, * indicates cross-validated hyperparameters, other hyperparameters were tuned by hand.

	Neural data (S4)	**Simulated data (S5)**
LtrRNN		
R	5^*	5^*
n	200^*	200^*
ϕ	tanh	tanh
Smoothness		
l	**50**	**15**
s	0.1	0.1
σ	0.1	0.1
Neural ODE		
Layers	$\mathbf{150 \times 150 \times 150}$	**N/A**
ϕ	**ReLU**	**N/A**
Regularization		
α	**0.01**	**0**
Cross-validation		
Train blocks	$1 \times 10 \times \mathbf{20}$	$1 \times 10 \times \mathbf{10}$
Test blocks	$1 \times 5 \times \mathbf{10}$	$1 \times 5 \times \mathbf{5}$
		neuron $\times$ time $\times$ trial

Table 2: **Approximate training time of ltrRNNs**. Here on the neural data of section 4. The variables which impact the most training time are the trial and neuron dimensions of the ltrRNN (not the rank or data time steps), as well as the neural ODE architecture. Hardware : desktop with an RTX 3090 Nvidia GPU and i7-12700K Intel CPU. * indicates the one used in section 4.

		Neurons	
		200	**400**
Trials	**370**	12min	22min
	740	16min*	40min

A.2 The dynamics of ltrRNNs

Rich changes of dynamics through oblique columns. Unlike for a matrix rank decomposition, a tensor rank decomposition can be unique even for non-orthogonal factors. A sufficient condition for uniqueness is that $r_{\mathbf{a}} + r_{\mathbf{b}} + r_{\mathbf{c}} \leq R - 2$ where, without loss of generality, $r_{\mathbf{a}}$ denotes the maximum number of linearly dependent columns of $\mathbf{a}$ [49]. In other words, fitting the changes of dynamics over trials as opposed to a single low rank RNN shared over all trials gives additional information regarding the columns and rows. In the case where the $\mathbf{a}_j$'s are not orthogonal, non-trivial qualitative changes in the vector fields can occur. Since for any given trial k, an ltrRNN is simply a low rank RNN [14],

$$\dot{\mathbf{x}}^{(k)} = \sum_{j} \mathbf{a}_j (c_j^{(k)} \mathbf{b}_j \cdot \phi(\mathbf{x}^{(k)})) + B\mathbf{u}^{(k)}(t) \tag{A.2.1}$$

so that the dynamics of $\mathbf{x}^{(k)}$ are constrained to $\text{span}\{\mathbf{a}_j\} \cup \{B_j\}$. Unlike low rank RNNs, ltrRNN are not necessarily invariant under changes of bases of $\mathbf{a}_j$'s. Nevertheless, we can introduce an

orthonormal basis $\{\tilde{\mathbf{a}}_j\}$ so that,

$$W^{(k)}\phi(\mathbf{x}^{(k)}) = \sum_i^R \tilde{\mathbf{a}}_i \sum_j (\tilde{\mathbf{a}}_i \cdot \mathbf{a}_j)(\mathbf{b}_j \cdot \phi(\mathbf{x}))c_j^{(k)} \tag{A.2.2}$$

In particular, notice that the dynamics along all $\tilde{\mathbf{a}}_i$ could potentially be affected by varying $c_j^{(k)}$.

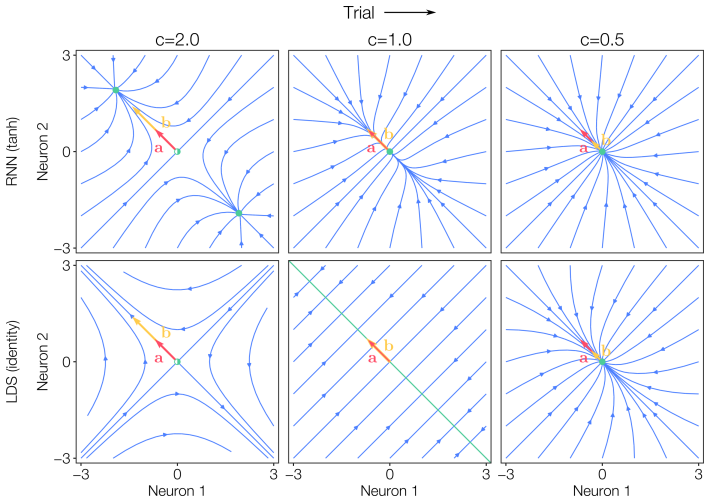
Conversely, the $\mathbf{a}_i$'s and $\mathbf{b}_i$'s being respectively orthonormal — e.g. as in a singular value decomposition — is not a sufficient condition for uniqueness of the tensor rank decomposition [50]. Nevertheless, constraining the $\mathbf{a}_i$ to be orthogonal and ensuring that the Kruskal constraint is satisfied, the vector fields are then orthogonal. In that case, varying $c_i^{(k)}$ for some i corresponds to rescaling the vector field along $\mathbf{a}_i$. Nevertheless, as is illustrated below with a single component, the leak term allows the system to display typical properties of linear and nonlinear dynamical systems.

Bifurcation in a tensor rank-one RNN. A classical example of bifurcation in a two-neuron system is that of the pitchfork supercritical bifurcation. Here, we show that it is essentially a tensor rank one RNN. We also illustrate how the corresponding linear RNN bifurcates. Let $\mathbf{a} = \mathbf{b} = [-1/\sqrt{2}, 1/\sqrt{2}]^T$ so that,

$$W^{(k)} = \frac{c}{2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \overset{\dot{\mathbf{x}}=0}{\Longrightarrow} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \frac{c}{2}\begin{bmatrix} \phi(\mathbf{x}_1) - \phi(\mathbf{x}_2) \\ -\phi(\mathbf{x}_1) + \phi(\mathbf{x}_2) \end{bmatrix} \tag{A.2.3}$$

That is $\mathbf{x}_1 = -\mathbf{x}_2$. We consider two cases, $\phi = \tanh$ and $\phi = \mathrm{id}$, both odd functions. Introducing the two in the previous equation, $-\phi(\mathbf{x}_2) = \phi(\mathbf{x}_1)$. So that $\mathbf{x}_i = c\phi(\mathbf{x}_i)$. Now considering each activation function separately,

- tanh: i) $c > 1$ has two solutions. ii) $c \leq 1$ one solution (the origin). At $c = 1$ the origin is a non-hyperbolic fixed point.
- id: i) $c = 1$ for any $\mathbf{x}_1$. The non-zero eigenvalue of the Jacobian of the system is negative, therefore it is a line attractor. ii) $\mathbf{x}_1 = 0$ for any c. Then the Jacobian of the system at the origin has both positive and negative eigenvalues for $c > 1$ and only negative for $c \leq 1$.



Supplementary Figure 1: **Bifurcation in a tensor rank one RNN**.

More general changes in vector fields. At the cost of increasing the rank, an ltrRNN can possibly transition between two arbitrary vector fields of ranks R_1 and R_2. For example, let $\mathbf{c}_j = [1, 0.9, ..., 0]$ for $j \in \{1, ..., R_1\}$ and $\mathbf{c}_j = [0, ..., 0.9, 1]$ for $j \in \{R_1 + 1, ..., R_1 + R_2\}$. There might however be multiple bifurcations between the first and last trial. More generally, given that any tensor has a (possibly high rank) tensor decomposition, any weight tensor can in theory be captured by an ltrRNN. This further illustrates the relevance of the result we found that ltrRNN of very low ranks fit data as well as full tensor RNNs.

A.3 Relationship between rank decomposition and eigendecomposition

In this section, we investigate the relationship between an arbitrary low rank decomposition of a matrix and the eigendecomposition. We assume that, on a given trial k, $W^{(k)}$ has the rank decomposition

$$W^{(k)} = \mathbf{a}\text{diag}(\mathbf{c}^{(k)})\mathbf{b}^T = \sum_{i=1}^{R'} c_i^{(k)} \mathbf{a}_i \mathbf{b}_i^T. \tag{A.3.1}$$

where $\mathbf{a} \in \mathbb{C}^{N \times R'}$, $\mathbf{b} \in \mathbb{C}^{R' \times N}$, $\mathbf{c}^{(k)} \in \mathbb{R}^{R'}$, and $\mathbf{a}_i, \mathbf{b}_i$ are the rows/colums of $\mathbf{a}, \mathbf{b}$ respectively. One such low rank decomposition is the eigendecomposition[2][3]

$$W^{(k)} = V\Lambda V^{-1} = V\text{diag}(\boldsymbol{\lambda})V^{-1} = \sum_{i=1}^{R} \lambda_i \mathbf{v}_i \tilde{\mathbf{v}}_i^T.$$

By the rank-nullity theorem, R is the rank of $W^{(k)}$, so that $R' \geq R$, with $R = R'$ when Equation (A.3.1) is a minimal rank decomposition. Equating the two decompositions gives a general expression for the eigenvalues:

$$\Lambda = V^{-1}\mathbf{a}\text{diag}(\mathbf{c})\mathbf{b}^T V \implies \lambda_i = (V^{-1}\mathbf{a}\text{diag}(\mathbf{c})\mathbf{b}^T V)_{ii} = \sum_j c_j (\tilde{\mathbf{v}}_i \cdot \mathbf{a}_j)(\mathbf{v}_i \cdot \mathbf{b}_j).$$

If $\lambda_i, \mathbf{a}, \mathbf{b}, \mathbf{c}^{(k)} \in \mathbb{R}$, this gives rise to the result stated in the main text:

$$\lambda_i = \sum_j c_j (\|\tilde{\mathbf{v}}_i\| \|\mathbf{a}_j\| \cos\theta_{ij}^{\tilde{\mathbf{v}},\mathbf{a}})(\|\mathbf{v}_i\| \|\mathbf{b}_j\| \cos\theta_{ij}^{\mathbf{v},\mathbf{b}}) \tag{A.3.2}$$

$$= \sum_j c_j \left(\frac{\|\mathbf{a}_j\|}{\cos(\theta_{ii}^{\tilde{\mathbf{v}},\mathbf{v}})} \cos\theta_{ij}^{\tilde{\mathbf{v}},\mathbf{a}}\right)(\|\mathbf{b}_j\| \cos\theta_{ij}^{\mathbf{v},\mathbf{b}}) \tag{A.3.3}$$

$$= \sum_j c_j \frac{\|\mathbf{a}_j\| \|\mathbf{b}_j\|}{\cos(\theta_{ii}^{\tilde{\mathbf{v}},\mathbf{v}})} \cos\theta_{ij}^{\tilde{\mathbf{v}},\mathbf{a}} \cos\theta_{ij}^{\mathbf{v},\mathbf{b}} \tag{A.3.4}$$

$$= \frac{1}{2} \sum_j c_j \frac{\|\mathbf{a}_j\| \|\mathbf{b}_j\|}{\cos(\theta_{ii}^{\tilde{\mathbf{v}},\mathbf{v}})} (\cos(\theta_{ij}^{\tilde{\mathbf{v}},\mathbf{a}} + \theta_{ij}^{\mathbf{v},\mathbf{b}}) + \cos(\theta_{ij}^{\tilde{\mathbf{v}},\mathbf{a}} - \theta_{ij}^{\mathbf{v},\mathbf{b}})). \tag{A.3.5}$$

Note that the above derivation makes no assumptions about the form of the low rank decomposition, other than that it is real-valued. Low rank decompositions commonly set $\|\mathbf{a}_i\| = \|\mathbf{b}_i\| = 1$, and often enforce orthogonality on the $\mathbf{a}_i$ and/or $\mathbf{b}_i$, thereby introducing additional constraints on the relationship between the c's and λ's. Normal matrices have $\cos(\theta_{ij}^{\tilde{\mathbf{v}},\mathbf{v}}) = \delta_{ij}$ and $\tilde{\mathbf{v}}_i = \mathbf{v}_i$, in which case further simplifications can be made.

B Motor learning

Pre-processing. Motor ($n = 72$ for Fig. 3, $n = 70$ for Sup. Fig. 4) and premotor ($n = 231$ for Fig. 3, $n = 137$ for Sup. Fig. 4) cortical neurons were used. The data was Gaussian filtered with a standard deviation of 40 ms (4 time bins). It was then centered by its baseline activity through subtracting neuron-wise the mean activity from around target onset to go-cue, and rescaled by dividing neuron-wise by the standard deviation of the execution period. Namely, the activity of a neuron $\bar{\mathbf{y}}_i(t)$ was given by

$$\bar{\mathbf{y}}_i(t) = \frac{\mathbf{y}_i(t) - \langle \mathbf{y}_i(t) \rangle_{t \leq 100}}{\langle (\mathbf{y}_i(t) - \langle \mathbf{y}_i(t) \rangle_{t>100})^2 \rangle_{t>100}} \tag{B.0.1}$$

[2] Assuming $W^{(k)}$ is diagonalisable. A similar argument using the Jordan normal form holds for defective matrices.

[3] Here, we write the left eigenvectors (rows of V^{-1}, transposed into column vectors) as $\tilde{\mathbf{v}}_r$. We can assume without loss of generality that the right eigenvectors $\mathbf{v}_i$ are normalised to unit length, in which case the orthonormality of left and right eigenvectors gives $\tilde{\mathbf{v}}_i \cdot \mathbf{v}_j = \delta_{ij} = \|\tilde{\mathbf{v}}_i\| \|\mathbf{v}_j\| \cos(\theta_{ij}^{\tilde{\mathbf{v}},\mathbf{v}}) \implies \|\tilde{\mathbf{v}}_i\| = 1/\cos(\theta_{ii}^{\tilde{\mathbf{v}},\mathbf{v}})$, where $\theta_{ii}^{\tilde{\mathbf{v}},\mathbf{v}}$ is the angle between the ith left and right eigenvector.

where $t = 0$ is the go cue. Example of activity upon this pre-processing is given in Supp. Fig. 2.

Modeling assumptions. We assumed that motor and premotor cortex was driven into an initial state by inputs from upstream regions during the preparatory period, after which the input shuts off so that the resulting activity during the reach evolves autonomously via the recurrent dynamics dynamics from that initial state. We therefore set $\mathbf{u}(t) = 0$ for $t > 100$ where $t = 0$ is the time of the go cue. Where the 100 ms account for a sensory delay.

Cross-validation procedure. We cross-validated the optimal rank and number of neurons of the ltrRNN. Low matrix or tensor rank models can be cross-validated by holding out specific entries of the matrix or tensor for training, and then used for testing. However, neural data has temporal correlation, such that the entry of the time-by-trial-by-neuron data tensor T_{ijk} is strongly correlated with $T_{i-1,jk}$ and $T_{i+1,jk}$. For example, assuming the data are continuous, a simple average of these entries will give an optimal estimate T_{ijk} in the limit of small time bins. Thus, the test set can be trivially inferred from the train set. We validated this intuition by performing the same cross-validation as section 4 but with $1 \times 1 \times 1$ blocks, and found the test loss was similar as using the train loss over the whole dataset (Sup. Fig. 3).

To counter this effect, sets of contiguous entries $[T_{ijk}, ..., T_{i+n,jk}]$ can be held out of training, and the interior of these blocks $[T_{i+q,jk}, ..., T_{i+n-q,jk}]$ used for testing [29]. Here, given that we are interested in uncovering smooth changes in neural activity over slow timescales, we hold out n-by-m matrices $[[T_{ijk}, ..., T_{i+n,jk}], ..., [T_{i,j+m,k}, ..., T_{i+n,j+m,k}]]$ (Fig. 3b. inset).

As mentioned in Supplementary Material A, our method allows inferring the dynamics of held-out trials. We found that using the cross-validation procedure from [29] infers similar ranks as holding out entire trials (Sup. Fig. 3). We nevertheless applied this procedure for the sake of being able to compare different classes of models.

Poisson log likelihood. Another method for fitting firing-rate models to spike data is to use the negative Poisson log likelihood loss [11, 10]. We fitted a ltrRNN ($R = 5$, $n = 200$ as in the MSE loss case) with softplus activation using negative Poisson log likelihood loss. We found similar but overall noisier results (Supp. Fig. 5). This may be due to the presence of high firing rate neurons which are normalized by the preprocessing procedure but not likelihood fitting.

C Task-trained RNN model of motor learning

Task design. A trial is split into a preparatory period $t \in [0, T_{go})$ and execution period $[T_{go}, T_{end}]$. Here we set $T_{go} = 2$, $T_{end} = 4$. To model a ballistic reach, the RNN receives input the target information and a hold cue during the preparatory period. During the execution period it evolves autonomously.

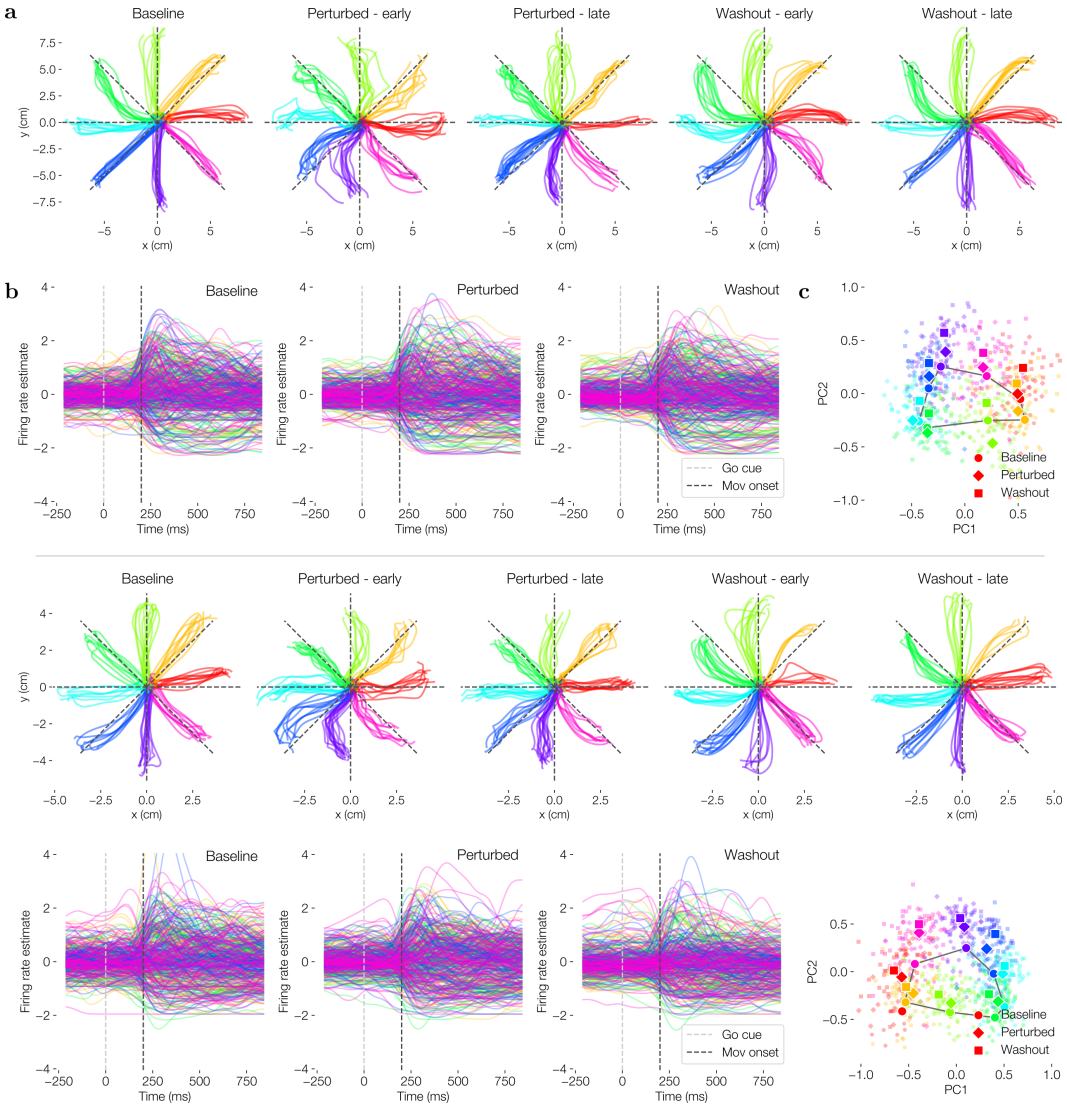
$$\mathbf{dx}^{(k)} = \left[W\phi(\mathbf{x}^{(k)}) - \mathbf{x}^{(k)} + \mathbb{1}_{t<T_{go}} B_{target}\mathbf{u}^{(k)}_{target} + \mathbb{1}_{t<T_{go}} B_{hold}\mathbf{u}^{(k)}_{hold}\right] dt + \sigma \mathbf{dW} \quad \text{(C.0.1)}$$

where $B_{target} \in \mathbb{R}^{n\times 2}$, $\mathbf{u}^{(k)}_{target} = [\cos(\theta^{(k)}), \sin(\theta^{(k)})]$ is a static vector representing the position of the target, $B_{hold} \in \mathbb{R}^{n\times 1}$, $\mathbf{u}^{(k)}_{hold} = 1$ a cue indicating to hold movement, and $\mathbf{dW}$ the infinitesimal increments of a Wiener process [51]. The dynamics of the hand are given in section 5 of the main text. The loss is taken to minimize the distance between the hand $\mathbf{y}^{(k)}(t)$ and the target $\mathbf{v}^{(k)}$ throughout the execution period, while keeping the hand still during the preparatory period, that is $L(W, B_{target}, B_{hold}, D) =$

$$\frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{T_{go}}\int_0^{T_{go}} ||\mathbf{y}^{(k)}(t)||^2 dt + \frac{1}{T_{end}-T_{go}}\int_{T_{go}}^{T_{end}} ||\mathbf{y}^{(k)}(t) - \mathbf{v}^{(k)}||^2 dt\right). \quad \text{(C.0.2)}$$

In particular, the speed of the reach is only constrained by the noise of the RNN and the hand. The dynamical system as a whole is evaluated with a differentiable adaptive step SDE solver [39] and trained with ADAM [52] during initial training, and SGD during motor perturbation learning.

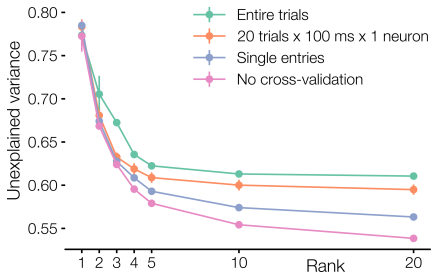
Analysis of the weights. We found that, consistent with the literature [16], the changes in weights resulting from the initial training were much larger than those of motor perturbation learning. PARAFAC on the full tensor of updates $\mathbf{W} - W_0 \otimes \mathbf{1}$ captured the weight tensor in 3 components (not shown), whose columns and rows were essentially those of performing SVD on W^*. Fitting



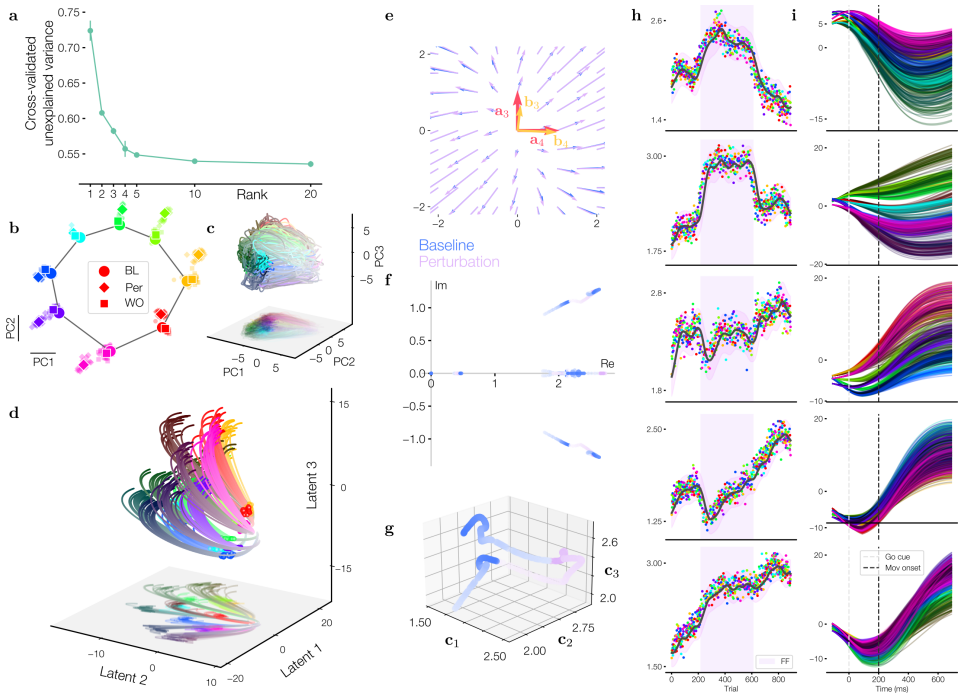
Supplementary Figure 2: **Persistent effects of motor learning**. Top: session used in the main text (Fig. 3). Bottom : session used in supplementary material (Sup. Fig. 4). **a**. Hand movement during the first and last 80 trials of perturbation learning and washout. The hand trajectories of some reach directions do not revert back post-washout (e.g. light green for top; dark green for bottom) **b**. Single neuron activity averaged within each condition. **c**. State at go cue $+100$ms. Larger full color marker are median within a condition. For some reach directions, the washout tends to be more similar to the perturbed state than the baseline.

additional PARAFAC components revealed that residual variability in the updates of pretraining was larger than the motor perturbation learning variability. Furthermore, unlike SVD, there is no guarantee that the components of fitting a rank $k+1$ PARAFAC model will be related to those of fitting a rank k model. Nevertheless, motor perturbation learning had a significant change on the eigenvalues and activity of the RNN (Fig. 5f,g)

To uncover an upper bound on rank of the weight tensor, we split the analysis into the pre-training and motor perturbation learning. We first performed SVD on the change in weights matrix $W^* - W_0$, where W^* are the weights of the network post-training, but pre-motor perturbation learning (Fig. 4b.). We found that the changes in weights were well captured by a rank-3 decomposition. Then, we performed PARAFAC on the change in weights tensor $\mathbf{W} - W^* \otimes \mathbf{1}$ of the motor perturbation learning. We found that this change of weight tensor was well approximated by a tensor rank



Supplementary Figure 3: **Comparison of cross-validation procedures**. Applied to the neural data of the session used in the main text.



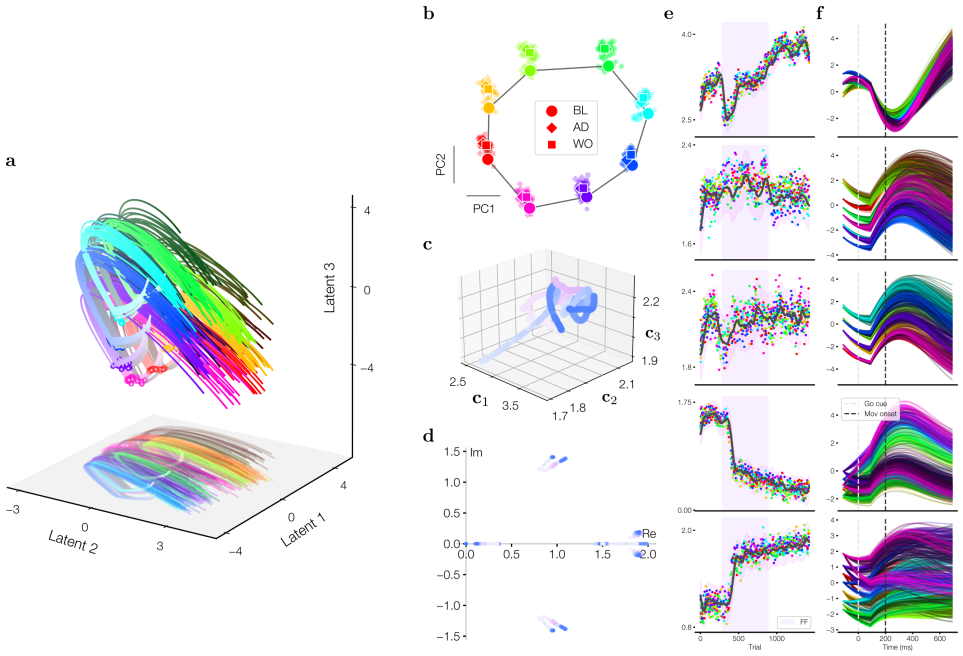
Supplementary Figure 4: **LtrRNN applied to an additional recording session**. **a**. Cross-validated loss with held out blocks of size 100ms by 20 trials. **b**. PCA on preprocessed data. **c**. State of the ltrRNN at go cue $+100$ms. **f**. Projection on first three $\mathbf{a}_j$. **e**. Projection of the vector field along $\mathbf{a}_j$ (see main text). **f**. Eigenspectrum of $W^{(k)}$ over trials. **g**. First three $\mathbf{c}_j$. **h**. Trial factors $\mathbf{c}_j$. **i**. Projection of $\mathbf{x}^{(k)}(t)$ on the corresponding $\mathbf{a}_j$.

2 decomposition. The combination of these results upper-bounds the tensor rank of the overall changes in weights to 5. Finally, we compared the subspace spanned by the columns of the SVD and PARAFAC decomposition by projecting the weight tensor $\mathbf{W} - W^* \otimes \mathbf{1}$ on the first three column and row singular vectors of $W^* - W_0$ and found approximately a remaining 0.2 unexplained variance, suggesting that the columns of $\mathbf{W}_0 - W^*$ and W^* were not orthogonal, but did not span the same subspace. Combined, these results suggest that the numerical tensor rank is at least 4 and at most 5, consistent with the results uncovered by ltrRNN from the RNN activity (Fig. 4e).

D Mathematical results

D.1 Adjoint derivation

In this section, we present a derivation of the adjoint mainly following [37]. Then, we derive the adjoint of recurrent neural networks.



Supplementary Figure 5: **Poisson log-likelihood fitting**. **a**. Projection on first three $\mathbf{a}_j$. **b**. State of the ltrRNN at go cue $+100$ms. **c**. First three $\mathbf{c}_j$. **d**. Eigenspectrum of $W^{(k)}$ over trials. **e**. Trial factors $\mathbf{c}_j$. **f**. Projection of $\mathbf{x}^{(k)}(t)$ on the corresponding $\mathbf{a}_j$.

D.1.1 State adjoint

Consider the dynamical system $\dot{\mathbf{x}} = f(\mathbf{x}, \boldsymbol{\theta}) \in \mathbb{R}^n$ where $\boldsymbol{\theta} \in \mathbb{R}^k$ is a set of parameters. Furthermore, let the functional $L : \mathbb{R}^n \to \mathbb{R}$ such that $L(\mathbf{x}(T))$ is our loss. First, define the *state adjoint*,

$$\mathbf{a}(t) = \frac{dL(\mathbf{x}(T))}{d\mathbf{x}(t)}. \tag{D.1.1}$$

Notice that since $\mathbf{x}(t + \Delta t)$ is a function of $\mathbf{x}(t)$,

$$\mathbf{a}(t) = \frac{dL(\mathbf{x}(T))}{d\mathbf{x}(t + \Delta t)} \frac{d\mathbf{x}(t + \Delta t)}{d\mathbf{x}(t)} = \mathbf{a}(t + \Delta t) \frac{d\mathbf{x}(t + \Delta t)}{d\mathbf{x}(t)}. \tag{D.1.2}$$

By Taylor expanding $\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t f(\mathbf{x}(t)) + O(\Delta t^2)$, we get,

$$\mathbf{a}(t) = \mathbf{a}(t + \Delta t)(I + \frac{d}{d\mathbf{x}(t)} f(\mathbf{x}(t)) + O(\Delta t^2)) \tag{D.1.3}$$

where I is the identity matrix. By rearranging,

$$\frac{\mathbf{a}(t + \Delta t) - \mathbf{a}(t)}{\Delta t} = \mathbf{a}(t + \Delta t) \frac{df(\mathbf{x}(t))}{d\mathbf{x}(t)} + O(\Delta t). \tag{D.1.4}$$

Taking the limit as $\Delta t \to 0$,

$$\frac{d\mathbf{a}(t)}{dt} = \mathbf{a}(t) \frac{df(\mathbf{x}(t))}{d\mathbf{x}(t)}. \tag{D.1.5}$$

We now have the dynamics of the adjoint; all that remains is that we find an initial (or rather terminal) condition. For this, notice that

$$\mathbf{a}(T) = \frac{dL(\mathbf{x}(T))}{d\mathbf{x}(T)} \tag{D.1.6}$$

is the usual gradient of L w.r.t. to its argument.

D.1.2 Parameter adjoint and gradient

In the above we derived the state adjoint $\mathbf{a}$. However, for our purposes we also require the *parameter adjoint* $dL(\mathbf{x}(T))/d\boldsymbol{\theta}$. For this, it suffices to augment the original dynamical system with its parameters[4] $\dot{\mathbf{z}} = [f(\mathbf{x}, \boldsymbol{\theta}), \mathbf{0}]$ and initial (later terminal) condition $\mathbf{z}(0) = [\mathbf{x}(0), \boldsymbol{\theta}]$. Defining the loss $\bar{L}(\mathbf{z}(T)) = L(\mathbf{x}(T))$, the adjoint of this augmented system is,

$$\mathbf{a}_{\mathbf{z}}(t) = \frac{d\bar{L}(\mathbf{z}(T))}{d\mathbf{z}(t)} = \left[\frac{d\bar{L}(\mathbf{z}(T))}{d\mathbf{x}(t)}, \frac{d\bar{L}(\mathbf{z}(T))}{d\boldsymbol{\theta}}\right] = \left[\frac{dL(\mathbf{x}(T))}{d\mathbf{x}(t)}, \frac{dL(\mathbf{x}(T))}{d\boldsymbol{\theta}}\right] = \left[\mathbf{a}_{\mathbf{x}}(t), \frac{dL(\mathbf{x}(T))}{d\boldsymbol{\theta}}\right], \tag{D.1.7}$$

which contains the desired term $dL(\mathbf{x}(T))/d\boldsymbol{\theta}$. It now remains to describe the dynamics of the augmented system. By the same argument as for the state adjoint (D.1.5),

$$\frac{d\mathbf{a}_{\mathbf{z}}(t)}{dt} = \mathbf{a}_{\mathbf{z}}(t)\frac{d\dot{\mathbf{z}}}{d\mathbf{z}}. \tag{D.1.8}$$

By (D.1.7) and by unconcatenating $\mathbf{z}$,

$$= \left[\mathbf{a}_{\mathbf{x}}(t), \frac{dL(\mathbf{x}(T))}{d\boldsymbol{\theta}}\right] \begin{bmatrix} \frac{df(\mathbf{x}(t))}{d\mathbf{x}(t)} & \frac{d\mathbf{x}(t)}{d\boldsymbol{\theta}} \\ \frac{d\mathbf{0}}{d\mathbf{x}(t)} & \frac{d\mathbf{0}}{d\boldsymbol{\theta}} \end{bmatrix} \tag{D.1.9}$$

$$= \left[\mathbf{a}(t)\frac{df(\mathbf{x}(t))}{\mathbf{x}(t)}, \mathbf{a}(t)\frac{df(\mathbf{x}(t))}{d\boldsymbol{\theta}}\right]. \tag{D.1.10}$$

Hence the following dynamical system can be evaluated,

$$\frac{d}{dt}\left[\mathbf{x}(t), \mathbf{a}(t), \frac{dL(\mathbf{x}(T))}{d\boldsymbol{\theta}}\right] = \left[f(\mathbf{x}(t), \boldsymbol{\theta}), \mathbf{a}(t)\frac{df(\mathbf{x}(t))}{\mathbf{x}(t)}, \mathbf{a}(t)\frac{df(\mathbf{x}(t))}{d\boldsymbol{\theta}}\right], \tag{D.1.11}$$

with terminal condition

$$\left[\mathbf{x}(T), \mathbf{a}(T), \frac{dL(\mathbf{x}(T))}{d\boldsymbol{\theta}}\right] = \left[\mathbf{x}(T), \frac{dL(\mathbf{x}(T))}{d\mathbf{x}(T)}, \mathbf{0}\right]. \tag{D.1.12}$$

Notice that to obtain the terminal condition, since it depends on $\mathbf{x}(T)$, the original dynamical system must be evaluated forward once.

D.1.3 RNN adjoint

Let $\dot{\mathbf{x}} = f(\mathbf{x}, W) = W\phi(\mathbf{x}) - \mathbf{x} + B\mathbf{u}(t)$ and $L(\mathbf{x}(T)) = ||D\phi(\mathbf{x}(T)) - \mathbf{y}||^2$ for $y \in \mathbb{R}^d$. Furthermore, as it will be convenient, let $W = \sum_i^R \boldsymbol{\alpha}_i \otimes \boldsymbol{\beta}_i$. We will derive one by one the terms needed to characterize the parameter adjoint. First the Jacobian is

$$\frac{df(\mathbf{x}(t), \boldsymbol{\theta})}{d\mathbf{x}(t)} = \sum_i^R \boldsymbol{\alpha}_i \otimes (\boldsymbol{\beta}_i \odot \phi'(\mathbf{x}(t))) - I \tag{D.1.13}$$

where $\odot$ denotes the element-wise product, I the identity matrix, and ϕ' the derivative of ϕ. Next,

$$\frac{df(\mathbf{x}(t), \boldsymbol{\theta})_i}{dW_{jk}} = \begin{cases} 0 & i \neq j \\ \phi(\mathbf{x})_j & i = j \end{cases}, \tag{D.1.14}$$

that is

$$\frac{df(\mathbf{x}(t), \boldsymbol{\theta})}{dW} = I \otimes \phi(\mathbf{x}). \tag{D.1.15}$$

Finally the terminal condition,

$$\frac{dL(\mathbf{x}(T))}{d\mathbf{x}(T)_i} = \frac{d}{dx_i}\sum_j^d (D_j \cdot \phi(\mathbf{x}) - y_j)^2 = \sum_j^d (D_j \cdot \phi(\mathbf{x}) - y_j)(D_{ij}\phi'(\mathbf{x})_i) \tag{D.1.16}$$

[4] Here $[\cdot, \cdot]$ denotes row concatenation.

that is,

$$\frac{dL(\mathbf{x}(T))}{d\mathbf{x}(T)} = \sum_j^d (D_j \cdot \phi(\mathbf{x}) - y_j)(D_j \odot \phi'(\mathbf{x})) = \phi'(\mathbf{x}) \odot \sum_j^d (D_j \cdot \phi(\mathbf{x}) - y_j) D_j \tag{D.1.17}$$

Or more explicitly, $\dot{\mathbf{a}}_\mathbf{z} =$

$$\begin{cases} \dot{\mathbf{a}}_\mathbf{x} = \left(\sum_i^R \boldsymbol{\alpha}_i \otimes \boldsymbol{\beta}_i \odot \phi'(\mathbf{x})\right)^T \mathbf{a}_\mathbf{x} - \mathbf{a}_\mathbf{x}, & \mathbf{a}_\mathbf{x}(T) = \phi'(\mathbf{x}(T)) \odot \sum_j^d (D_j \cdot \phi(\mathbf{x}(T)) - y_j) D_j \\ \dot{\mathbf{a}}_W = \mathbf{a}_\mathbf{x} \otimes \phi(\mathbf{x}), & \mathbf{a}_W(T) = 0 \end{cases} \tag{D.1.18}$$

D.2 Rank of the gradient

D.2.1 The gradient as a composition of operators

In this section, we prove Theorems 1. In order to derive bounds on the singular values of $\nabla_W L = \mathbf{a}_W(0)$, we shall now consider $\mathbf{a}_\mathbf{x}$ and $\phi(\mathbf{x})$ as linear operators with integration. Namely,

$$\mathbf{a}_\mathbf{x}\mathbf{y} := \int_0^T \mathbf{a}_\mathbf{x}(t) y(t) dt \tag{D.2.1}$$

where $\mathbf{y} \in \mathcal{H}$ for $\mathcal{H}$ some suitable Hilbert space, such as L^2 for our case.

More formally, let $\mathbf{a}_\mathbf{x}, \phi(\mathbf{x}) \in \mathcal{B}_{0,0}$, where $\mathcal{B}_{0,0}$ is the Banach space of i) compact ii) bounded operators from $\mathcal{H}$ to $\mathbb{R}^n$, such that $\mathbf{a}_\mathbf{x}, \phi(\mathbf{x}) : \mathcal{H} \to \mathbb{R}^n$. In particular, we note that compactness follows from the image of $\mathbf{a}_\mathbf{x}$, that is $\mathbf{a}_\mathbf{x}(\mathcal{H})$, to be a vector subspace of $\mathbb{R}^n$ and therefore be of finite rank. By the same argument, $\phi(\mathbf{x})$ is compact, and therefore so is $\phi(\mathbf{x})^*$ by Schauder's theorem [53], where * is adjunction. Furthermore, notice that $\phi(\mathbf{x}), \mathbf{a}_\mathbf{x}$ are solutions of dynamical systems with differentiable right hand side and therefore bounded (in $\mathbb{R}^n$) if they are evaluated for finite time, and therefore bounded when seen as operators. The following result can now be applied:

Lemma 2 ([53]). *Let $T \in \mathcal{B}_{0,0}$, then T admits a singular value decomposition. Furthermore, this singular value decomposition is of finite rank.*

We can now prove the main theorem.

Proof of Theorem 1. Notice that the composition of the two operators is: $\mathbf{a}_\mathbf{x} \circ \phi(\mathbf{x})^* = \nabla_W L$. Furthermore, the singular values of $\nabla_W L$ are,

$$\sigma_i^{\nabla_W L} = \min_{\substack{U \subseteq \mathbb{R}^n, \\ \dim U = \\ n-i-1}} \max_{\substack{\mathbf{y} \in U, \\ ||\mathbf{y}||=1}} ||\nabla_W L \mathbf{y}|| = \min_{\substack{U \subseteq \mathbb{R}^n, \\ \dim U = \\ n-i-1}} \max_{\substack{\mathbf{y} \in U, \\ ||\mathbf{y}||=1}} ||\mathbf{a}_\mathbf{x} \phi(\mathbf{x})^* \mathbf{y}|| \tag{D.2.2}$$

which can be bounded as,

$$\min_{\substack{U \subseteq \mathbb{R}^n, \\ \dim U = \\ n-i-1}} \max_{\substack{\mathbf{y} \in U, \\ ||\mathbf{y}||=1}} ||\mathbf{a}_\mathbf{x} \phi(\mathbf{x})^* \mathbf{y}|| \leq \min_{\substack{U \subseteq \mathbb{R}^n, \\ \dim U = \\ n-i-1}} \max_{\substack{\mathbf{y} \in U, \\ ||\mathbf{y}||=1}} ||\mathbf{a}_\mathbf{x}|| ||\phi(\mathbf{x})^* \mathbf{y}|| \tag{D.2.3}$$

$$= \sigma_1^{\mathbf{a}_\mathbf{x}} \min_{\substack{U \subseteq \mathbb{R}^n, \\ \dim U = \\ n-i-1}} \max_{\substack{\mathbf{y} \in U, \\ ||\mathbf{y}||=1}} ||\phi(\mathbf{x})^* \mathbf{y}|| \tag{D.2.4}$$

that is,

$$\tag{D.2.5}$$

$$\sigma_i^{\nabla_W L} \leq \sigma_1^{\mathbf{a}_\mathbf{x}} \sigma_i^{\phi(\mathbf{x})^*} \tag{D.2.6}$$

Now notice that, for any operator, akin to the matrix case, T_1, T_2, the adjoint of their composition is $(T_1T_2)^* = T_2^*T_1^*$. Furthermore,

$$\sigma_i^{(\nabla_W L)^T} = \min_{\substack{U\subseteq\mathbb{R}^n,\\ \dim U = \\ n-i-1}} \max_{\substack{\mathbf{y}\in U,\\ ||\mathbf{y}||=1}} ||(\mathbf{a_x}\phi(\mathbf{x})^*)^*\mathbf{y}|| = \min_{\substack{U\subseteq\mathbb{R}^n,\\ \dim U = \\ n-i-1}} \max_{\substack{\mathbf{y}\in U,\\ ||\mathbf{y}||=1}} ||\phi(\mathbf{x})\mathbf{a}_\mathbf{x}^*\mathbf{y}|| \tag{D.2.7}$$

$$\leq \sigma_1^{\phi(\mathbf{x})} \min_{\substack{U\subseteq\mathbb{R}^n,\\ \dim U = \\ n-i-1}} \max_{\substack{\mathbf{y}\in U,\\ ||\mathbf{y}||=1}} ||\mathbf{a}_\mathbf{x}^*\mathbf{y}|| \tag{D.2.8}$$

$$= \sigma_1^{\phi(\mathbf{x})}\sigma_i^{\mathbf{a}_\mathbf{x}^*} \tag{D.2.9}$$

Noticing that $\sigma_i^{\mathbf{a}_\mathbf{x}^*} = \sigma_i^{\mathbf{a}_\mathbf{x}}$ and $\sigma_i^{\nabla_W L} = \sigma_i^{(\nabla_W L)^T}$,

$$\sigma_i^{\nabla_W L} \leq \sigma_1^{\phi(\mathbf{x})}\sigma_i^{\mathbf{a}_\mathbf{x}} \tag{D.2.10}$$

Combining D.2.6 and D.2.10 we obtain the sought upper bound of Theorem 1,

$$\sigma_i^{\nabla_W L} \leq \min\left\{\sigma_1^{\mathbf{a}_\mathbf{x}}\sigma_i^{\phi(\mathbf{x})}, \sigma_1^{\phi(\mathbf{x})}\sigma_i^{\mathbf{a}_\mathbf{x}}\right\} \tag{D.2.11}$$

Similar steps can be used to derive the lower bound of Theorem 1, using instead the identity $\sigma_n^{T_1}||T_2\mathbf{y}|| \leq ||T_1T_2\mathbf{y}||$ where $T_1, T_2 \in \mathcal{B}_{0,0}$ and $\sigma_n^{T_1}$ is the smallest non-zero singular value of T_1. □

We however note that, unlike the upper bound, the lower bound we provide does not have any numerical use, as the smallest singular value of the adjoint or of the firing rate is practically 0 (for example, well bellow machine precision).

We further point out that a more explicit characterization of the singular values of the gradient can be obtained. Let $U_i^{\mathbf{a}_\mathbf{x}}, U_i^{\phi(\mathbf{x})}$ be the right singular vectors of respectively $\mathbf{a_x}$ and $\phi(\mathbf{x})$, so that $V_i^{\mathbf{a}_\mathbf{x}}(t)$, $V_i^{\phi(\mathbf{x})}(t)$ are the left singular vectors and $\sigma_i^{\mathbf{a}_\mathbf{x}}$, $\sigma_j^{\phi(\mathbf{x})}$ the singular values.

$$\nabla_W L = \sum_{i,j=1}^{n} \left(V_i^{\mathbf{a}_\mathbf{x}} \otimes V_j^{\phi(\mathbf{x})}\right) \sigma_i^{\mathbf{a}_\mathbf{x}}\sigma_j^{\phi(\mathbf{x})} \int_0^T U_i^{\mathbf{a}_\mathbf{x}}(t)U_j^{\phi(\mathbf{x})}(t)dt \tag{D.2.12}$$

The characterization of the rank of the gradient has thus shifted to the firing rate and adjoint spaces. Furthermore, the integral is just the inner product between their right singular vectors and therefore of magnitude bounded by 1. Hence, for the dynamics of the weights to be large in a given direction in weight space $V_i^{\mathbf{a}_\mathbf{x}} \otimes V_j^{\phi(\mathbf{x})}$, all three of: the singular values of the firing rate, the state adjoint, as well as their cofluctuation in time, must not be small.

D.2.2 Weight gradient for time discretizations

Finally, we mention that our detour through functional analysis was for the sake of mathematical rigour, and that in practical applications, the RNN and its adjoint are evaluated at discrete time steps $0, \Delta t..., q\Delta t$. In that case, the gradient can be estimated as a simple matrix-matrix product. Let $A = [\mathbf{a_x}(0), ..., \mathbf{a_x}(q\Delta t)]$ and $B = [\phi(\mathbf{x})(0), ..., \phi(\mathbf{x})(q\Delta t)]$. Then $\nabla_W L = \Delta t AB^T$, and the bounds D.2.6 and D.2.10 follow from classic matrix-matrix product bounds [54]. In particular, given that $\phi(\mathbf{x})$ and $\mathbf{a_x}$ are smooth, without loss of generality,

$$\sigma_i^{\nabla_W L} = \lim_{\min_j(t_{j+1}-t_j)\to 0} \sigma_i^{AB^T} \tag{D.2.13}$$

This simple matrix-matrix product opens up the possibility of fast RNN adjoint implementations as, unlike computing $\mathbf{a_x}\frac{df}{dW}$ in general adjoint solvers, which requires $O(qn^3)$ time and $O(n^2)$ memory complexity for an n-dimensional RNN and q time steps, here the time complexity drops to $O(qn^2)$.

D.2.3 Rank of linear RNNs

In this section we prove Theorem 2.

Proof of claim 1. If $W = \sum_i^R \boldsymbol{\alpha}_i \otimes \boldsymbol{\beta}_i$, then by Lemma 1, the dynamics of the adjoint are constrained to the span of the rows of W, namely, $\dot{\mathbf{a}}_\mathbf{x} \in \operatorname{span}\{\boldsymbol{\beta}_i\}$. Therefore, $\mathbf{a}_\mathbf{x} \in \operatorname{span}\{\boldsymbol{\beta}_i\} \cup \{\mathbf{a}_\mathbf{x}(T)\}$, which is a at most $R+1$ dimensional subspace. If $\mathbf{a}_\mathbf{x}$ is constrained to a at most $R+1$ dimensional subspace, then $\dot{\mathbf{a}}_W = \mathbf{a}_\mathbf{x} \otimes \mathbf{x}$ is also constrained to a at most $R+1$ dimensional subspace. Since $\mathbf{a}_W(T) = 0$, $\mathbf{a}_W$ is constrained to the same subspace as its dynamics, and in particular, $\operatorname{rank} \mathbf{a}_W(0) = \operatorname{rank} \nabla_W L^{(0)} \leq R+1$. □

Proof of claim 2. Suppose $W^{(k)} = \sum_i^{2R+m+d} \boldsymbol{\alpha}_i^{(k)} \otimes \boldsymbol{\beta}_j^{(k)}$, where $\boldsymbol{\alpha}_i^{(k)}, \boldsymbol{\beta}_i^{(k)} \in \operatorname{span}\{\boldsymbol{\alpha}_i^{(0)}\} \cup \{\boldsymbol{\beta}_i^{(0)}\} \cup \{B_i\} \cup \{D_i\} := V$. In particular, notice that V is only dependent on the initial weight. Then by a similar argument as above, $\mathbf{a}_\mathbf{x}^{(k)} \in V$, which implies $\mathbf{a}_W(0)^{(k)} = \nabla_W L^{(k)} \in V$. Therefore $W^{(k+1)} = W^{(k)} + \gamma \nabla_W L^{(k)} = \sum_i^{2R+m+d} \boldsymbol{\alpha}_i^{(k+1)} \otimes \boldsymbol{\beta}_j^{(k+1)}$ with $\boldsymbol{\alpha}_i^{(k+1)} \in V$. That is $\operatorname{rank} W^{(k+1)} \leq 2R+m+d$. □

Proof of claim 3. Mutatis mutandis $\boldsymbol{\beta}^{(k+1)} \in V$, that is $\mathbf{x} \in V$. Therefore, $\mathbf{W}^{(k)} = \sum_{ij}^{2R+m+d} c_{ij}^{(k)} \boldsymbol{\alpha}_i \otimes \boldsymbol{\alpha}_j$ for some $c_{ij}^{(k)}$'s, where $\boldsymbol{\alpha}_i \in V$. Or equivalently, $\mathbf{W} = \sum_{ij}^{2R+m+d} \boldsymbol{\alpha}_i \otimes \boldsymbol{\alpha}_j \otimes \mathbf{c}_{ij}$. That is, $\operatorname{rank} \mathbf{W} \leq (2R+m+d)^2$. □

D.3 Extensions of our results

Loss integrated over time. Commonly, the loss considered might be integrated,

$$\mathcal{L}(T) := \int_0^T L(\mathbf{x}(t), \mathbf{y}(t)) dt \tag{D.3.1}$$

The parameter adjoint is dependent only linearly on the state adjoint, we may therefore integrate the state adjoint for all initial conditions.

$$\mathcal{L}(T) = \left(\int_T^0 \mathbf{a}_\mathbf{x}(t) + \int_t^0 \dot{\mathbf{a}}_\mathbf{x}(t') dt' \right) dt \tag{D.3.2}$$

The term inside the first integral is just the solution of time-varying autonomous LDS, therefore,

$$\int_T^0 \mathbf{a}_\mathbf{x}(t) dt = \int_T^0 \Phi(0,t) \frac{dL(\mathbf{x}(t), \mathbf{y}(t))}{d\mathbf{x}(t)} dt \tag{D.3.3}$$

Where Φ is the linear dynamical system state transition matrix [55]. But notice that this is the solution to the controlled LDS,

$$\dot{\mathbf{a}}_\mathbf{x} = \mathbf{a}_\mathbf{x} \frac{df(\mathbf{x}(t))}{d\mathbf{x}(t)} + \frac{dL(\mathbf{x}(t), \mathbf{y}(t))}{d\mathbf{x}(t)}, \quad \dot{\mathbf{a}}_\mathbf{x}(T) = 0 \tag{D.3.4}$$

In the specific case of an RNN,

$$\dot{\mathbf{a}}_\mathbf{x} = \left(\sum_i^R \alpha_i \otimes \beta_i \odot \phi'(\mathbf{x}) \right)^T \mathbf{a}_\mathbf{x} - \mathbf{a}_\mathbf{x} + \phi'(\mathbf{x}(t)) \odot \sum_j^d (D_j \cdot \phi(\mathbf{x}(t)) - \mathbf{y}_j) D_j, \quad \dot{\mathbf{a}}_\mathbf{x}(T) = 0 \tag{D.3.5}$$

Therefore, Theorem 1 remains unchanged for a loss integrated over time. For Theorem 2, Claims 2-3 remain unchanged, while Claim 1 becomes $\operatorname{rank} \nabla_W L^{(0)} \leq R + \min\{d, m\}$.

Gradient with respect to other parameters. So far we have focused on the gradient of the weights of the RNN. As we have seen, the space over which the state adjoint $\mathbf{a}_\mathbf{x}(t)$ evolves as well as the trajectories of the system itself $\mathbf{x}(t)$ determine the space over which the gradient evolves. But those are respectively dependent on the decoder D and encoder B of the system. If all parameters D, B, W

of the system are optimized simultaneously, as is most often the case, we may wonder how our bounds hold.

First, for B, notice that by a similar argument as for W, $\frac{df(\mathbf{x},B\mathbf{u}(t))}{dB} = I \otimes \mathbf{u}(t)$, so that $\dot{\mathbf{a}}_B = \mathbf{a}_\mathbf{x} \otimes \mathbf{u}(t)$. Therefore, following essentially the same derivation as that of Theorem 1, the following bound can be derived,

$$\sigma_i^B \leq \min\{\sigma_1^{\mathbf{a}_\mathbf{x}}\sigma_i^{\mathbf{u}}, \sigma_1^{\mathbf{u}}\sigma_i^{\mathbf{a}_\mathbf{x}}\}. \tag{D.3.6}$$

By a similar argument as for the derivation of Theorem 2, $B_i^{(k)} \in \text{span}\{\alpha_i\} \cup \{\beta_i\} \cup \{B_i\} \cup \{D_i\}$.

Second, for D, since $\frac{df}{dD} = 0$, that is $\dot{\mathbf{a}}_D = 0$,

$$\mathbf{a}_D(T) = \mathbf{a}_D(0) = \nabla_D L = \frac{dL(T)}{dD}. \tag{D.3.7}$$

In other words, the gradient of the loss with respect to the decoder weights have zero dynamics.

The gradient of a functional w.r.t. a given parameter is independent of the gradient of that functional with respect to another parameter if these two parameters do not depend on one another. Therefore these results hold regardless of which combination of W, B or D is optimized.

Batched updates. Most often, the weights are updated in batches. That is $\Delta W^{(k)} = Q^{-1}\sum_q^Q \nabla_W L^{(k,q)}$ where q is the index over the batched dimension. Since the column and row spaces of $\nabla_W L^{(k,q)}$ remain unchanged, Theorem 2 2-3. remain unchanged, while 1. becomes $\text{rank}\,\Delta W^{(0)} \leq R + \min\{d, m\}$. For Theorem 1, the common singular value identities [54] $\sigma_{i+j-1}(A+B) \leq \sigma_i(A) + \sigma_j(B)$ and $\sigma_i(cA) = c\sigma_i(A)$ for $c \in \mathbb{R}_0^+$ can be used. Then, $\sigma_{\sum i_q - Q+1}(\Delta W^{(k)}) \leq Q^{-1}\sum_q^Q \sigma_{i_q}(\nabla_W L^{(k,q)})$.

Momentum-based optimizers. Momentum-based optimizers such as Adam [52] are commonly used to train RNNs on behavioural tasks. Here we focus on the first moment, a similar derivation can be undertaken for higher moments. In that case, a momentum variable is introduced, which is updated as $M^{(k+1)} = \beta M^{(k)} + (1-\beta)\nabla_W L^{(k)}$, where β determines the speed of the exponential decay. The weights are then updated as $W^{(k+1)} = W^{(k)} - \alpha M^{(k+1)}$, where α is the learning rate. Which implies, $\Delta W^{(k)} = W^{(k+1)} - W^{(k)} = -\alpha\sum_j^k (1-\beta)^j \nabla_W L^{(j)}$. Using the same identities are for batched updates, $\sigma_{\sum i_q - k+1}(\Delta W^{(k)}) \leq \sum_j^k (1-\beta)^j \sigma_{i_q}(\nabla_W L^{(j)})$.

D.4 Numerical simulations

Similarly to [13], we illustrate our mathematical results on random RNNs. Since constant inputs are one dimensional ($B\mathbf{u} = \sum B_i u_i$), we instead use time-varying inputs parameterized with LDS:

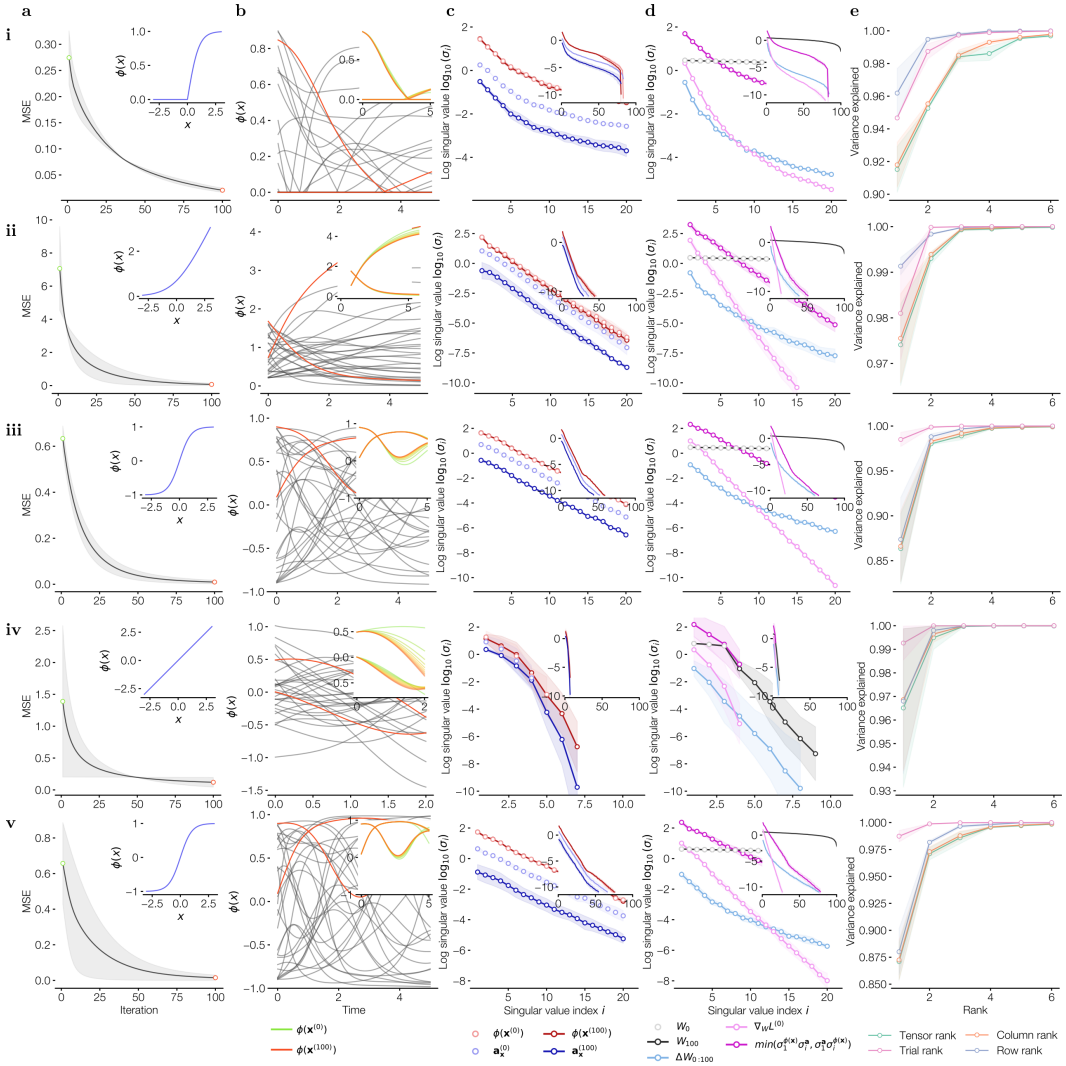
$$\dot{\mathbf{u}} = M\mathbf{u} - \mathbf{u} \quad \mathbf{u}(0) = \mathbf{u}_0 \tag{D.4.1}$$

where $\mathbf{u}(t) \in \mathbb{R}^m$, $M_{ij} \sim \mathcal{N}(0, 1\sqrt{m})$, $\mathbf{u}_0 \sim \mathcal{N}(0, \mathbf{1}/2)$. The target outputs are set as $\mathbf{y} \in \mathbb{R}^l$, $\mathbf{y}_i \sim \mathcal{U}(-1, 1)$. The loss is defined as,

$$L(W) = ||D\phi(\mathbf{x})(T) - \mathbf{y}||^2 \tag{D.4.2}$$

where $\mathbf{x}$ is the solution of an RNN as considered thus far.

In Sup. Fig. 6 we show the effect of varying ϕ, the rank R of the initial weights as well as the standard deviation (or strength) g of the initial weights such that $W_{ij}^{(0)} \sim \mathcal{N}(0, g^2)$.



Supplementary Figure 6: **Singular values of adjoints and gradients**. **a**. Loss over training. Inset: activation function. **b**. Activity during the last trial (black). Inset: activity of two example neurons over training. **c**. Singular values of the firing rate and adjoint. Inset: additional singular values. **d**. Singular values of the gradient, weights, and the bound we derive. **e**. Variance explained per rank of the tensor decomposition of weight tensor $\mathbf{W} - W^{(0)} \otimes \mathbf{1}$. We additionally plot the variance explained over performing matrix decomposition on all possible unfoldings of the weight tensor.
Architectures. Unless noted, the initial weight std was $g = 1.5$, the input and output dimensions $m = d = 2$. **i**. Rectified tanh. We found that non-smooth activition functions seem to give the slowest decay of the singular values of the firing rate and adjoint. **ii-iii**. Softplus and tanh, which are the most common activation functions in neuroscience, have exponentially decaying firing rate and adjoint singular values, and therefore (by Theorem 1) exponentially decaying gradient singular values. **iv**. Low rank ($R = 3$) linear RNN. As per Theorem 2, the first gradient step is of rank $R + 1 = 4$. **v**. Tanh in a chaotic regime ($g = 2.1$). Despite being in a chaotic regime, the firing rate, the adjoint, and therefore (by Theorem 1) the gradient have exponentially decaying singular values.

References

[1] Niru Maheswaranathan, Alex Williams, Matthew Golub, Surya Ganguli, and David Sussillo. Universality and individuality in neural dynamics across large populations of recurrent networks. *Advances in neural information processing systems*, 32, 2019.

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

[3] Valerio Mante, David Sussillo, Krishna V Shenoy, and William T Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *nature*, 503(7474):78–84, 2013.

[4] Jing Wang, Devika Narain, Eghbal A Hosseini, and Mehrdad Jazayeri. Flexible timing by temporal scaling of cortical responses. *Nature neuroscience*, 21:102–110, 2018.

[5] Christopher J Cueva, Peter Y Wang, Matthew Chin, and Xue-Xin Wei. Emergence of functional and structural properties of the head direction system by optimization of recurrent neural networks. *arXiv preprint arXiv:1912.10189*, 2019.

[6] Abigail A Russo, Ramin Khajeh, Sean R Bittner, Sean M Perkins, John P Cunningham, Laurence F Abbott, and Mark M Churchland. Neural trajectories in the supplementary motor area and motor cortex exhibit distinct geometries, compatible with different classes of computation. *Neuron*, 107(4):745–758, 2020.

[7] 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.

[8] Matthew G Perich, Charlotte Arlt, Sofia Soares, Megan E Young, Clayton P Mosher, Juri Minxha, Eugene Carter, Ueli Rutishauser, Peter H Rudebeck, Christopher D Harvey, et al. Inferring brain-wide interactions using data-constrained recurrent neural network models. *BioRxiv*, pages 2020–12, 2020.

[9] Feng Zhu, Andrew Sedler, Harrison A Grier, Nauman Ahad, Mark Davenport, Matthew Kaufman, Andrea Giovannucci, and Chethan Pandarinath. Deep inference of latent dynamics with spatio-temporal super-resolution using selective backpropagation through time. *Advances in Neural Information Processing Systems*, 34:2331–2345, 2021.

[10] 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.

[11] Justin Jude, Matthew G Perich, Lee E Miller, and Matthias H Hennig. Robust alignment of cross-session recordings of neural population activity by behaviour via unsupervised domain adaptation. *arXiv preprint arXiv:2202.06159*, 2022.

[12] Lea Duncker, Laura Driscoll, Krishna V Shenoy, Maneesh Sahani, and David Sussillo. Organizing recurrent network dynamics by task-computation to enable continual learning. *Advances in neural information processing systems*, 33:14387–14397, 2020.

[13] 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.

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

[15] Matthew G Perich, Juan A Gallego, and Lee E Miller. A neural population mechanism for rapid learning. *Neuron*, 100(4):964–976, 2018.

[16] Barbara Feulner, Matthew G Perich, Raeed H Chowdhury, Lee E Miller, Juan A Gallego, and Claudia Clopath. Small, correlated changes in synaptic connectivity may facilitate rapid motor learning. *Nature communications*, 13(1):5163, 2022.

[17] Peter C Humphreys, Kayvon Daie, Karel Svoboda, Matthew Botvinick, and Timothy P Lillicrap. Bci learning phenomena can be explained by gradient-based optimization. *bioRxiv*, pages 2022–12, 2022.

[18] Ildefons Magrans de Abril, Junichiro Yoshimoto, and Kenji Doya. Connectivity inference from neural recording data: Challenges, mathematical bases and research directions. *Neural Networks*, 102:120–137, 2018.

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

[20] Adil G Khan, Jasper Poort, Angus Chadwick, Antonin Blot, Maneesh Sahani, Thomas D Mrsic-Flogel, and Sonja B Hofer. Distinct learning-induced changes in stimulus selectivity and interactions of gabaergic interneuron classes in visual cortex. *Nature neuroscience*, 21(6):851–859, 2018.

[21] Angus Chadwick, Adil G Khan, Jasper Poort, Antonin Blot, Sonja B Hofer, Thomas D Mrsic-Flogel, and Maneesh Sahani. Learning shapes cortical dynamics to enhance integration of relevant sensory input. *Neuron*, 111(1):106–120, 2023.

[22] Sukbin Lim, Jillian L McKee, Luke Woloszyn, Yali Amit, David J Freedman, David L Sheinberg, and Nicolas Brunel. Inferring learning rules from distributions of firing rates in cortical neurons. *Nature neuroscience*, 18(12):1804–1810, 2015.

[23] Ian Stevenson and Konrad Koerding. Inferring spike-timing-dependent plasticity from spike train data. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[24] Scott Linderman, Christopher H Stock, and Ryan P Adams. A framework for studying synaptic plasticity with neural spike train data. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[25] 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.

[26] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[27] Alex H Williams, Tony Hyun Kim, Forea Wang, Saurabh Vyas, Stephen I Ryu, Krishna V Shenoy, Mark Schnitzer, Tamara G Kolda, and Surya Ganguli. Unsupervised discovery of demixed, low-dimensional neural dynamics across multiple timescales through tensor component analysis. *Neuron*, 98(6):1099–1115, 2018.

[28] Hugo Soulat, Sepiedeh Keshavarzi, Troy Margrie, and Maneesh Sahani. Probabilistic tensor decomposition of neural population spiking activity. *Advances in Neural Information Processing Systems*, 34:15969–15980, 2021.

[29] Arthur Pellegrino, Heike Stein, and N Alex Cayco-Gajic. Disentangling mixed classes of covariability in large-scale neural data. *bioRxiv*, pages 2023–03, 2023.

[30] Rishidev Chaudhuri, Berk Gerçek, Biraj Pandey, Adrien Peyrache, and Ila Fiete. The intrinsic attractor manifold and population dynamics of a canonical cognitive circuit across waking and sleep. *Nature neuroscience*, 22(9):1512–1520, 2019.

[31] Richard J Gardner, Erik Hermansen, Marius Pachitariu, Yoram Burak, Nils A Baas, Benjamin A Dunn, May-Britt Moser, and Edvard I Moser. Toroidal topology of population activity in grid cells. *Nature*, 602(7895):123–128, 2022.

[32] Manuel Beiran, Nicolas Meirhaeghe, Hansem Sohn, Mehrdad Jazayeri, and Srdjan Ostojic. Parametric control of flexible timing through low-dimensional neural manifolds. *Neuron*, 2023.

[33] Xulu Sun, Daniel J O'Shea, Matthew D Golub, Eric M Trautmann, Saurabh Vyas, Stephen I Ryu, and Krishna V Shenoy. Cortical preparatory activity indexes learned motor memories. *Nature*, 602(7896):274–279, 2022.

[34] Darby M Losey, Jay A Hennig, Emily R Oby, Matthew D Golub, Patrick T Sadlter, Kristin M Quick, Stephen I Ryu, Elizabeth C Tyler-Kabara, Aaron P Batista, Byron M Yu, et al. Learning alters neural activity to simultaneously support memory and action. *bioRxiv*, pages 2022–07, 2022.

[35] Robert J Van Beers, Patrick Haggard, and Daniel M Wolpert. The role of execution noise in movement variability. *Journal of neurophysiology*, 91(2):1050–1063, 2004.

[36] L Bittner. Ls pontryagin, vg boltyanskii, rv gamkrelidze, ef mishechenko, the mathematical theory of optimal processes. viii+ 360 s. new york/london 1962. john wiley & sons. preis 90/–, 1963.

[37] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[38] Balasubrahmanyan Srinivasan, Upadrasta Ravikirana Prasad, and Nalam Jaganmohan Rao. Back propagation through adjoints for the identification of nonlinear dynamic systems using recurrent neural models. *IEEE Transactions on Neural Networks*, 5(2):213–228, 1994.

[39] Xuechen Li, Ting-Kam Leonard Wong, Ricky T. Q. Chen, and David Duvenaud. Scalable gradients for stochastic differential equations. *International Conference on Artificial Intelligence and Statistics*, 2020.

[40] Per Christian Hansen. *Rank-deficient and discrete ill-posed problems: numerical aspects of linear inversion*. SIAM, 1998.

[41] Jarosław Buczyński and Joseph M Landsberg. Ranks of tensors and a generalization of secant varieties. *Linear Algebra and its Applications*, 438(2):668–689, 2013.

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

[43] Cengiz Pehlevan, Tao Hu, and Dmitri B Chklovskii. A hebbian/anti-hebbian neural network for linear subspace learning: A derivation from multidimensional scaling of streaming data. *Neural computation*, 27(7):1461–1495, 2015.

[44] Shanshan Qin, Shiva Farashahi, David Lipshutz, Anirvan M Sengupta, Dmitri B Chklovskii, and Cengiz Pehlevan. Coordinated drift of receptive fields in hebbian/anti-hebbian network models during noisy representation learning. *Nature Neuroscience*, pages 1–11, 2023.

[45] Steffen Schotthöfer, Emanuele Zangrando, Jonas Kusch, Gianluca Ceruti, and Francesco Tudisco. Low-rank lottery tickets: finding efficient low-rank neural networks via matrix differential equations. *arXiv preprint arXiv:2205.13571*, 2022.

[46] Charles H Martin and Michael W Mahoney. Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning. *The Journal of Machine Learning Research*, 22(1):7479–7551, 2021.

[47] Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. *Advances in Neural Information Processing Systems*, 32, 2019.

[48] Othmar Koch and Christian Lubich. Dynamical low-rank approximation. *SIAM Journal on Matrix Analysis and Applications*, 29(2):434–454, 2007.

[49] John A Rhodes. A concise proof of kruskal's theorem on tensor decomposition. *Linear Algebra and its Applications*, 432(7):1818–1824, 2010.

[50] Tamara G Kolda. Orthogonal tensor decompositions. *SIAM Journal on Matrix Analysis and Applications*, 23(1):243–255, 2001.

[51] Grigorios A Pavliotis. *Stochastic processes and applications: diffusion processes, the Fokker-Planck and Langevin equations*, volume 60. Springer, 2014.

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

[53] Walter Rudin. Functional analysis. *(No Title)*, 1973.

[54] Roger A Horn and Charles R Johnson. Topics in matrix analysis cambridge university press. *Cambridge, UK*, 1991.

[55] Rudolf Emil Kalman. Mathematical description of linear dynamical systems. *Journal of the Society for Industrial and Applied Mathematics, Series A: Control*, 1(2):152–192, 1963.