Extracting computational mechanisms from neural data using low-rank RNNs

Adrian Valente[*]
École Normale Supérieure
PSL Research University

Jonathan W. Pillow
Princeton Neuroscience Institute
Princeton University

Srdjan Ostojic
École Normale Supérieure
PSL Research University

Abstract

An influential framework within systems neuroscience posits that neural computations can be understood in terms of low-dimensional dynamics in recurrent circuits. A number of methods have thus been developed to extract latent dynamical systems from neural recordings, but inferring models that are both predictive and interpretable remains a difficult challenge. Here we propose a new method called Low-rank Inference from Neural Trajectories (LINT), based on a class of low-rank recurrent neural networks (lrRNNs) for which a link between connectivity and dynamics has been previously demonstrated. By fitting such networks to trajectories of neural activity, LINT yields a mechanistic model of latent dynamics, as well as a set of axes for dimensionality reduction and verifiable predictions for inactivations of specific populations of neurons. Here, we first demonstrate the consistency of our method and then apply it to two use cases: (i) we reverse-engineer "black-box" vanilla RNNs trained to perform cognitive tasks, and (ii) we infer latent dynamics and neural contributions from electrophysiological recordings of nonhuman primates performing a similar task.

1 Introduction

As large-scale neural recordings in behaving animals are becoming commonplace, a pressing question is how computational principles can be extracted from the electrical activity of thousands of cells. An influential framework posits that neural computations rely on latent low-dimensional dynamics [54, 60] distributed across populations of neurons [66, 43]. In line with this proposal, a number of data-analysis methods have been developed to infer latent dynamics from neural recordings [65, 25, 36, 32, 12, 31, 24, 33, 10, 19, 16, 40, 41, 45, 22] (reviewed in [11]). While these statistical approaches often provide compelling descriptions of the recorded data, they generally lack a mechanistic interpretation that would, for instance, allow them to make predictions for responses to novel interventions on the underlying neural circuits. How to identify mechanistic, predictive models from neural data thus currently remains an important challenge.

Recurrent neural networks (RNNs) have emerged as key models for studying neural computations [64]. Indeed, RNNs can be trained to solve various cognitive tasks, and lead to dynamics surprisingly similar to those observed in neural recordings [29, 56, 5, 61, 39], and have also successfully been trained to reproduce the activity of neurons recorded *in vivo* [38, 6, 14, 35]. While potentially predictive, the obtained networks are however typically challenging to understand mechanistically [55, 28, 27, 26, 63, 57], and ongoing research directions aim at reducing them to simplified, interpretable models, for example through the use of linearized dynamical systems [12, 19, 50] or through "network distillation" methods [44, 22].

[*]Correspondence to `adrian.valente@ens.psl.eu`.
Code available at `https://github.com/adrian-valente/lowrank_inference/`

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Here, we exploit a particular class of interpretable RNNs, namely *low-rank RNNs* (lrRNNs) [30, 3, 9, 46, 47, 58], to develop a new method that extracts mechanistic and predictive low-dimensional models from observed neural activity, and can be applied to both artificial and biological neural networks. Previous work has performed theoretical analyses of low-rank RNNs, either designed or trained on specific tasks [30, 9], but has not introduced methods for fitting them to neural activity. Our method, Low-rank Inference from Neural Trajectories (LINT) infers a minimal rank lrRNN from a dataset and then exploits the theory of low-rank networks to relate the obtained connectivity with low-dimensional dynamics and computations. It produces three outputs: a set of axes for dimensionality reduction of neural population activity, an effective connectivity that implements a latent dynamical system, and predictions for inactivations on specific subsets of recorded neurons (fig. 1a).

Our contributions can be summarized in three steps: first, we verify the consistency of LINT by applying it to data simulated from lrRNNs, and show that it recovers the effective part of the connectivity that reproduces the dynamics and computations. Second, we apply LINT to data generated from full-rank RNNs trained on cognitive tasks, and demonstrate that the resulting lrRNNs can capture their dynamics and offer mechanistic insights into how they function. Notably, we identify a novel population-based mechanism enabling context-dependent switching in a vanilla RNN, and verify it by targeted inactivation experiments. Finally, we apply LINT to electrophysiological recordings in nonhuman primates performing a context-dependent decision making task [29]; this reveals that a rank-1 network can capture most aspects of the dynamics present in the data, and that computations in this neural circuit appear to be supported by a small proportion of all recorded neurons.

2 Approach

Here we describe our method for extracting an interpretable low-dimensional projection and dynamical model from neural trajectories by fitting a low-rank recurrent network to data.

Low-rank RNNs (lrRNNs). We start from rate-based recurrent neural networks of N units, each characterized by an activation variable x_i which follows the dynamics:

$$\tau \frac{dx_i}{dt} = -x_i + \sum_{j=1}^{N} J_{ij}\phi\left(x_j\right) + \sum_{s=1}^{N_{in}} I_i^{(s)} u_s(t) + \eta_i(t). \tag{1}$$

Here $\boldsymbol{J}$ represents the network connectivity matrix and ϕ is a nonlinear transfer function defining the neural firing rate $\phi(x_i)$, which we take here to be $\tanh$. Each network also receives N_{in} input signals $u_s(t)$ via a set of weights $\boldsymbol{I}^{(s)}$ that we refer to as *input vectors*.

Low-rank RNNs represent a subclass of models in which the connectivity matrix is constrained to be of finite rank $R \ll N$ [30, 3, 9]. In this case, $\boldsymbol{J}$ can be written as a sum of outer products of *connectivity vectors* $\boldsymbol{n}^{(r)}$ and $\boldsymbol{m}^{(r)}$:

$$J_{ij} = \frac{1}{N} \sum_{r=1}^{R} m_i^{(r)} n_j^{(r)}. \tag{2}$$

To define a unique representation, we take as $\boldsymbol{m}^{(r)}$ the left singular vectors of the connectivity matrix, and as $\boldsymbol{n}^{(r)}$ the right singular vectors with an appropriate normalization.

Network inference. We infer lrRNNs from data by training the connectivity parameters to reproduce recorded neural trajectories (either single-trial or condition-averaged trajectories). Specifically, we use back-propagation on connectivity vector parameters $n_i^{(r)}$ and $m_i^{(r)}$ and input parameters $I_i^{(s)}$ to minimize the squared difference between target and reproduced trajectories:

$$\mathcal{L} = \sum_{c=1}^{C} \sum_{i=1}^{N} \sum_{t=1}^{T} (\phi(x_i^{(c)}(t)) - \phi(\hat{x}_i^{(c)}(t)))^2 \tag{3}$$

where $x_i^{(c)}(t))$ represents the target trajectory in condition c, for neuron i and timestep t and $\hat{x}_i^{(c)}(t)$ the corresponding trajectory produced by the model.

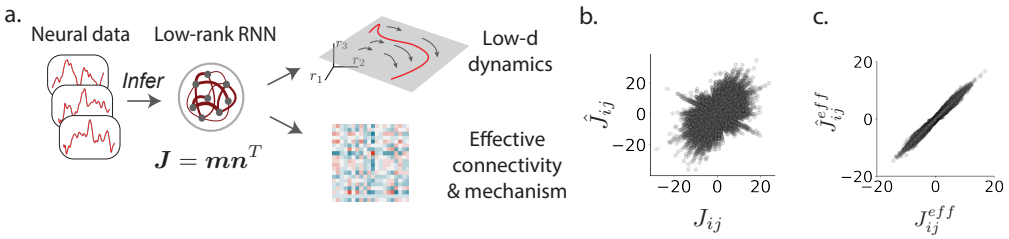


Figure 1: **a**. LINT pipeline: Low-rank RNNs are fitted to either simulated or recorded neural trajectories. The obtained lrRNNs provide interpretable low-dimensional dynamics as well as a computational mechanism based on an effective connectivity structure. **b-c**. LINT tested on trajectories simulated from a rank-1 lrRNN trained to perform the CDM task (see Table 1 for results across tasks). **b**. For each pair of neurons, relationship between the original (J_{ij}) and inferred ($\hat{J}_{ij}$) synaptic connectivity (512^2 pairs, $r = 0.57$). **c**. For the same networks, relationship between the original effective connectivity (J_{ij}^{eff}, see text) and the inferred one ($\hat{J}_{ij}^{eff}$, $r = 0.99$)

Dimensionality reduction. One property of lrRNNs is that, in the absence of noise, they constrain the activity vector $\boldsymbol{x}(t)$ to evolve in a low-dimensional subspace spanned by the R connectivity vectors $\boldsymbol{m}^{(r)}$ and the N_{in} input vectors $\boldsymbol{I}^{(s)}$ [3]. The trial-averaged trajectories therefore explore at most $R + N_{in}$ dimensions and can be parametrized as:

$$\boldsymbol{x}(t) = \sum_{r=1}^{R} \kappa_r(t)\boldsymbol{m}^{(r)} + \sum_{s=1}^{N_{in}} v_s(t)\boldsymbol{I}^{(s)}, \tag{4}$$

where $v_s(t)$ are the low-pass filtered input signals $u_s(t)$ (see Appendix A), and κ_r are a set of latent variables generated by recurrent activity. Thus, lrRNNs provide by design a reduction of N-dimensional neural activity to at most $R + N_{in}$ dimensions that can be directly interpreted in terms of components on a *recurrent subspace* spanned by the connectivity vectors $\boldsymbol{m}^{(r)}$ and on an *input subspace* spanned by inputs vectors $\boldsymbol{I}^{(s)}$ [61].

Latent dynamics. In lrRNNs, the latent variables $\kappa_r(t)$ form a non-linear low-dimensional dynamical system, described by:

$$\frac{d}{dt}\boldsymbol{\kappa}(t) = F(\boldsymbol{\kappa}(t), \boldsymbol{u}(t)) \tag{5}$$

where $\boldsymbol{\kappa}(t) = \{\kappa_r(t)\}_{r=1..R}$ is an R-dimensional vector representing the activity on the recurrent subspace, $\boldsymbol{u}(t)$ represents the input signals, and F is a non-linear function that can be directly determined from network connectivity parameters [3, 9] (Appendix A). Moreover, it can be shown that rank-R networks are universal approximators of R-dimensional dynamical systems [3], and thus that the function F can approximate any non-linear mapping in R dimensions.

Task-optimized RNNs. We applied our method first to trajectories produced by task-optimized RNNs, which produce an output signal $z(t)$ from the recurrent dynamics (1) via a linear readout:

$$z(t) = \sum_{i=1}^{N} w_i \phi(x_i(t)) \tag{6}$$

In task-optimized networks, the parameters are trained with backpropagation through time to minimize the squared error between the output $z(t)$ and a target $z^*(t)$. In this work, we consider four cognitive tasks: Decision Making (DM), Working Memory (WM), Context-Dependent Decision Making (CDM, fig. 2a) and Delayed Match-to-Sample (DMS) (see Appendix B for task definitions). To analyze the computations underlying these tasks, we generated neural trajectories from both low-rank (in section 3.1) and full-rank (see section 3.2) task-optimized RNNs. All networks were implemented in pytorch [34]; training details can be found in Appendix C.

Table 1: Synthetic data validation results for LINT
(CC: connectivity correlation, ECC: effective connectivity correlation)

Task	Rank	Trajectory R^2	CC	ECC
Decision Making (DM)	1	0.97	0.50	0.99
Working Memory	2	0.96	0.43	0.93
Context-dependent DM	1	0.91	0.57	0.99
Delayed Match-to-Sample	2	0.98	0.39	0.63

3 Results

3.1 Validation with synthetic data and effective aspects of connectivity

We first validated the capacity of the LINT method to recover low-rank connectivity features and reproduce neural trajectories on data simulated from lrRNNs trained on cognitive tasks. For this, we first trained lrRNNs with 512 neurons to produce a correct behavioral output on four systems neuroscience tasks, each time retaining the minimal rank solution [9] (see Table 1). For each task-optimized lrRNN, we then generated simulated trajectories corresponding to trials in the trained task, and applied our method to fit these trajectories using equivalent models with the same rank and number of neurons. We found that the inferred networks were able to reliably reproduce the original trajectories when fed with the same inputs, as quantified by the R^2 scores between original and fitted trajectories. Although the inferred networks were not explicitly trained on behavioral outputs, they were able to perform their task accurately when plugged to the original readout vector, implying that they also captured behavioral aspects of the original networks.

The output of LINT is an inferred low-rank connectivity that we compared with the original one. We noted that the inferred connectivity weights are not identical to the original ones, although a certain degree of correlation is present (fig. 1b and Table 1, column CC for connectivity correlation). However, from a theoretical point of view, a range of different connectivity matrices can lead to identical dynamics [37, 13, 3]. The low-rank framework allows for a precise characterization of the relevant part of the connectivity, and in particular allows us to define an effective connectivity matrix $\boldsymbol{J}^{\text{eff}}$ that captures the minimal features required to obtain certain dynamics. Indeed, for this class of networks, the low-dimensional dynamics described by equations (4)-(5) depend on the exact entries on the $\boldsymbol{m}^{(r)}$ and $\boldsymbol{I}^{(s)}$ vectors, but not on the individual entries on the $\boldsymbol{n}^{(r)}$ vectors. Instead, $\boldsymbol{n}^{(r)}$ vectors influence the dynamics only through their global projections on the $\boldsymbol{m}^{(r)}$ and $\boldsymbol{I}^{(s)}$ vectors (see details in Appendix A). Thus, the components of $\boldsymbol{n}^{(r)}$ orthogonal to the space spanned by $\boldsymbol{m}^{(r)}$ and $\boldsymbol{I}^{(s)}$ are irrelevant for the dynamics, and removing them leads to an effective connectivity matrix $\boldsymbol{J}^{\text{eff}}$ which captures the minimal features required for obtaining particular dynamics. Our results on the synthetic data showed indeed a very high degree of similarity between original and inferred effective connectivities, demonstrating that LINT recovered the relevant aspects of the neural connectivity (Table 1, ECC for Effective Connectivity Correlation, and fig. 1c).

In conclusion, LINT is able to accurately fit neural trajectories generated by low-rank RNNs, to infer networks that give rise to similar trajectories and behavioral performance, and to recover the essential features of the connectivity.

3.2 Application to reverse-engineering full-rank RNNs

We next ask to which extent our method can infer computational mechanisms from activity generated by unconstrained, full-rank networks. Indeed, RNNs trained on simple tasks without a rank constraint often exhibit low-dimensional dynamics that can be related to the way computations unfold in the network [55, 28, 27, 26, 57]. It has been found that such low-dimensional dynamics can be reproduced by low-rank networks [3], yet the best approach for inferring the corresponding low-rank connectivity remains to be determined. Here we show that our method vastly outperforms a direct low-rank approximation of the connectivity matrix based on truncating the SVD [47]. We then demonstrate how the inferred low-rank models can be used to interpret the computational mechanisms in the original full-rank networks.

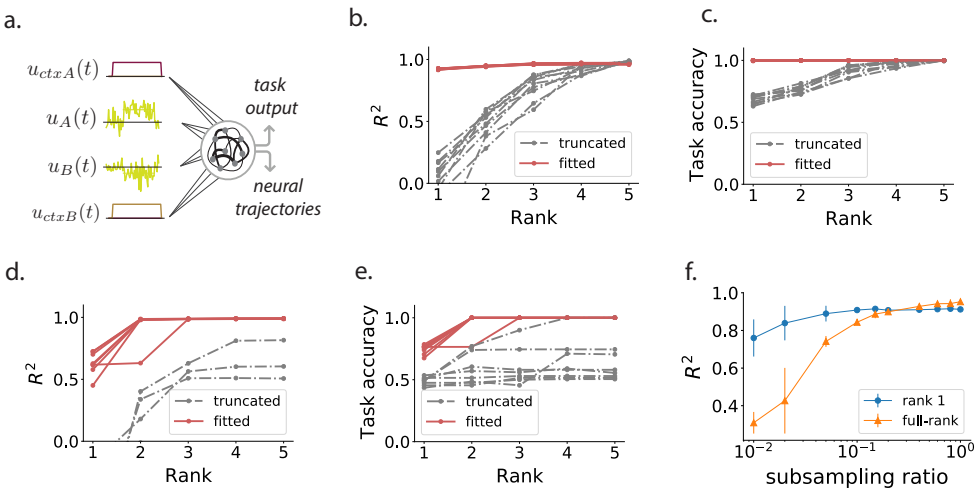


Figure 2: LINT applied to full-rank task-optimized RNNs. **a**. Description of the context-dependent decision making (CDM) task inputs and outputs. Neural trajectories correspond to the neural firing rates, task output refers to a linear readout as in eq. 6. **b**. For the CDM task, similarity between trajectories produced by original full-rank networks and a series of fitted networks of increasing rank (red), or low-rank networks obtained by truncating the original connectivity matrix (grey). Each line corresponds to a randomly initialized full-rank network. **c**. Task accuracy of the same networks when associated with the original task readout. **d-e**. Same as b-c. for the delayed match-to-sample (DMS) task. **f**. For the CDM task, similarities between trajectories of the original and fitted networks of rank one (blue) or full-rank (orange), when networks are fitted only to a random subset of neurons of the original network. Error bars: mean $\pm$1std over 10 random subsamples for each ratio value (see also sup. fig. 8).

3.2.1 Extracting low-dimensional dynamics through low-rank connectivity

We considered full-rank, vanilla RNNs trained on two systems neuroscience tasks, respectively context-dependent decision-making[29] (CDM) and delayed match-to-sample[5] (DMS). The full-rank RNNs reached a 100% accuracy on each task, and—as expected—exhibited low-dimensional dynamics. From these RNNs, we generated trajectories corresponding to trials in the trained tasks, and then used LINT to infer lrRNNs of increasing rank.

In the CDM task, a network received two signal inputs which varied randomly around a positive or negative mean, as well as two binary context cues, one of which was set to 1 in each trial. The network was trained to output the average sign of the signal indicated by the active context, while ignoring the other signal (fig. 2a, see Appendix B for task details). Applying LINT, we found that rank-1 networks are sufficient to accurately reproduce the trajectories of the trained full-rank networks. Increasing the rank improved the goodness-of-fit only marginally (fig. 2b). In contrast, low-rank networks obtained by truncating the SVD of the trained full-rank connectivity matrix required a rank of 4 or 5 to reach a comparable accuracy (fig. 2b), and much higher for different hyperparameters (sup. fig. 1). This shows that LINT captures a simplified connectivity structure that could not be trivially extracted from the original connectivity. Moreover, even though not explicitly trained on the behavioral outputs, the inferred rank-1 networks performed the task correctly when plugged onto the original readout, implying that they faithfully captured the task-related dynamics (fig. 2c).

For the DMS task, we similarly found that rank-2 networks were able both to capture the dynamics of the trained full-rank network and to accurately perform the task, with no notable improvement when the rank is increased (fig. 2d-e). As for the CDM task, LINT vastly outperformed direct truncation of the full-rank connectivity matrix. For both these tasks, reproducing the experiment over a larger number of full-rank RNNs, trained with diverse hyperparameters and random seeds led to similar results (sup. fig. 1).

Finally, it is important to note that the CDM and DMS tasks are particularly prone to low-dimensional implementations, which is not necessarily the case for all computations of interest in systems neuroscience [18]. To probe the capacities of LINT with higher-dimensional tasks, we applied it to full-rank networks trained on the K-back task, where a network receives a random number P of stimulations in $\{-1, 1\}$ and has to output the K-th last stimulation received, a task directly inspired from sequence working memory paradigms [62]. This task requires the implementation of an internal working memory with a capacity of at least K bits, and should thus require a rank at least K. We indeed observed that the rank identified by LINT increased with K, but only linearly (see details in appendix F and sup. fig. 9). These results show that LINT is also helpful in probing the dimensionality required for more complex tasks.

Subsampling. In experimental settings, one cannot expect to have access to all units of a network. It is thus important to assess whether we can still recover the relevant low-dimensional dynamics and information when only a handful of neurons from a network are recorded. We considered a full-rank network trained on the CDM task, and fitted networks to its trajectories, either without constraining their rank or by fixing it to one. Without subsampling, the inferred full-rank networks slightly outperformed rank-1 networks in reproducing the original trajectories. Yet when considering subsamples of neurons, rank-1 networks appear to be more robust than unconstrained ones, maintaining a good performance until subsampling ratios as low as 1% of original neurons (fig. 2f). This experiment was reproduced on the DMS task setting, showing a similar advantage for low-rank fitted networks (sup. fig. 8).

3.2.2 Extracting computational mechanisms from inferred low-rank connectivity

Low-rank models of behavioral tasks open the door to mechanistic interpretations of the underlying dynamics [30, 9]. Here we show how LINT allows us to extract computational mechanisms from full-rank networks performing the CDM task.

A first output of LINT is a set of interpretable axes defining a task-related subspace for dimensionality reduction. The inferred rank-1 model of the CDM task specifically yields five axes that correspond to the two stimulus inputs, the two context inputs and an internal latent variable, generated by recurrent connectivity, which represents integrated evidence and therefore choice. Projecting the activity of the full-rank network along these axes shows how low-dimensional dynamics transform inputs into the choice output (fig. 3). In this case, the axes determined by LINT are closely related to those obtained by standard targeted dimensionality reduction (TDR, sup. fig. 2) [29]. However, in contrast with TDR, the axes inferred by LINT correspond to connectivity features causing the low-dimensional dynamics, and provide a method for an unsupervised discovery of recurrently-generated latent variables. In particular, we did not *a priori* specify that the activity along $\boldsymbol{m}$ should encode choice, but directly observed that choice was generated along that axis through recurrent dynamics.

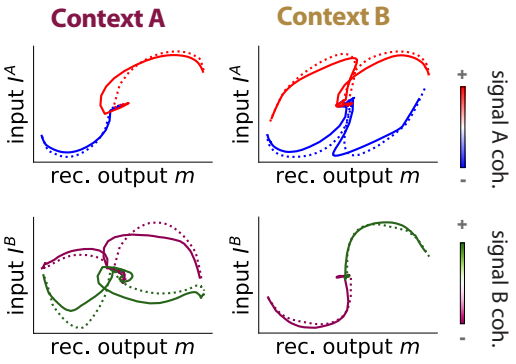


Figure 3: Low-dimensional projections produced by LINT. Trial-averaged trajectories for different task conditions, in the original full-rank network (full lines) and the inferred low-rank network (dashed lines), projected on axes obtained from the fitted lrRNN connectivity: input A and B axes, and the output axis of the rank-1 recurrent connectivity $\boldsymbol{m}$, which encodes choice. All trajectories start at center.

Going beyond dimensionality reduction, LINT extracts an effective low-dimensional model of the task. For the CDM task, the inferred model is a one-dimensional, non-linear latent dynamical system, where the latent variable integrates the relevant evidence. To understand how context-dependent selection of the evidence was performed, we analyzed the connectivity parameters of the inferred rank-1 RNN. Indeed, recent work has introduced a clustering method for analyzing low-rank connectivity, and has showed that in trained lrRNNs, context-dependent integration relies on gain-modulation mechanism mediated by two distinct populations of neurons [9]. Applying a similar clustering approach to rank-1 networks inferred by LINT led to a

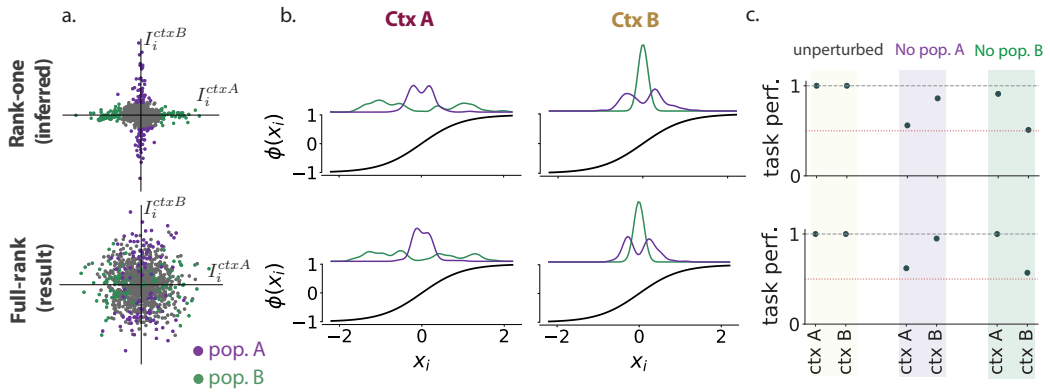


Figure 4: Extracting mechanisms and testing predictions from a low-rank network (top) inferred from a full-rank network trained on the CDM task (bottom). **a**. Three populations of neurons (grey, green and purple) are identified by clustering connectivity parameters in the fitted rank-1 network (top). Here, we plot the same populations in joint distributions of contextual input vectors in the rank-1 (top) vs full-rank (bottom) networks. The populations are not directly identifiable from input parameters in the original network (bottom). **b**. Neural transfer function (tanh) and smoothed distributions of mean neural activity for populations A and B in the inferred rank-1 network (top) and the full-rank network (bottom). **c**. Inactivation experiments: task performance on each context for unperturbed networks and after inactivating populations A and B in the fitted rank-1 network (top) and inactivating the same neurons in the original unconstrained network (bottom). Red dots: chance level.

comparable mechanism. Specifically, this method identified two populations distinguished by strong values of the fitted weights for the context-cue inputs (fig. 4a top and sup. fig. 2). In each context, these context-cue inputs place the neurons belonging to the two populations at different positions on the non-linear transform function $\phi(x)$ (fig. 4b top), thereby modulating in opposite ways their gains (defined for each neuron as the local slope $\phi'(x)$). Combined with different statistics of the connectivity parameters on each of the populations, this modulation is sufficient to implement the desired context-switching behavior of the network (see details in Appendix D). The same pattern of gain modulation was found in the original full-rank network, indicating that it is using a similar mechanism (fig. 4b bottom).

The computational mechanism extracted using LINT produces predictions for inactivations that can be directly tested in the original full-rank model. The analysis described above assigns neurons to different populations that have specific computational roles. Specifically, in the inferred rank-1 network, inactivating separately each population leads to a specific loss of performance in a single context, but not the other one (fig. 4c top and sup. fig. 2 for detailed error patterns). Since individual neurons in the original networks directly correspond to neurons in the rank-1 network on a one-to-one basis, the identified populations can be directly mapped back onto the original network to reproduce the inactivation experiment. The predicted effects of inactivations on context-dependent performance were directly replicated in the full-rank network, demonstrating that it relies on the identified computational mechanism (fig. 4c bottom and sup. fig. 2). One can note that although the two identified populations clearly stand out in the inferred rank-1 connectivity, they are not directly apparent in the original full-rank connectivity (fig. 4a bottom), showing that although the mechanism extracted by LINT applies to the full-rank network, using an lrRNN was a necessary step to uncover it.

3.3 Application to neural recordings

We next applied our method to *in vivo* recordings. We considered here electrophysiological recordings from non-human primates performing a context-dependent decision-making task similar to that studied in previous sections [29]. More specifically, two macaques (designed by A and F) were presented with random dots stimuli that varied along two dimensions: the overall motion direction of the dots, and their overall color, ranging from two extremes with a set of intermediary coherences in between. At the beginning of each trial, a cue indicated a context for the trial, ordering the

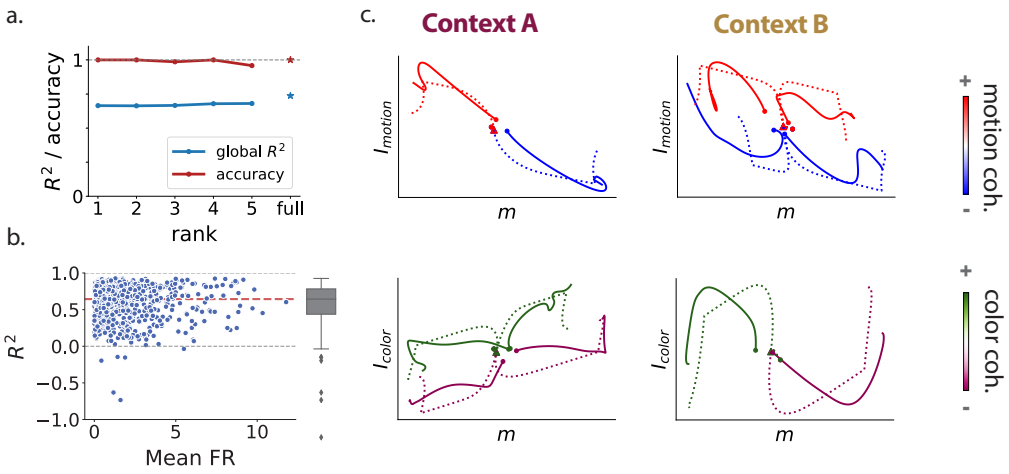


Figure 5: LINT applied to neural recordings from a non-human primate performing a context-dependent task (monkey A) [29]. **a**. Global R^2 of model trajectories with respect to original ones, and task accuracy of inferred networks of increasing ranks. **b**. For the rank-1 inferred network, R^2 of each recorded neuron plotted against its average data firing rate, median R^2 in red, with marginal box-plot of R^2 values. 14 neurons with an $R^2 < -1$ are not shown, all with a mean firing rate < 2.2 (see sup. fig. 4). **c**. Trial-averaged trajectories in the recorded data (full lines) and for the rank-1 model (dashed lines), for different task conditions (only one coherence value for each coherence sign is plotted here), projected on axes identified by LINT (motion and color input axes and the output axis $\boldsymbol{m}$ of the rank-1 recurrent matrix). Trajectories start at the center.

subject to report either the average motion or color with an eye saccade, while ignoring the irrelevant evidence. Non-simultaneous recordings were performed in the frontal eye field (FEF) an area of the macaque prefrontal cortex, with respectively 727 neurons for monkey A and 574 neurons for monkey F recorded in all 72 conditions the task presented (ignoring error trials).

For the analysis, we started by binning (5 ms) and smoothing (50 ms std Gaussian window) spike train data, and then computed the trial-averaged response of every neuron in each of the task conditions, forming a pseudo-population tensor of size $N \times T \times 72$ with N the number of neurons in each monkey and $T = 150$ the number of discrete time steps. We denoised this tensor by projecting its first axis on the subspace spanned by the top 12 principal components of the pseudo-population activity, and then projecting back to the high-dimensional space spanned by all neurons [29]. Finally, in order to consider only task-related neural activity, we subtracted from each neuron's trajectory its condition-averaged mean. We denote the entries of the final obtained tensor by $\tilde{x}_{i,t,c}$.

We applied LINT to the obtained trajectories, training the activation of each unit in the low-rank RNN (noted $x_{i,t,c}$) to match the corresponding pre-processed activity. Importantly, the inferred networks received inputs following the same structure than in the CDM task of previous sections, that is two noisy signal inputs and two contextual cue inputs (fig. 2a), and were left unconstrained for the first 350 ms of each trial while receiving only contextual cues, to account for the original task procedure. In particular, we made the implicit hypothesis that choice signals are generated by the recurrent activity of the network from received inputs. The quality of fits was quantified by leaving out a random subset of 8 conditions during network inference, and evaluating the R^2 of fitted networks on these left-out conditions.

We found that for both monkeys the neural activity was well reproduced by a rank-1 network, with minimal improvements when the rank was increased (fig. 5a and sup. fig. 5, $R^2 = 0.66$ for monkey A, and $R^2 = 0.57$ for monkey F). Moreover, when simulated independently, the inferred rank-1 networks were able to perform the task by adding a linear readout (accuracy curves in fig. 5a, sup. fig. 5). This was the case even though the recurrent and input connections were not trained on task performance, demonstrating that the reproduced trajectories contained information about the choice made by the monkey. The activity of both original and reproduced trajectories can moreover be projected on the axes identified by LINT, providing a geometrical picture of task execution (fig. 5c). These axes were

closely related to those found by targeted dimensionality reduction: notably the context, motion, and color axes identified by TDR closely match the corresponding input axes in the inferred network, while the choice TDR axis could be identified to the $\boldsymbol{m}$ axis of the rank-1 recurrent connectivity (sup. figs. 4,5). The projections of the original activity on the connectivity axes (full lines on fig. 5b) therefore showed how inferred connectivity explains the geometry of recorded data.

A closer look at the distribution of inferred connectivity weights provided more information on the neural mechanisms at play (sup. figs. 6, 7). In contrast to the fits of task-optimized networks, we observed for both monkeys a clearly non-normal, heavy-tailed distribution of inferred connectivity parameters (Fisher's kurtosis between 4.9 and 62.7 for different connectivity parameters), echoing some past observations on biological synaptic weights [51, 4]. Separating with a clustering algorithm (GMM) large and small-weight neurons showed that the latter could be inactivated without affecting the task performance of the network in a significant way (sup. figs. 6, 7), even though they represented a majority of neurons in the circuits (570/727 neurons in monkey A, 389/574 in monkey F). This suggests that a minority of neurons with large connectivity weights supports the computation performed by these networks.

4 Discussion

In this paper, we introduced LINT, a new method for inferring a latent dynamical system from neural activity based on the theory of low-rank RNNs. This method yields a mechanistic, interpretable and predictive model of neural trajectories, and bridges different levels of analysis, from state-space geometry to neural connectivity. After verifying the consistency of our method, we demonstrated its potential to extract the mechanisms used by "black-box" vanilla RNNs trained to perform cognitive tasks. In particular, we found that low-rank RNNs reproduced well the dynamics of full-rank RNNs trained on a variety of tasks and across a range of hyperparameters. Moreover, the obtained lrRNNs provided interpretable low-dimensional projections of the fitted activity, as well as predictions for the effects of inactivations of specific populations of neurons. These predictions were verified in the original network, validating computational mechanisms derived from inferred low-rank connectivity weights. Finally, we applied LINT to neural activity recorded in the prefrontal cortex of nonhuman primates during a context-dependent decision-making task. LINT was able to infer rank-1 networks that reproduced both neural activity and task performance, from which low-dimensional projections and interpretable connectivity could be extracted.

Our results pave the way for a better understanding of how RNNs perform computations. In particular, LINT is complementary to other approaches seeking to reverse-engineer RNNs [55, 29, 50] in that it infers an effective connectivity that links low-dimensional dynamics with interactions at the single neuron level, opening possibilities for goal-oriented interventions and a better control on the behavior of trained networks. Moreover, LINT leads to compressed, distilled versions of large networks that can be of practical interest to engineers, as suggested by recent research in low-rank training of deep neural networks [20]. Extensions to reverse-engineering more complex architectures like GRUs or LSTMs, which have already been shown to rely on low-dimensional trajectories [28, 26, 21] would be an interesting direction for future research.

It is worth noting that other statistical methods such as mTDR [1] can provide a better fit to the same data, but do not identify a dynamical system that generates the trajectories. LINT is related to switching latent dynamical systems [10, 50, 24, 19, 12, 58] in that it bridges descriptive latent-space analyses like mTDR with methods that identify a predictive model of the dynamics, such as LFADS [33]. A distinctive feature of LINT is however that (unlike LFADS) it retains a mechanistic description at the level of neural connectivity of how the dynamics emerge from interactions between neurons.

As a number of other data analysis methods for neuroscience, our approach relies on the ubiquitous observation that neural dynamics appear to be confined to low-dimensional manifolds within the activity state-space [7, 17, 43]. Recent works have proposed that this low-dimensionality might be an artifact stemming from the simplicity of the examined tasks [18], or have reported higher-dimensional activity patterns [53, 23, 59]. This raises the question of how far our method could account for activity in more complex tasks. Our results on the K-back task suggest that LINT scales well with task dimensionality, and hint to a possible compositional implementation of internal computations

that could be unpacked by breaking them into simpler primitives [63, 8]. Further confronting our and related methods to higher-dimensional task designs however remains an important endeavour.

Distinguishing signals generated by dynamics within a given area from external inputs it receives is a major challenge for any interpretation of neural recordings, and a focus of recent work [16, 45, 42]. Most methods for inferring latent dynamics from activity data rely on *ad hoc* assumptions about the inputs received by a circuit. Following previous work [29], when analyzing recordings from the prefrontal cortex we hypothesized that choice was recurrently generated within the recorded area. LINT however potentially provides a new approach for testing hypotheses on input structure, by comparing models fits obtained from single-trial recordings, or by building multi-area models [2] from large-scale recordings that are increasingly available nowadays [49, 48, 52, 35].

Acknowledgments and Disclosure of Funding

The authors would like to thank Valerio Mante for kindly providing access to the monkey data, and João Barbosa for relevant discussions. AV and SO were supported by the NIH Brain Initiative project U01-NS122123 and the program "Ecoles Universitaires de Recherche" launched by the French Government and implemented by the ANR, with the reference ANR-17-EURE-0017. JWP was supported by grants from the Simons Collaboration on the Global Brain (SCGB AWD543027), the NIH BRAIN initiative (R01EB026946), and a U19 NIH-NINDS BRAIN Initiative Award (5U19NS104648).

References

[1] Mikio C Aoi, Valerio Mante, and Jonathan W Pillow. Prefrontal cortex exhibits multidimensional dynamic encoding during decision-making. *Nature neuroscience*, 23(11):1410–1420, 2020.

[2] Joao Barbosa, Remi Proville, Chris C Rodgers, Srdjan Ostojic, and Yves Boubenec. Flexible selection of task-relevant features through across-area population gating. *bioRxiv*, 2022.

[3] 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.

[4] György Buzsáki and Kenji Mizuseki. The log-dynamic brain: how skewed distributions affect network operations. *Nature Reviews Neuroscience*, 15(4):264–278, 2014.

[5] Warasinee Chaisangmongkon, Sruthi K Swaminathan, David J Freedman, and Xiao-Jing Wang. Computing by robust transience: how the fronto-parietal network performs sequential, category-based decisions. *Neuron*, 93(6):1504–1517, 2017.

[6] Zach Cohen, Brian DePasquale, Mikio C Aoi, and Jonathan W Pillow. Recurrent dynamics of prefrontal cortex during context-dependent decision-making. *bioRxiv*, 2020.

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

[8] Laura Driscoll, Krishna Shenoy, and David Sussillo. Flexible multitask computation in recurrent networks utilizes shared dynamical motifs. *bioRxiv*, 2022.

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

[10] Lea Duncker, Gergo Bohner, Julien Boussard, and Maneesh Sahani. Learning interpretable continuous-time models of latent stochastic dynamical systems. In *International Conference on Machine Learning*, pages 1726–1734. PMLR, 2019.

[11] 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.

[12] Daniel Durstewitz. A state space approach for piecewise-linear recurrent neural networks for identifying computational dynamics from neural measurements. *PLoS computational biology*, 13(6):e1005542, 2017.

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

[14] Arseny Finkelstein, Lorenzo Fontolan, Michael N Economo, Nuo Li, Sandro Romani, and Karel Svoboda. Attractor dynamics gate cortical information flow during decision-making. *Nature Neuroscience*, 24(6):843–850, 2021.

[15] Timo Flesch, Keno Juechems, Tsvetomira Dumbalska, Andrew Saxe, and Christopher Summerfield. Orthogonal representations for robust context-dependent task performance in brains and neural networks. *Neuron*, 2022.

[16] Aniruddh R Galgali, Maneesh Sahani, and Valerio Mante. Residual dynamics resolves recurrent contributions to neural computation. *bioRxiv*, 2021.

[17] Juan A Gallego, Matthew G Perich, Lee E Miller, and Sara A Solla. Neural manifolds for the control of movement. *Neuron*, 94(5):978–984, 2017.

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

[19] Joshua Glaser, Matthew Whiteway, John P Cunningham, Liam Paninski, and Scott Linderman. Recurrent switching dynamical systems models for multiple interacting neural populations. *Advances in neural information processing systems*, 33:14867–14878, 2020.

[20] Siddhartha Rao Kamalakara, Acyr Locatelli, Bharat Venkitesh, Jimmy Ba, Yarin Gal, and Aidan N Gomez. Exploring low rank training of deep neural networks. *arXiv preprint arXiv:2209.13569*, 2022.

[21] Kamesh Krishnamurthy, Tankut Can, and David J Schwab. Theory of gating in recurrent neural networks. *Physical Review X*, 12(1):011011, 2022.

[22] Christopher Langdon and Tatiana A Engel. Latent circuit inference from heterogeneous neural responses during cognitive tasks. *bioRxiv*, 2022.

[23] Frederic Lanore, N Alex Cayco-Gajic, Harsha Gurnani, Diccon Coyle, and R Angus Silver. Cerebellar granule cell axons support high-dimensional representations. *Nature Neuroscience*, 24(8):1142–1150, 2021.

[24] Scott Linderman, Matthew Johnson, Andrew Miller, Ryan Adams, David Blei, and Liam Paninski. Bayesian learning and inference in recurrent switching linear dynamical systems. In *Artificial Intelligence and Statistics*, pages 914–922. PMLR, 2017.

[25] Jakob H Macke, Lars Buesing, John P Cunningham, Byron M Yu, Krishna V Shenoy, and Maneesh Sahani. Empirical models of spiking in neural populations. *Advances in neural information processing systems*, 24, 2011.

[26] Niru Maheswaranathan and David Sussillo. How recurrent networks implement contextual processing in sentiment analysis. In *Proceedings of the 37th International Conference on Machine Learning*, pages 6608–6619, 2020.

[27] 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.

[28] 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.

[29] 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.

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

[31] Marcel Nonnenmacher, Srinivas C Turaga, and Jakob H Macke. Extracting low-dimensional dynamics from multiple large-scale neural population recordings by learning to predict correlations. *Advances in Neural Information Processing Systems*, 30, 2017.

[32] Marius Pachitariu, Biljana Petreska, and Maneesh Sahani. Recurrent linear models of simultaneously-recorded neural populations. *Advances in neural information processing systems*, 26, 2013.

[33] Chethan Pandarinath, Daniel J O'Shea, Jasmine Collins, Rafal Jozefowicz, Sergey D Stavisky, Jonathan C Kao, Eric M Trautmann, Matthew T Kaufman, Stephen I Ryu, Leigh R Hochberg, et al. Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature methods*, 15(10):805–815, 2018.

[34] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[35] 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, 2021.

[36] Biljana Petreska, Byron M Yu, John P Cunningham, Gopal Santhanam, Stephen Ryu, Krishna V Shenoy, and Maneesh Sahani. Dynamical segmentation of single trials from population neural data. *Advances in neural information processing systems*, 24, 2011.

[37] Eli Pollock and Mehrdad Jazayeri. Engineering recurrent neural networks from task-relevant manifolds and dynamics. *PLoS computational biology*, 16(8):e1008128, 2020.

[38] Kanaka Rajan, Christopher D Harvey, and David W Tank. Recurrent network models of sequence generation and memory. *Neuron*, 90(1):128–142, 2016.

[39] Evan D Remington, Devika Narain, Eghbal A Hosseini, and Mehrdad Jazayeri. Flexible sensorimotor computations through rapid reconfiguration of cortical dynamics. *Neuron*, 98(5):1005–1019, 2018.

[40] Omid G Sani, Hamidreza Abbaspourazad, Yan T Wong, Bijan Pesaran, and Maryam M Shanechi. Modeling behaviorally relevant neural dynamics enabled by preferential subspace identification. *Nature Neuroscience*, 24(1):140–149, 2021.

[41] Omid G Sani, Bijan Pesaran, and Maryam M Shanechi. Where is all the nonlinearity: flexible nonlinear modeling of behaviorally relevant neural dynamics using recurrent neural networks. *bioRxiv*, 2021.

[42] Britton A Sauerbrei, Jian-Zhong Guo, Jeremy D Cohen, Matteo Mischiati, Wendy Guo, Mayank Kabra, Nakul Verma, Brett Mensh, Kristin Branson, and Adam W Hantman. Cortical pattern generation during dexterous movement is input-driven. *Nature*, 577(7790):386–391, 2020.

[43] Shreya Saxena and John P Cunningham. Towards the neural population doctrine. *Current opinion in neurobiology*, 55:103–111, 2019.

[44] Rylan Schaeffer, Mikail Khona, Leenoy Meshulam, Ila Fiete, et al. Reverse-engineering recurrent neural network solutions to a hierarchical inference task for mice. *Advances in Neural Information Processing Systems*, 33:4584–4596, 2020.

[45] Marine Schimel, Ta-Chu Kao, Kristopher T Jensen, and Guillaume Hennequin. ilqr-vae: control-based learning of input-driven dynamics with applications to neural data. *bioRxiv*, 2021.

[46] 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.

[47] 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.

[48] João D Semedo, Amin Zandvakili, Christian K Machens, M Yu Byron, and Adam Kohn. Cortical areas interact through a communication subspace. *Neuron*, 102(1):249–259, 2019.

[49] Markus Siegel, Timothy J Buschman, and Earl K Miller. Cortical information flow during flexible sensorimotor decisions. *Science*, 348(6241):1352–1355, 2015.

[50] Jimmy Smith, Scott Linderman, and David Sussillo. Reverse engineering recurrent neural networks with jacobian switching linear dynamical systems. *Advances in Neural Information Processing Systems*, 34, 2021.

[51] Sen Song, Per Jesper Sjöström, Markus Reigl, Sacha Nelson, and Dmitri B Chklovskii. Highly nonrandom features of synaptic connectivity in local cortical circuits. *PLoS biology*, 3(3):e68, 2005.

[52] Nicholas A Steinmetz, Peter Zatka-Haas, Matteo Carandini, and Kenneth D Harris. Distributed coding of choice, action and engagement across the mouse brain. *Nature*, 576(7786):266–273, 2019.

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

[54] David Sussillo. Neural circuits as computational dynamical systems. *Current opinion in neurobiology*, 25:156–163, 2014.

[55] 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.

[56] David Sussillo, Mark M Churchland, Matthew T Kaufman, and Krishna V Shenoy. A neural network that finds a naturalistic solution for the production of muscle activity. *Nature neuroscience*, 18(7):1025–1033, 2015.

[57] Elia Turner, Kabir V Dabholkar, and Omri Barak. Charting and navigating the space of solutions for recurrent neural networks. *Advances in Neural Information Processing Systems*, 34:25320–25333, 2021.

[58] Adrian Valente, Srdjan Ostojic, and Jonathan W Pillow. Probing the relationship between latent linear dynamical systems and low-rank recurrent neural network models. *Neural computation*, 34(9):1871–1892, 2022.

[59] Ivan Voitov and Thomas D Mrsic-Flogel. Cortical feedback loops bind distributed representations of working memory. *Nature*, 608(7922):381–389, 2022.

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

[61] Jing Wang, Devika Narain, Eghbal A Hosseini, and Mehrdad Jazayeri. Flexible timing by temporal scaling of cortical responses. *Nature neuroscience*, 21(1):102–110, 2018.

[62] Yang Xie, Peiyao Hu, Junru Li, Jingwen Chen, Weibin Song, Xiao-Jing Wang, Tianming Yang, Stanislas Dehaene, Shiming Tang, Bin Min, et al. Geometry of sequence working memory in macaque prefrontal cortex. *Science*, 375(6581):632–639, 2022.

[63] Guangyu Robert Yang, Madhura R Joglekar, H Francis Song, William T Newsome, and Xiao-Jing Wang. Task representations in neural networks trained to perform many cognitive tasks. *Nature neuroscience*, 22(2):297–306, 2019.

[64] Guangyu Robert Yang and Xiao-Jing Wang. Artificial neural networks for neuroscientists: A primer. *Neuron*, 107(6):1048–1070, 2020.

[65] Byron M Yu, Afsheen Afshar, Gopal Santhanam, Stephen Ryu, Krishna V Shenoy, and Maneesh Sahani. Extracting dynamical structure embedded in neural activity. *Advances in neural information processing systems*, 18, 2005.

[66] Rafael Yuste. From the neuron doctrine to neural networks. *Nature reviews neuroscience*, 16(8):487–497, 2015.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] The three contributions exposed in the introduction are detailed in the results section. The specific scope is detailed in the introduction and the discussion.
   (b) Did you describe the limitations of your work? [Yes] Limitations are exposed in the discussion, notably with respect to hypotheses made about the inputs.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See discussion.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A] No novel theoretical results are exposed here, the theoretical results on low-rank RNNs already exist in the literature, notably in [3]
   (b) Did you include complete proofs of all theoretical results? [N/A] Same answer.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] Code will be made available upon publication.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] In the supplemental material, Appendix C
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Figure 2 contains whole distributions for the fitting experiments, standard deviation for the subsampling experiment, and the material exposed in section 3.3 and figure 5 has been fully reproduced on data from a second macaque subject. Moreover, additional hyperparameters are explored in sup. fig. 1.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] In the supplemental material, Appendix C
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] Providers of the dataset [29] cited in section 3.3
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No] Trained network weights will be provided upon publication.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] No human participants.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] No human participants not offensive content.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] No human participants.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] No human participants.
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] No human participants.

A Low-rank RNN dynamics

Here, we detail the theoretical link between connectivity and low-dimensional dynamics in low-rank RNNs, and outline its consequences for the main results. This section mostly summarizes the mathematical results of the papers [30], [3] and [9]. Sections A.3 and A.4 can be read without the preceding ones.

A.1 Low-dimensional dynamics

Let us first recall the formalism. We consider a network of N units, each characterized by an activation that follows the dynamics given in eq. (1) of the main text:

$$\tau \frac{dx_i}{dt} = -x_i + \sum_{j=1}^{N} J_{ij}\phi\left(x_j\right) + \sum_{s=1}^{N_{in}} I_i^{(s)} u_s(t) + \eta_i(t). \tag{7}$$

with a connectivity matrix of rank $R \ll N$, written as:

$$J_{ij} = \frac{1}{N}\sum_{r=1}^{R} m_i^{(r)} n_j^{(r)}. \tag{8}$$

where, to remove all degeneracy, we define vectors $\boldsymbol{m}^{(r)}$ as the left singular vectors of the connectivity matrix, and vectors $\boldsymbol{n}^{(r)}$ as the right singular vectors multiplied by the corresponding singular value (determined up to a change in sign). A consequence of this definition is that the $\boldsymbol{m}^{(r)}$ vectors are all orthogonal to each other, and the $\boldsymbol{n}^{(r)}$ vectors as well. As a further simplifying assumption we will also assume that the $\boldsymbol{I}^{(s)}$ vectors are also orthogonal to each other, and to each of the $\boldsymbol{m}^{(r)}$ vectors.

Injecting equation (8) into the differential equation (7) gives the vector dynamics:

$$\tau \frac{d\boldsymbol{x}}{dt} = -\boldsymbol{x}(t) + \sum_{r=1}^{R} \boldsymbol{m}^{(r)} {\boldsymbol{n}^{(r)}}^{\top} \phi(\boldsymbol{x}(t)) + \sum_{s=1}^{N_{in}} \boldsymbol{I}^{(s)} u_s(t) + \boldsymbol{\eta}(t), \tag{9}$$

where ϕ is applied in an elementwise manner. A direct consequence of (9) is is that the vector of neural activations $\boldsymbol{x}(t)$ is constrained to evolve in a subspace spanned by the vectors $\boldsymbol{m}^{(r)}$ and $\boldsymbol{I}^{(s)}$, of dimension $R + N_{in}$. Since these vectors form a basis of this subspace, one can decompose $\boldsymbol{x}(t)$ as:

$$\boldsymbol{x}(t) = \sum_{r=1}^{R} \kappa_r(t)\boldsymbol{m}^{(r)} + \sum_{s=1}^{N_{in}} v_s(t)\boldsymbol{I}^{(s)}, \tag{10}$$

where the $\kappa_r(t)$ are a set of recurrently generated latent variables, defined as the projection of $\boldsymbol{x}(t)$ on $\boldsymbol{m}^{(r)}$:

$$\kappa_r(t) := \frac{1}{\|\boldsymbol{m}^{(r)}\|^2}\sum_{j=1}^{N} m_j^{(r)} x_j(t), \tag{11}$$

and the $v_s(t)$ are low-pass filtered versions of the input signals $u_s(t)$, following:

$$\tau \frac{dv_s}{dt} = -v_s(t) + u_s(t). \tag{12}$$

The dynamics of the internal variables $\kappa_r(t)$ can also be described by a set of self-consistent differential equations, obtained by projecting equation (9) on each $\boldsymbol{m}^{(r)}$:

$$\tau \frac{d\kappa_r}{dt} = -\kappa_r(t) + \underbrace{\frac{1}{N}\sum_{j=1}^{N} n_j^{(r)}\phi(x_j(t))}_{:=\kappa_r^{rec}(t)}, \tag{13}$$

(where we have taken into account the orthogonality of each input vector $\boldsymbol{I}^{(s)}$ with the vectors $\boldsymbol{m}^{(r)}$ as a simplifying assumption). The second term of the r.h.s. of this equation, $\kappa_r^{rec}(t)$, which represents

the recurrent contribution to the input of r-th latent variable, can be expanded as:

$$\kappa_r^{rec}(t) = \frac{1}{N}\sum_{j=1}^{N} n_j^{(r)} \phi\left(\sum_{r'=1}^{R} \kappa_r'(t) m_j^{(r')} + \sum_{s=1}^{N_{in}} I_j^s v_s(t)\right), \tag{14}$$

which only depends on the set of $\kappa_r(t)$ and $v_s(t)$ functions, closing the system of equations.

Equations (13)-(14) show that a rank-R RNN is an R-dimensional dynamical system, which can be written as:

$$\frac{d}{dt}\boldsymbol{\kappa}(t) = F(\boldsymbol{\kappa}(t), \boldsymbol{u}(t)) \tag{15}$$

with $\boldsymbol{\kappa}(t)$ an R-dimensional state-space vector, and $\boldsymbol{u}(t)$ the N_{in}-dimensional vector of input signals. Here, F represents a non-linear map from $\mathbb{R}^{N_{in}+R}$ to $\mathbb{R}^R$ that depends exclusively on the parameters defining the network connectivity, that is the $\{I_i^{(s)}, m_i^{(r)}, n_i^{(r)}\}_{s,r,i}$. It can be shown that these parameters can be chosen in order to approximate any non-linear map F, so that the class of rank-R RNN are universal approximators for R-dimensional dynamical systems [3].

A.2 Mean-field approximation

Moreover, the map F can be more explicitly defined in the large network limit $N \to \infty$, if the connectivity parameters follow a particular distribution, namely a joint mixture-of-Gaussians distribution [3, 9]. Notably, in this framework, each neuron is characterized by its weights on the N_{in} input vectors and on the $2R$ recurrent connectivity vectors. Hence every neuron i can be seen as a point in a $(N_{in} + 2R)$-dimensional *connectivity space* of coordinates $(I_i^{(1)}, \ldots, I_i^{(N_{in})}, n_i^{(1)}, \ldots, n_i^{(R)}, m_i^{(1)}, \ldots, m_i^{(R)}) := (\underline{I}, \underline{n}, \underline{m})$.

We can then assume that these points are sampled from a probability distribution :

$$P(I^{(1)}, \ldots, I^{(N_{in})}, n^{(1)}, \ldots, n^{(R)}, m^{(1)}, \ldots, m^{(R)}) := P(\underline{I}, \underline{n}, \underline{m}). \tag{16}$$

We will take the assumption that the distribution $P(\cdot)$ is a mixture-of-Gaussians as in [3, 9]. It can then be written as:

$$P(\underline{n}, \underline{m}, \underline{I}, \underline{w}) := \sum_{p=1}^{P} \alpha_p \mathcal{N}\left(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p\right), \tag{17}$$

with p components to the mixture, each associated to the probability α_p, the mean $\boldsymbol{\mu}_p$ and the covariance matrix $\boldsymbol{\Sigma}_p$. We will add the simplifying assumption that all components are zero-centered, and vary just in terms of their covariances, so that $\boldsymbol{\mu}_p = \mathbf{0}$. Then, following a mean-field approximation, the dynamics in equation (13) can be simplified for each r to:

$$\frac{d\kappa_r}{dt} = -\kappa_r(t) + \sum_{r'=1}^{R} \tilde{\sigma}_{n^{(r)}m^{(r')}} \kappa_{r'}(t) + \sum_{s=1}^{N_{in}} \tilde{\sigma}_{n^{(r)}I^{(s)}} v_s(t). \tag{18}$$

where for any connectivity parameters a, b in $\{m^{(r)}, n^{(r)}, I^{(s)}\}$, $\tilde{\sigma}_{ab}$ represents an *effective coupling*, given by a weighted average over populations of the associated covariances:

$$\tilde{\sigma}_{ab} = \sum_{p=1}^{P} \alpha_p \sigma_{ab}^{(p)} \langle \Phi' \rangle_p, \tag{19}$$

where $\sigma_{ab}^{(p)}$ represent the covariance between parameters a and b in the p-th component (so that e.g. $\sigma_{n^{(1)}m^{(2)}}^{(p)} = \frac{1}{N}\sum_{i\in p} n_i^{(1)} m_i^{(2)}$), and $\langle \Phi' \rangle_p$ represents an average gain over neurons of the p-th component, defined as the average over these neurons of the derivative of the neural transfer

function $\phi'(x_i)$. Since these gains are distributed normally on each population, this average depends non-linearly on the κ_r and v_s values following equation:

$$\langle \Phi' \rangle_p = \langle \Phi' \rangle \left(\sqrt{\sum_{r'=1}^{R} (\sigma^{(p)}_{m^{(r')}})^2 \kappa^2_{r'} + \sum_{s=1}^{N_{in}} (\sigma^{(p)}_{I^{(s)}})^2 v_s^2} \right), \tag{20}$$

where $\langle \Phi' \rangle (\Delta)$ is the Gaussian integral:

$$\langle \Phi' \rangle (\Delta) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} \mathrm{d}z \, e^{-z^2/2} \phi'(\Delta z). \tag{21}$$

These equations finish to make the system given by equations (18) a closed system of non-linear coupled differential equations where the unknowns are the variables $\kappa_r(t)$.

A.3 Intuitive interpretation of the mean-field results

A more intuitive interpretation of those derivations is that equation (18) locally approximates the recurrent dynamics as a linear dynamical system (LDS), driven by the effective couplings between latent variables $\tilde{\sigma}_{ab}$ (which can vary depending on the state of the network $\boldsymbol{\kappa}(t)$ and on the inputs $\boldsymbol{u}(t)$). More specifically, the effective coupling from latent variable κ_r to latent variable $\kappa_{r'}$ (where possibly $r' = r$) is given by $\sigma_{n^{(r')}m^{r)}}$, and the effective coupling from input s to latent variable r by $\sigma_{n^{(r)}I^{(s)}}$.

These effective couplings can be modulated through equation (20) by inputs as well as the latent state of the network, leading to different dynamics in different parts of the neural state-space. In particular, inputs can have a purely modulating effect by modifying only the gain of specific population, hence modifying the effective couplings and the resulting network dynamics.

A.4 Notion of effective connectivity

A consequence of equation (18) is notably is that the low-dimensional dynamics depend on the exact entries on the $\boldsymbol{m}^{(r)}$ and $\boldsymbol{I}^{(s)}$ vectors, but not on the individual entries on the $\boldsymbol{n}^{(r)}$ vectors. Instead, $\boldsymbol{n}^{(r)}$ vectors influence the dynamics only through their covariances with the $\boldsymbol{m}^{(r)}$ and $\boldsymbol{I}^{(s)}$ vectors. Thus, all components of $\boldsymbol{n}^{(r)}$ orthogonal to the subspace spanned by $\boldsymbol{m}^{(r)}$ and $\boldsymbol{I}^{(s)}$ are irrelevant for the dynamics, and removing them leads to an effective connectivity matrix $\boldsymbol{J}^{\text{eff}}$ which captures the minimal features required for obtaining particular dynamics.

Thus, for a rank-R RNN, $\boldsymbol{J}^{\text{eff}}$ is computed as:

$$\boldsymbol{J}^{\text{eff}} = \frac{1}{N} \left(\sum_{r=1}^{R} \boldsymbol{m}^{(r)} \boldsymbol{n}_{\parallel}^{(r)\top} \right), \tag{22}$$

where each $\boldsymbol{n}_{\parallel}^{(r)}$ is defined as the orthogonal projection of $\boldsymbol{n}^{(r)}$ on the subspace spanned by the $\boldsymbol{m}^{(r)}$ and $\boldsymbol{I}^{(s)}$.

B Cognitive tasks

Here we describe the input and output structure for the four cognitive tasks used in this study. In all tasks we use the notations from section 2, considering N_{in} input signals $u_s(t)$ and one target output $z^*(t)$, that can be defined only at specific timepoints of a trial. Durations are given in an abstract time mapping, but tasks are implemented in a discretized time with timestep $dt = 20ms$

Decision-making task (DM). The network receives one input signal $u_s(t)$, equal to Gaussian white noise with standard deviation 0.1 (as for subsequent tasks), added to a mean coherence $\overline{u}$ drawn uniformly from $\pm 0.1 \times \{1, 2, 4\}$. The target output z^* is defined only at the final timestep of the trial, and is equal to the sign of the trial coherence. Each trial starts with a 100ms fixation period with no input, followed by an 800ms stimulus epoch, a 300ms delay epoch and a decision timestep.

Working memory task (WM). This task is inspired on the traditional parametric working memory and comparison experimental paradigm. The network receives one input signal, equal to:

$$u(t) = f_1\delta_1(t) + f_2\delta_2(t) + \xi(t)$$

where f_1 is randomly sampled in $[10, .., 34]$, $f_2 - f_1$ is randomly sampled in $\{-24, -16, -8, 8, 16, 24\}$ with the constraint that $10 \leq f_2 \leq 34$ and $\xi(t)$ is white Gaussian noise. The target output $z^*(t)$ is defined only at the final timestep and equal to the exact value $f_2 - f_1$. Each trial starts with a 100ms fixation period with no input, followed by a 100ms stimulus 1 epoch (where $\delta_1(t) = 1$), followed by a 500ms delay epoch, followed by a 100ms stimulus 2 epoch (where $\delta_2(t) = 1$) and a decision timestep.

Context-dependent decision-making (CDM). This task aims at modelling the experimental work of (Mante, Sussillo et al., 2013) [29]. The network receives four inputs, two noisy input signals $u_A(t)$ and $u_B(t)$ defined as for the DM task with independently drawn coherences and noise, and two contextual inputs $u_{ctxA}(t)$ and $u_{ctxB}(t)$ defined as a one-hot encoding of the trial context. The target output during the final task epoch is set to the sign of the coherence of the input indicated by the active contextual cue.

For the application of LINT to a full-rank network, the task was defined in five epochs: a 100ms fixation epoch with no inputs, a 350ms epoch with only contextual inputs, an 800ms stimulus epoch, with both noisy stimuli and contextual inputs, a 100 ms delay epoch and a 20ms decision epoch which is the only one where the target output $z^*(t)$ was defined.

For the application of LINT to electrophysiological recordings, four epochs were used: a 350ms epoch where only contextual inputs were active, a 650ms stimulation epoch, an 80ms delay epoch and a 20ms decision epoch (used for computing task accuracy on the fitted networks). The trained networks were constrained to reproduce neural activity only during the last three epochs. The coherences used in this part of the work were sampled from $\pm\{0.047, 0.15, 0.5\}$ for monkey A, $\pm\{0.07, 0.19, 0.54\}$ for monkey F.

Delay Match-to-Sample (DMS). This task reproduces a paradigm where two consecutive stimuli each either of type A or B are presented, and the subject distinguishes between matches (both stimuli of the same type) and non-matches (stimuli of different types). In our models, the stimuli are given through two input signals $u_A(t)$ and $u_B(t)$, with Gaussian white noise centered around 0 or 1 while the corresponding stimulus is active. The task comprises five epochs, a fixation epoch of duration 100ms, a first stimulation epoch of 500ms, a delay epoch of variable duration between 500 and 3000ms, a second stimulation epoch of 500ms and a decision epoch of 1000ms. During each stimulation epoch a single type between A and B is sampled and the corresponding input signal is set to a mean of 1. The target output $z^*(t)$ during the decision epoch is equal to 1 for a match, -1 for a non-match.

C Training details and hyperparameters.

Task-optimized lrRNNs. The full-rank RNNs were defined following a discretized version of equation (1):

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \frac{\Delta t}{\tau}\left(-\boldsymbol{x}_t + \boldsymbol{J}\phi(\boldsymbol{x}_t) + \sum_{s=1}^{N_{in}} \boldsymbol{I}^{(s)} u_{s,t} + \boldsymbol{\eta}_t\right), \tag{23}$$

with $\boldsymbol{\eta}_t$ a random normal vector with independent entries and standard deviation 0.05 on each entry, and $\Delta t = 20ms$, $\tau = 100ms$. All networks were defined in pytorch [34] and trained using the ADAM optimizer. We trained networks on 800 random trials, for a certain number of epochs being divided in 25 batches of 32 trials. We considered networks with $N = 512$ units for this part of the work.

For the low-rank RNNs (section 3.1), we trained the $\boldsymbol{m}^{(r)}$ and $\boldsymbol{n}^{(r)}$ vectors. For each task, we found the minimal rank by training networks of increasing rank until they performed the task with more than 95% accuracy. For the DM and CDM tasks, the weights were all initialized following a random Gaussian distribution of standard deviation 1, and 4 for the readout weights w_i. For the WM task,

the rank-2 networks were initialized from the SVD of the connectivity matrix of a full-rank RNN previously trained on the task. For the DMS task, this initialization trick was also used, as well as the following shaping procedure: rank-2 networks were first trained on trials with a maximal delay of 700ms, then 1000ms, and finally 4000ms. The learning rates used were of 0.01 for all low-rank networks, and on the order of 10^{-4} for full-rank networks.

For the full-rank RNNs (section 3.2), we trained the connectivity matrix $\boldsymbol{J}$. The initial weights were sampled from centered Gaussian distributions with a standard deviation of 1 for input and readout weights, and $\rho/\sqrt{N}$ for connectivity coefficients J_{ij}. The values of ρ used in the main text were 0.1 for the CDM task, and 0.8 for the DMS task, with results for other values displayed in sup. fig. 1.

LINT on data from task-optimized lrRNNs. For the application of LINT to synthetic data generated from the task-optimized lrRNNs, we generated 800 trials for each task, simulated the trajectories displayed by the task-optimized networks on these trials, and trained identical RNNs, all initialized with random weights, to reproduce these trajectories. The networks were then tested on 800 newly sampled trajectories to obtain the reported values of R^2. Given trials $k \in \{1, \dots, K\}$, timesteps $t \in \{1, \dots, T\}$ and neurons $i \in \{1, \dots, N\}$, the global R^2 is computed as:

$$R^2 = 1 - \frac{\sum_k \sum_t \sum_i \left(x_i^{(k)}(t) - \tilde{x}_i^{(k)}(t) \right)^2}{\sum_k \sum_t \sum_i \left(\tilde{x}_i^{(k)}(t) - \langle \tilde{x}_i^{(k)}(t) \rangle_{k,t,i} \right)^2} \tag{24}$$

following the notations of section 2.

LINT on data from task-optimized vanilla RNNs. For this application of LINT, synthetic data was generated from task-optimized full-rank RNNs by simulating their responses to 800 trials for each of the two tasks. Inferred low-rank networks were trained for 500 epochs on those trials, and then tested on 800 newly generated trials. Reported R^2 values were computed with equation (24).

LINT on electrophysiological data. Low-rank networks trained to reproduce trajectories were trained on a set of 64 random conditions out of 72, and tested on the remaining 8 conditions to compute reported R^2. To compute the task accuracy, a linear decoder $\boldsymbol{w}$ was trained on the decision epoch of the task to report the correct choice on all 72 conditions, and the obtained accuracy was reported.

D Context-dependent decision-making mechanism

Here we complete the explanations of the main text detailing the mechanisms by which the full-rank network performs the CDM task.

From a dynamical point of view, low-dimensional visualizations have shown that the neural activity first moves along input-driven directions, before being slowly integrated along a choice axis (fig. 3 and sup. fig. 2). The rank-1 inferred network illuminates this behavior in terms of input and recurrent connectivity: the input-driven direction (TDR stimulus A, stimulus B and context axes) correspond to input vectors of the network, directly inserting the stimulus signals into the network. Choice, however, is generated through the recurrent connections, more specifically through a rank-1 feature of these connections: features that have to be integrated are selected by the $\boldsymbol{n}$ vector (which we call the *input-selection vector* [9]), and are projected onto the $\boldsymbol{m}$ vector, which then encodes choice. Hence, overlaps between the signal input vectors $\boldsymbol{I}^A$ and $\boldsymbol{I}^B$ and the input-selection vector drive the integration of input signal into this recurrent loop.

However, the network selects in a context-dependent manner which of the signals is integrated into the loop. We demonstrate in the main-text how this context-dependent selection relies on two separate populations of neurons. More specifically, these two groups of neurons have different weights on the contextual input vectors ($\boldsymbol{I}^{ctxA}$ and $\boldsymbol{I}^{ctxB}$) which drive them towards different gain regimes: for example, the neurons with the strongest weights on the $\boldsymbol{I}^{ctxA}$ vector are driven towards the saturating parts of their non-linear transfer function in context A (fig. 4b). It happens that these neurons exhibit a positive correlation between their entries on $\boldsymbol{I}^B$ and $\boldsymbol{n}$, necessary for integrating the input B signal into the recurrent dynamics. While these neurons see their gain decreased, the effective overlap between vectors $\boldsymbol{I}^A$ and $\boldsymbol{n}$ as described in equation (19) is decreased, and hence input B cannot be integrated anymore. Through this gain-modulation mechanism, the network is able

to ignore the irrelevant signal during context A. The opposite mechanism naturally happens with another population during context B.

This mechanism has a very consequence for ablation experiments: we have shown in fig. 4c how inactivating specifically the green and purple population decrease performance only in one context at every time. A more detailed picture appears when we look at the specific psychometric matrices of the perturbed networks (sup. fig. 2e). For example, when inactivating population B (green), the network actually keeps integrating input A even in context B: it is able to perform only the context A task. The opposite error pattern appears when population A (purple) is inactivated. These error patterns are exactly retrieved in the full-rank networks, showing that the same neurons have an exactly similar role, even though it can be directly retrieved from looking at the full-rank connectivity.

E Delay Match-to-Sample full-rank network reverse-engineering

To display another example of how LINT can be exploited to reverse-engineer mechanisms used by a "black-box" full-rank RNN, we apply it to the networks trained on another task. Here we consider the DMS task, where the network receives two consecutive stimuli chosen among two possible classes A and B, separated by a delay period, and has to output during its response period a positive value if the stimuli were of the same class ("match"), a negative value otherwise ("non-match", see full details in appendix B).

When training full-rank RNNs on this task, it appeared that they could be well approximated by lrRNNs of a rank usually equal to two (fig. 2), irrespective of their training hyperparameters (fig. 1). Here, we focus on a full-rank RNN trained with an initial connectivity of standard deviation equal to $0.8/\sqrt{N}$ and $N = 512$ units, which was well approximated by a rank-2 lrRNN (R^2 fit qualities for individual neurons shown in sup. fig. 3a). Indeed, this network, although trained without any constraint on its connectivity, exhibited low-dimensional trajectories as can be seen by performing a PCA (sup. fig. 3b). An idea of how neural geometry enables the networks to perform the task can be obtained by observing projections of the trajectories on spaces spanned by the first principal components. Typically, observing projections on the top 3 components (sup. fig. 3c), it becomes apparent that the network relies on four fixed points, one for remembering that the first stimulus is A during the delay period (middle right), one for remembering that the first stimulus is B (at the left), also used to indicate a match, one to indicate a non-match (at the right) and finally one to indicate only a match A-A (at the right). Stimulus inputs seem to drive activity through transient tunnels from one fixed point to another. This picture provides a certain grasp of phenomena happening in the network, but does not illuminate how dynamics enable activity to correctly jump between fixed points, nor how connectivity enables this dynamical picture to emerge.

The rank-2 network inferred by LINT has its activity constrained by design to a four-dimensional subspace of the neural state-space, spanned by the two input vectors of the network $\boldsymbol{I}^A$ and $\boldsymbol{I}^B$ as well as the two output vectors of the recurrent connectivity $\boldsymbol{m}^{(1)}$ and $\boldsymbol{m}^{(2)}$. These four vectors indeed have a significant overlap with the top four principal components (sup. fig. 3d), with the difference that they disentangle input from recurrent contributions to neural activity. The rank-2 connectivity is also characterized by two input-selection vectors of the recurrent connectivity $\boldsymbol{n}^{(1)}$ and $\boldsymbol{n}^{(2)}$, which will be a key driver of the network dynamics, as well as a task readout $\boldsymbol{w}$.

Projecting the neural trajectories on the subspace spanned by the two output recurrent vectors shows indeed that the activity of the rank-2 network reproduces well that of the original network, and that both networks rely on four fixed points to perform the task in the manner outlined above (sup. fig. 3e). In the absence of inputs, the neural dynamics of the rank-2 network stay confined to the two-dimensional $\boldsymbol{m}^{(1)}$-$\boldsymbol{m}^{(2)}$ space, so that the full autonomous dynamics can be visualized as a vector field, illuminating the dynamical inner workings of such networks (sup. fig. 3f). This reveals the existence of the four stable fixed points in the dynamical landscape, as well as an unstable fixed point at the origin and four saddle points. Moreover, while tonic inputs are received, which is the case during each stimulation period, the dynamics shift to a two-dimensional affine subspace of the neural state space, parallel to the $\boldsymbol{m}^{(1)}$-$\boldsymbol{m}^{(2)}$ plane but shifted along the received input vector. The dynamics while a tonic input is received can thus also be visualized as a two-dimensional vector field (sup. fig. 3g), explaining how inputs drive transitions from one fixed point to another. More specifically, only the two stable fixed point on the lower part of the plane are kept during a stimulation by input A, while dynamics have a slight clockwise rotational component. Hence, if input A is

received during a short stimulation period, they are driven towards the lower right part of the plane, where they can stay in a fixed point during the delay period (red and orange trajectories), and if a second input A is received, they will be driven towards the lower left fixed point. Conversely, under input B stimulation, only the two fixed points on the top of the plane are kept, with a very slight counterclockwise rotational component on this plane (at least under the vicinity of the origin, although not very visible in the plotted field). Hence, if stimulus B is received first, the network will be driven to the top left fixed point. If a second stimulus B is received, the network will stay in that state, whereas if a second input A is received it will be driven towards the top right fixed point (non-match).

These full dynamical landscapes provide us with a step-by-step decomposition of the task trials, and can also lead to predictions for trials taht do not appear in the task (for example we could predict the network behavior if three stimulations are received, or with longer or shorter stimulations). Moreover, the rank-2 connectivity can illuminate how this dynamical behavior emerges from the learned synaptic weights [3, 9]. A detailed account would go beyond the scope of this text, but some insights can be extracted easily: first, looking at the distribution of the weights of all neurons on vectors $\boldsymbol{m}^{(1)}$ and $\boldsymbol{m}^{(2)}$ (sup. fig. 3h), it appears that they separate in four clusters, each scattered around a different mean forming a quadrilateral on this plane. This type of population structure has been shown to enable the apparition of polygons of stable fixed points in the neural state space (see notably [3]). The way in which inputs modify dynamics can also be explained by connectivity features: more specifically, inputs A and B seem to modify the gains of different groups of neurons, as was the case for the contextual cues in the CDM task. Importantly, the neurons that are driven to a low-gain regime by input A (defined as the set of neurons i such that $|I_i^A| > 1$) are characterized by a negative correlation between vectors $\boldsymbol{n}^{(1)}$ and $\boldsymbol{m}^{(2)}$, and neurons driven to a high-gain regime by the same input exhibit a different correlation between these two vectors. Due to this differential distribution, while input A is received the effective coupling between the two latent variables κ_1 and κ_2 is modified. A converse situation happens with input B, explaining how input A generates this slight clockwise rotational component in the dynamics, and input B generates an opposite rotation. These local rotations, coupled with correlations between inputs and the recurrent vectors lead to the apparition of the plotted dynamical landscapes. These analyses could be verified by examining gains of neurons and performing ablation studies in the full-rank network, as has been done for the CDM task.

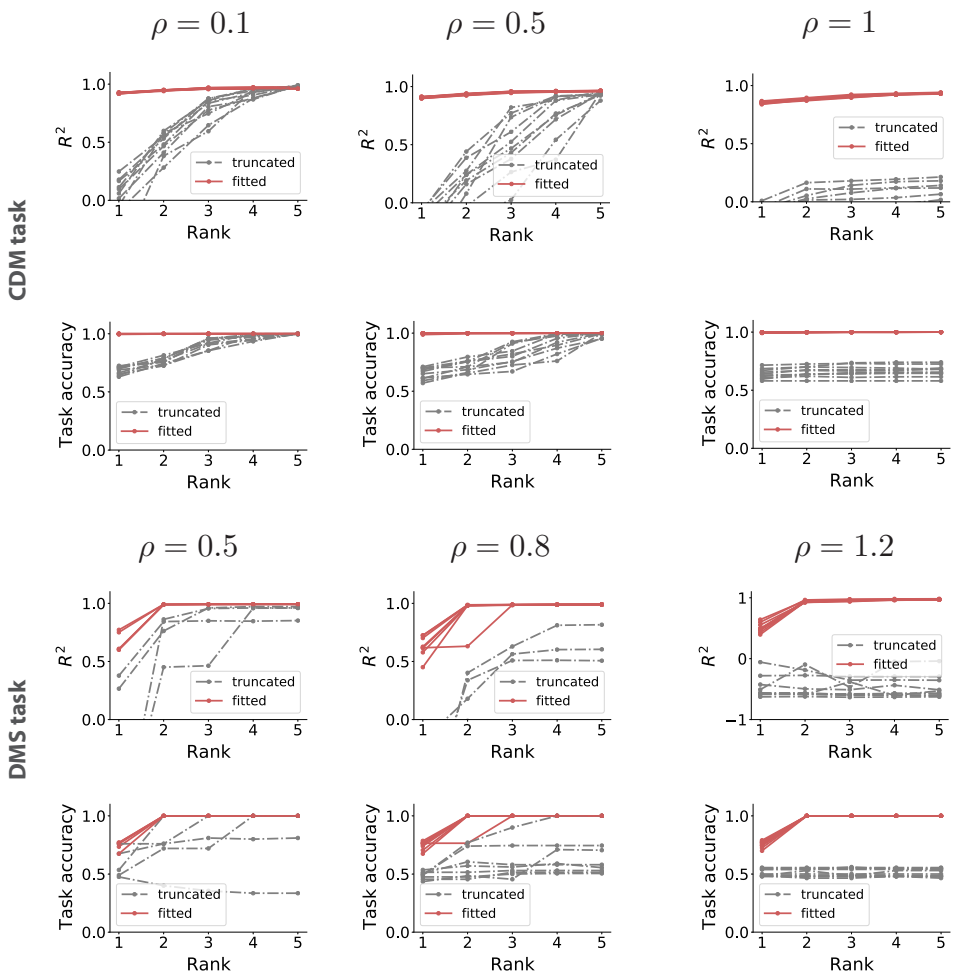
F Experiments on K-back tasks

To probe the capabilities of the LINT method on tasks that by design require higher-dimensional dynamics, we implemented it on the K-back task, inspired from classical paradigms of sequential working memory tasks [62]. In this task, for a certain fixed K, networks received a random number $P \geq K$ of stimulations in $\{-1, +1\}$ and had to output the K-th last stimulation received. In particular, for $K = 1$, this task corresponds to the classic running-memory flip-flop task [55]. Stimulations were received through a single input signal $u(t)$ transmitted by an input vector $\boldsymbol{I}$ and outputs were obtained through a linear readout following eqs. (1) and (6). Each stimulation lasted for 50 ms, and was separated from the next one, or the response period, by a 50 ms delay. We implemented the task following the methods outlined in appendix C, with 512-neuron networks. P was capped at 12.

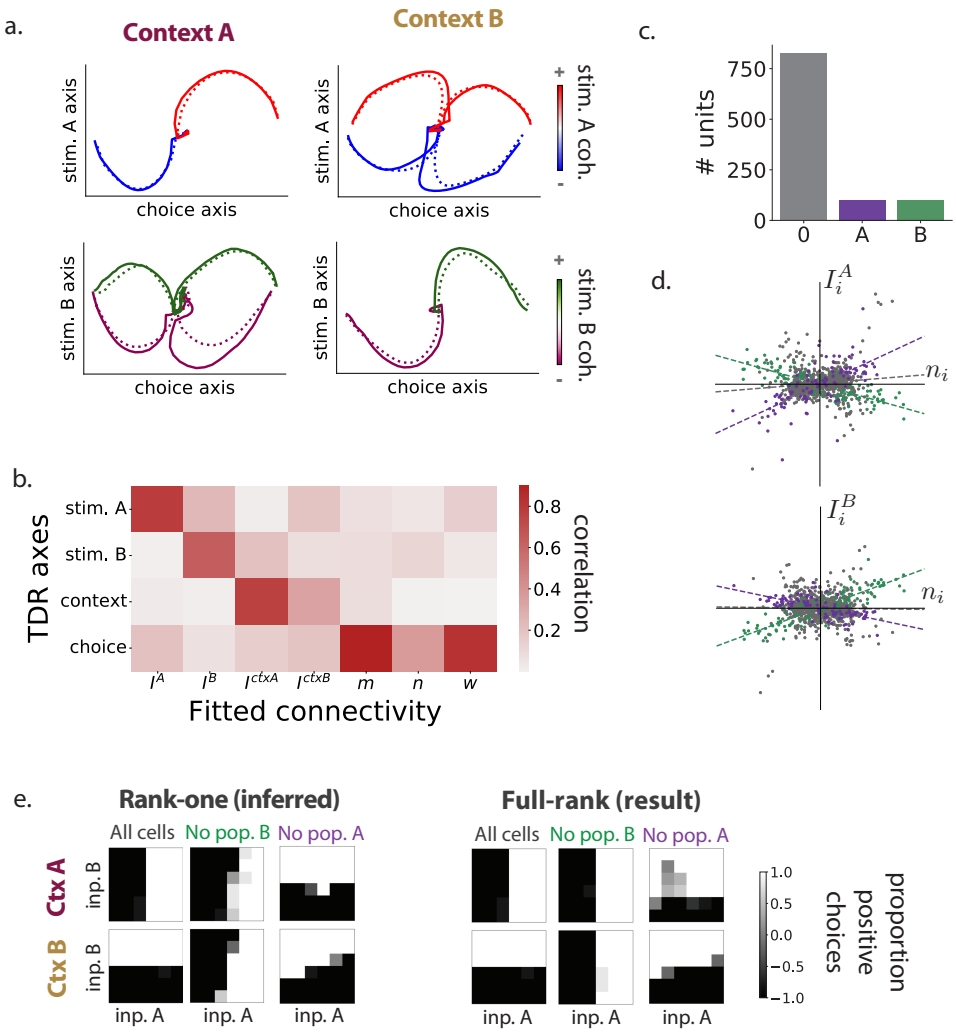
This task requires the networks to implement an internal memory with a capacity of at least K bits, it is thus reasonable to believe that a rank at least K is necessary to perform it. We first directly sought to validate these intuitions by training networks of increasing ranks to behaviorally perform the task, for K between 2 and 6. We observed that indeed the rank increased as K did, and that a rank equal to K seemed to be sufficient to implement the K-back task (sup. fig. 9a).

We next probed whether LINT could be applied to full-rank networks trained on the K-back task, despite the higher dimensionality. We thus trained full-rank networks on the task, for K from 2 to 6 (and an initial connectivity of standard deviation $\rho/\sqrt{N}$ for $\rho = 0.8$), and fitted their trajectories with low-rank networks of increasing ranks. Again, we found that full-rank networks trained on the K-back task appeared to be imitated well by low-rank networks with a rank as low as possible, i.e. K, both in terms of trajectories and behavior (sup. fig. 9).

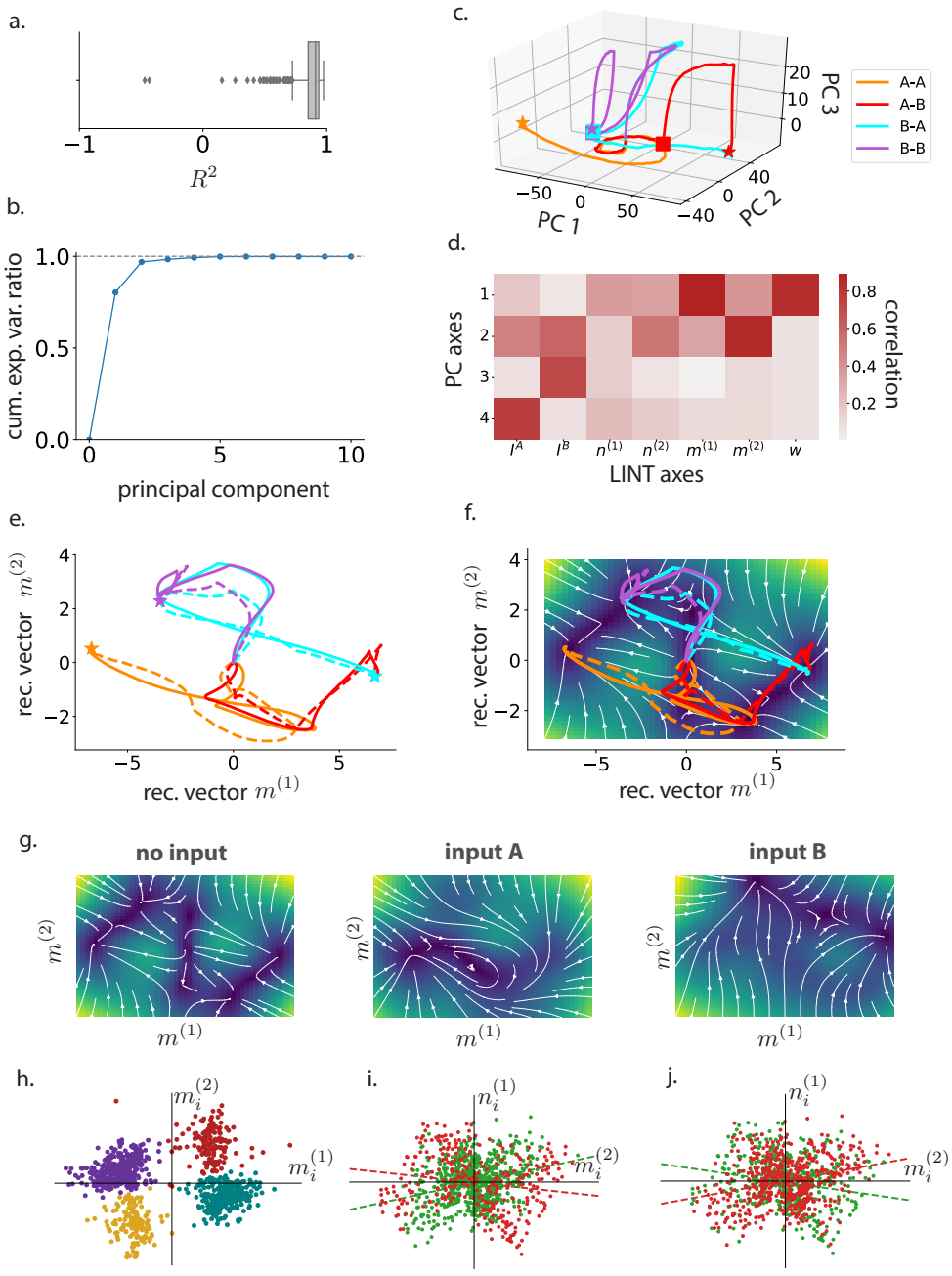
G Supplementary figures



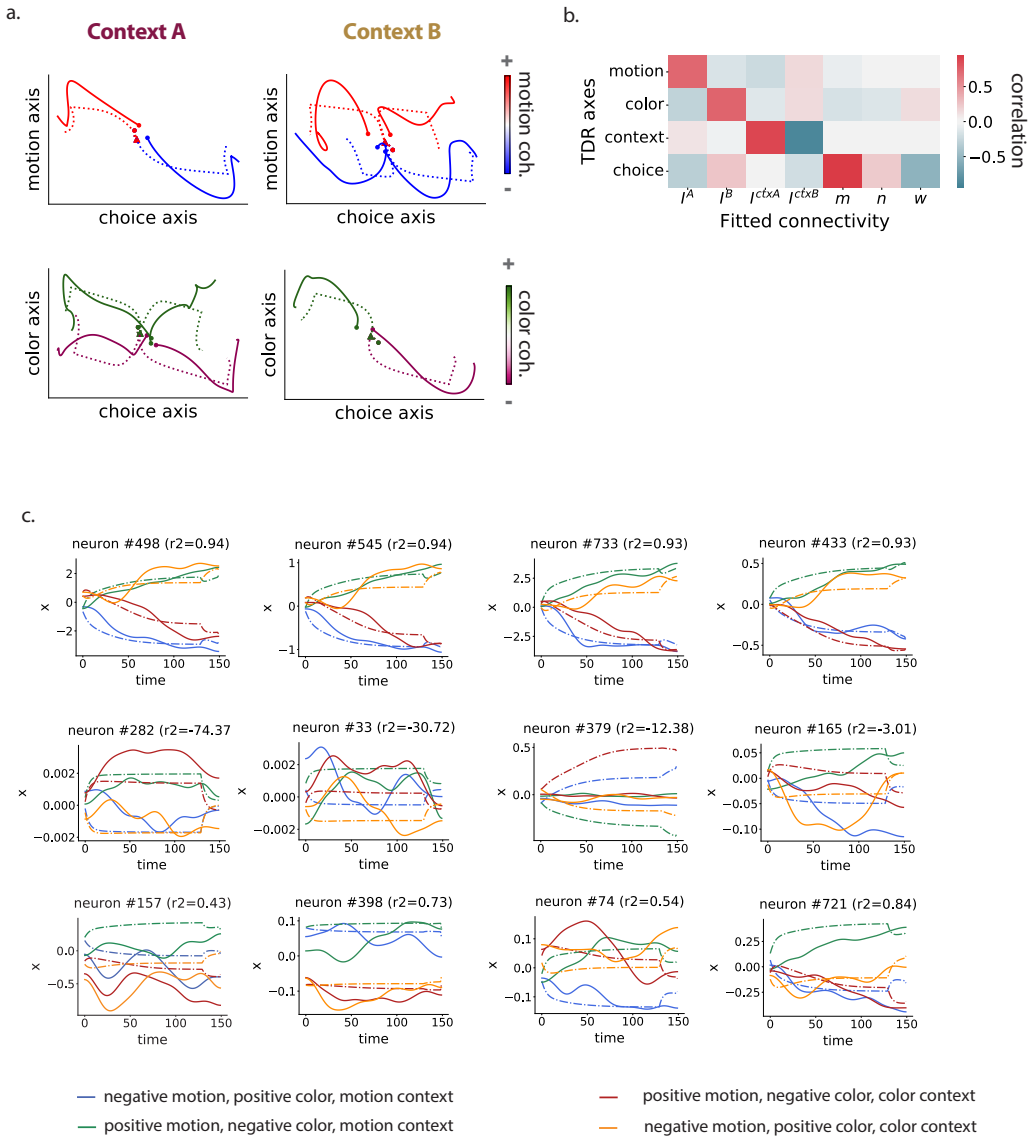
Supplementary Figure 1: For different values of ρ, where the standard deviation of initial recurrent weights is $\rho/\sqrt{N}$, we train 10 unconstrained networks, and either fit low-rank networks to their trajectories with increasing ranks or truncate their connectivity matrices to the same rank. The obtained R^2 similarity between original and fitted trajectories and task accuracy when plugging low-rank networks to the original readout are illustrated with a different line for each original unconstrained network. Top 2 rows: experiments on Context-dependent decision making task. Bottow 2 rows: experiments on the Delay Match-to-Sample task. Note that with higher initial random recurrent weights, unconstrained networks tend to go to what has been termed as the "lazy" training regime, with potentially higher dimensional trajectories with respect to the the "rich" training regimes for smaller initial weights [15]. This is visible through the poor performance of truncated connectivity matrices, but does not harm the effectiveness of our method.



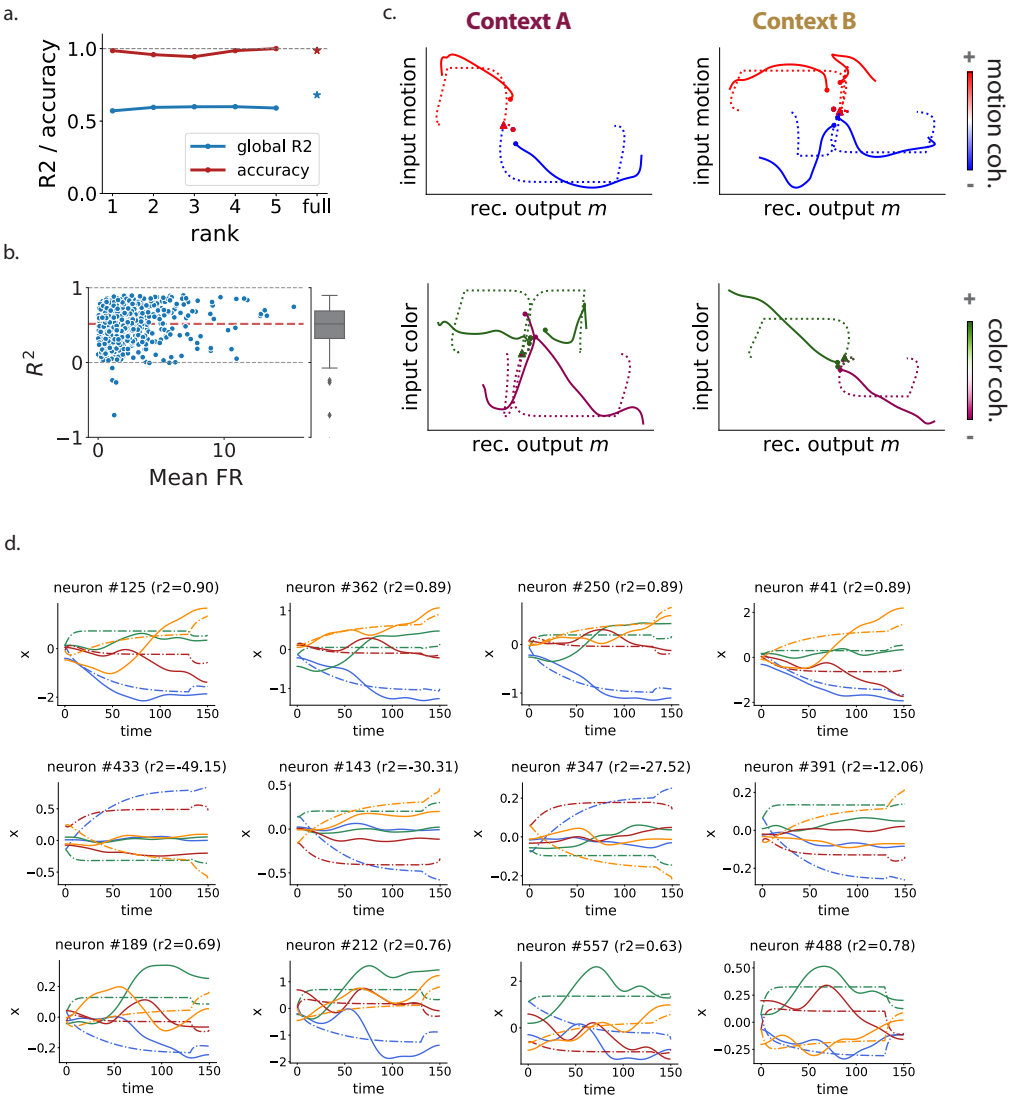
Supplementary Figure 2: Additional figures on LINT applied to a full-rank network performing the CDM task (exhibited in figs. 3 and 4). **a**. Low-dimensional projections of trial-averaged population trajectories for several combinations of context, stimuli A and B and choice, as in fig. 3 in the original full-rank network (full lines) and the inferred rank-1 network (dashed lines), projected on axes found by targeted dimensionality reduction (TDR) [29] applied to the full-rank network. **b**. Correlation between the four axes found by TDR on the full-rank network and the connectivity axes inferred by LINT. **c**. Number of units assigned to each of the three populations used for the reverse-engineering. Here, we manually defined population A as the 100 units with the strongest absolute context A input weight in the rank-1 network (see fig. 3a top), and population B as the 100 units with the strongest context B input weight not in population A. Applying Bayesian GMM clustering with 3 components and a strong mean precision prior gives very similar results. **d**. For the inferred rank-1 network, joint distributions of connectivity weights on the input vector $\boldsymbol{I}^A$ and $\boldsymbol{n}$, as well as on $\boldsymbol{I}^B$ and $\boldsymbol{n}$. For each population, linear regression lines are plotted. **e**. Psychometric response matrices in each context for all combination of stimulus coherences, for the inferred and original network when they are left unperturbed or after lesioning populations A and B. Unperturbed matrices indicate expected behavior. One can observe that inactivation of population B leads the networks to always behave as if in context A (losing its capacity to perform in context B), whereas the opposite phenomenon happens when population A is inactivated.



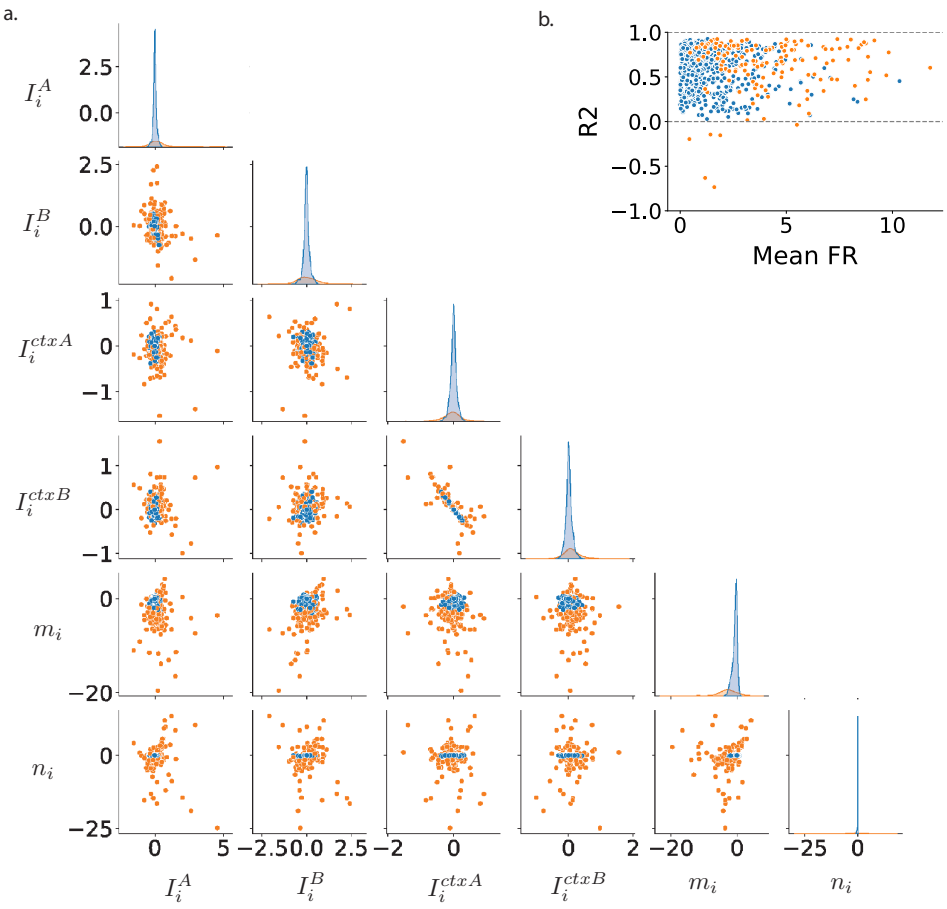
Supplementary Figure 3: LINT applied to a full-rank RNN trained on the DMS task. A rank-2 network was inferred and is analyzed in this figure (see Appendix E for details). **a**. Boxplot representing the distribution of R^2 fitting values for individual neurons. **b**. Cumulative explained variance ratio for top 10 principal components of a PCA applied to trajectories of the full-rank network. **c**. Trajectories in the four possible task conditions in the original full-rank network, projected on the top 3 principal components (squares: delay period, stars: end of trial). **d**. Correlation between axes inferred by a PCA on the full-rank trajectories and connectivity axes of the inferred rank-2 network. **e**. Trajectories of the full-rank network (full lines) and the rank-2 model (dashed lines) in the four task conditions, projected on the two recurrent connectivity output vectors $\boldsymbol{m}^{(1)}$ and $\boldsymbol{m}^{(2)}$ (same colors as in c). **f**. Same trajectories, superposed on the vector field representing autonomous dynamics in the rank-2 RNN. Colors indicate speed of the dynamics (blue: slow, yellow:fast). **g**. Vector fields representing the dynamics on the $\boldsymbol{m}^{(1)}$-$\boldsymbol{m}^{(2)}$ plane. **h-j**. Connectivity parameter distributions on the rank-2 models... **h**. on the two recurrent output vectors $m^{(1)}$ and $m^{(2)}$ - four populations can be identified by GMM clustering. **i**. on the connectivity vectors $m^{(2)}$ and $n^{(1)}$ - low-gain neurons while input A is received in red, others in green with overlaid linear regressions for both groups. **j**. on the same vectors, with low-gain neurons while input B is received in red, others in green.



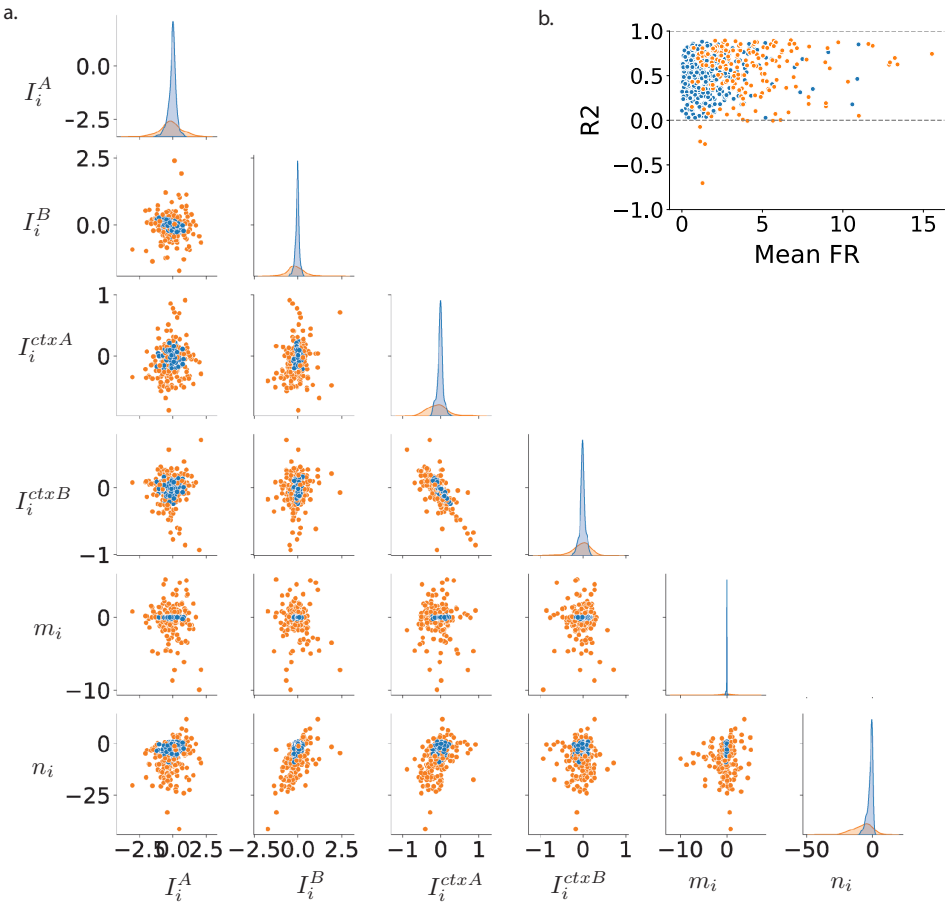
Supplementary Figure 4: Additional illustrations of LINT applied to electrophysiological recordings in monkey A. **a**. Two-dimensional projections of trial-averaged population trajectories for several combinations of context, choice, and motion or color coherence (indicated by the color code), as in fig. 5c, in the recorded data (full lines) and the rank-1 model (dashed lines), projected on axes inferred by TDR. **b**. Correlation coefficients between axes identified by TDR and LINT connectivity axes. **c**. Pre-processed data responses and rank-1 model responses for individual neurons to 4 different task conditions (uniquely identified by a context, a color coherence, and a motion coherence. Strongest coherences displayed here). Top row: four best fitted neurons. Middle row: four worse fitted neurons. Bottom row: four randomly selected neurons.



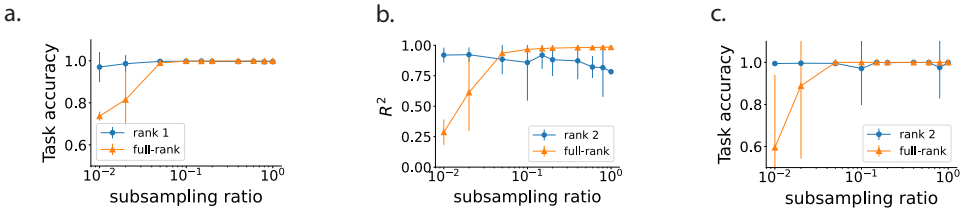
Supplementary Figure 5: LINT applied to electrophysiological recordings of a second monkey (monkey F) performing the same task. **a-c**. Same as fig. 5 for monkey F data. For panel b, 13 neurons for which $R^2 < -1$ do not appear, all having a mean firing rate of less than 3.8 Hz. **d**. Same as sup. fig. 4 for monkey F, with the 4 best fitted neurons, the 4 worse fitted neurons, and 4 randomly selected neurons.



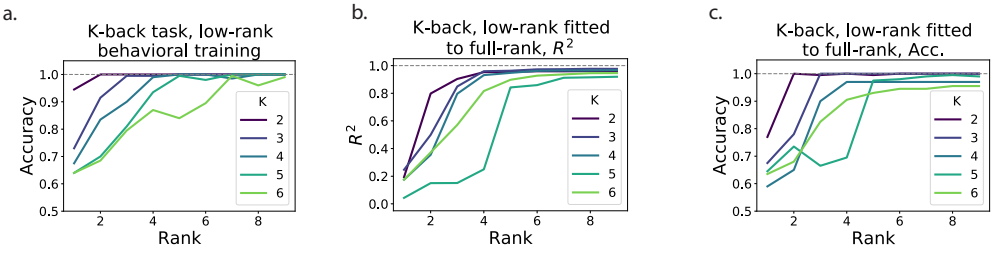
Supplementary Figure 6: Distribution of learned connectivity parameters for monkey A. **a**. Full six-dimensional distribution of the inferred weights on the input and recurrent connectivity vectors of the rank-1 model, plotted through two-dimensional and one-dimensional marginals. GMM clustering can identify two groups of neurons: large-weight neurons (orange) and small-weight neurons (blue). Inactivating the blue population does not affect task performance of the network. **b**. Same clusters visualized in the mean firing-rate - R^2 point cloud (fig. 5a).



Supplementary Figure 7: Same as sup. fig. 6



Supplementary Figure 8: Additional results on subsampling experiments. **a**. Task accuracy for the same fitted networks visualized in fig. 2f. **b-c**. Similar experiment as that described in fig. 2f, this time where trajectories were generated from a full-rank network trained to perform the DMS task, and respectively full-rank and rank-2 networks were fitted to random subsamples of neurons of the original network. Error bars: mean $\pm$ std over 10 random subsamples for each ratio value.



Supplementary Figure 9: Networks trained on the K-back task (see appendix F). **a**. Final accuracy of low-rank networks of increasing rank trained to perform the task, for K between 2 and 6. Notice that a rank K seems necessary and sufficient for the K-back task. **b**. R^2 values for the trajectories of full-rank networks trained on the K-back task, and networks of increasing rank trained to reproduce their trajectories, for K between 2 and 6. **c**. Final task accuracy for the same low-rank networks.