Parametric control of flexible timing through low-dimensional neural manifolds

Manuel Beiran[1], **Nicolas Meirhaeghe**[2], **Hansem Sohn**[3], **Mehrdad Jazayeri**[3,4,*], **Srdjan Ostojic**[1,*]

[1] Laboratoire de Neurosciences Cognitives et Computationnelles, INSERM U960, Ecole Normale Superieure - PSL University, 75005 Paris, France.
[2] Harvard-MIT Division of Health Sciences and Technology, Massachusetts Institute of Technology, Cambridge, MA 02139, USA.
[3] McGovern Institute for Brain Research, Massachusetts Institute of Technology, Cambridge, MA 02139, USA.
[4] Department of Brain & Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, MA 02139, USA.
* corresponding authors

Abstract

Biological brains possess an unparalleled ability to generalize adaptive behavioral responses from only a few examples. How neural processes enable this capacity to extrapolate is a fundamental open question. A prominent but underexplored hypothesis suggests that generalization is facilitated by a low-dimensional organization of collective neural activity. Here we tested this hypothesis in the framework of flexible timing tasks where dynamics play a key role. Examining trained recurrent neural networks we found that confining the dynamics to a low-dimensional subspace allowed tonic inputs to parametrically control the overall input-output transform and enabled smooth extrapolation to inputs well beyond the training range. Reverse-engineering and theoretical analyses demonstrated that this parametric control of extrapolation relies on a mechanism where tonic inputs modulate the dynamics along non-linear manifolds in activity space while preserving their geometry. Comparisons with neural data from behaving monkeys confirmed the geometric and dynamical signatures of this mechanism.

1 Introduction

Humans and other animals exhibit a prominent cognitive ability to infer relevant behavioral responses to a wide range of stimuli from only a handful of examples (Markman, 1989; Lake et al., 2015). This capacity to generalize and extrapolate well beyond the training set has been particularly challenging to realize in artificial systems (Lake et al., 2017; Sinz et al., 2019; Saxe et al., 2021), raising the question of what biological mechanisms might enable it. Theoretical analyses have argued that the dimensionality of neural activity is a key parameter, with high dimensionality promoting stimulus discrimination, and low dimensionality instead facilitating generalization (DiCarlo and Cox, 2007; DiCarlo et al., 2012; Rigotti et al., 2013; Fusi et al., 2016; Chung et al., 2018; Cayco-Gajic and Silver, 2019; Bernardi et al., 2020; Badre et al., 2021; Nogueira et al., 2021; Flesch et al., 2021). The extent to which these early findings extend to more complex cognitive tasks remains an open question, especially when temporal dynamics are involved (Laje and Buonomano, 2013; Cueva et al., 2020; Bi and Zhou, 2020). While low-dimensional organization of recorded neural activity has emerged as a ubiquitous experimental observation across a large variety of behavioral paradigms (Gao et al., 2015; Gallego et al., 2017; Jazayeri and Ostojic, 2021), and in particular in flexible timing tasks that rely on temporal dynamics of activity (Mello et al., 2015; Merchant and Averbeck, 2017; Wang et al., 2018; Remington et al., 2018b; Gámez et al., 2019; Sohn et al., 2019; Egger et al., 2019; Meirhaeghe et al., 2021), its computational role within this setting remains to be fully established. In this study we therefore set out to examine the hypothesis that embedding neural activity in a low-dimensional subspace enables neural networks to extrapolate to previously unseen stimuli within flexible timing tasks.

This hypothesis can be naturally formulated within the framework of dynamical systems (Remington et al., 2018a; Vyas et al., 2020) in which inputs act as control signals that provide either explicit instructions (Wang et al., 2018), sensory measurements (Sohn et al., 2019; Egger et al., 2019) or contextual information (Remington et al., 2018b; Sohn et al., 2019). This framework has been particularly successful in models of interval timing. For example, it has been shown that tonic inputs instructing a network to produce different time intervals can be construed as control signals that adjust the speed with which activity in the network evolves toward an action triggering threshold (Wang et al., 2018). Similarly, it has been shown that brief pulse inputs to a network trained to reproduce time intervals can act as control signals to move the system's state toward specific regions responsible for measurement and/or production of time intervals (Sohn et al., 2019; Egger et al., 2019). Tonic contextual inputs moreover allow networks to flexibly adjust their output to identical inputs (Remington et al., 2018b; Sohn et al., 2019). However, in all these previous studies, the function of the control was only probed within the range for which the network was trained for. The problem of extrapolation instead translates to querying the network's low-dimensional dynamics under the influence of novel control signals outside of the training range. The conditions under which different types of control signals applied to a recurrent neural system might accommodate extrapolation are currently not understood.

Here, we tackled this question by combining network modeling, theory and neural data analysis. We analyzed recurrent neural networks (RNNs) trained on a set of flexible timing tasks, where the goal was to produce a time interval depending on various types of inputs (Wang et al., 2018; Remington et al., 2018b; Sohn et al., 2019). To investigate the role of the dimensionality of neural trajectories, we exploited the class of low-rank neural networks in which the dimensionality of dynamics is directly constrained by the rank of the recurrent connectivity matrix (Mastrogiuseppe and Ostojic, 2018; Schuessler et al., 2020a; Beiran et al., 2021; Dubreuil et al., 2020). We then compared the performance, generalization capabilities and neural dynamics of these low-rank RNNs with networks trained in absence of any constraint on the connectivity and the dimensionality of dynamics. We moreover contrasted tasks in which information was provided in terms of explicit tonic inputs, implicit temporal

intervals between input pulses, and contextual cues.

We found that, when driven by tonic contextual inputs, networks with low-rank connectivity were able to extrapolate well beyond their training range, while networks trained without the constraint on dimensionality were not. More specifically, in low-rank networks, contextual inputs parametrically controlled the input-output transform and its extrapolation. Reverse-engineering the trained low-rank RNNs showed that low-dimensional non-linear manifolds underlay generalization, and revealed a mechanism where contextual inputs modulated the dynamics along the manifolds while leaving their geometry invariant. A mean-field analysis of low-rank models further confirmed this mechanism, elucidated how the geometry of inputs with respect to recurrent connectivity controls flexible dynamics on invariant manifolds, and led to predictions that we tested on trained networks. Population analyses of neural data recorded in monkeys performing a time-interval reproduction task (Sohn et al., 2019) confirmed the key signatures of the mechanism. Altogether, our work brings forth evidence for computational benefits of controlling low-dimensional activity through tonic inputs, and provides a mechanistic framework for understanding the role of resulting neural manifolds for generalizing beyond the training set in flexible timing tasks.

2 Results

2.1 Dimensionality and generalization in flexible timing tasks

We trained recurrent neural networks (RNNs) on a series of flexible timing tasks, in which networks received various contextual and/or timing inputs and had to produce a time interval that was a function of those inputs (Fig. 1 A, B). In all tasks, the network output was a weighted sum of the activity of its units (see Methods). This output was required to produce a temporal ramp following an externally provided 'Set' signal. The output time interval corresponded to the time between the 'Set' signal and the time at which the network output crossed a fixed threshold (Wang et al., 2018; Remington et al., 2018b; Sohn et al., 2019). Inputs to the network determined the target interval, and therefore the desired slope of the output ramp, but the nature of inputs depended on the task (Fig. 1 B). Our goal was to test the hypothesis that low-dimensional activity favours generalization, i.e. helps networks extend their outputs to inputs not seen during training. We therefore examined the full input-output transform performed by the trained networks (Fig. 1 C), and specifically contrasted standard RNNs, which we subsequently refer to as "unconstrained RNNs", with low-rank RNNs in which the connectivity was constrained to minimize the embedding dimensionality of the collective dynamics (Fig. 1 D, Dubreuil et al. (2020)).

We first considered the Cue-Set-Go (CSG) task (Wang et al., 2018), where the goal was to produce an output time interval associated to a previously shown cue that varies from trial to trial. Correctly performing this task required learning the association between each cue and target output interval. In the network model of the task, the cue information was provided by the scalar amplitude of a tonic cue input (Fig. 1 B top). We found that networks with rank-two connectivity could solve this task with similar performance to unconstrained networks (Fig. 2 A, S1), but led to lower-dimensional dynamics (Fig. 2 A-B top, see Methods). During training, we used four different input amplitudes that are linearly related to their corresponding output intervals (four different color lines in Fig. 2 A). We then examined how the two types of networks generalized to previously unseen amplitudes. We found that both rank-two and unconstrained networks were able to interpolate to cue amplitudes in between those used for training (Fig. 2 C top). However, only low-rank networks showed extrapolation when networks were probed with cue amplitudes beyond the training range. More specifically, as the amplitude of the cue input was increased, low-rank networks generated increasing output time intervals

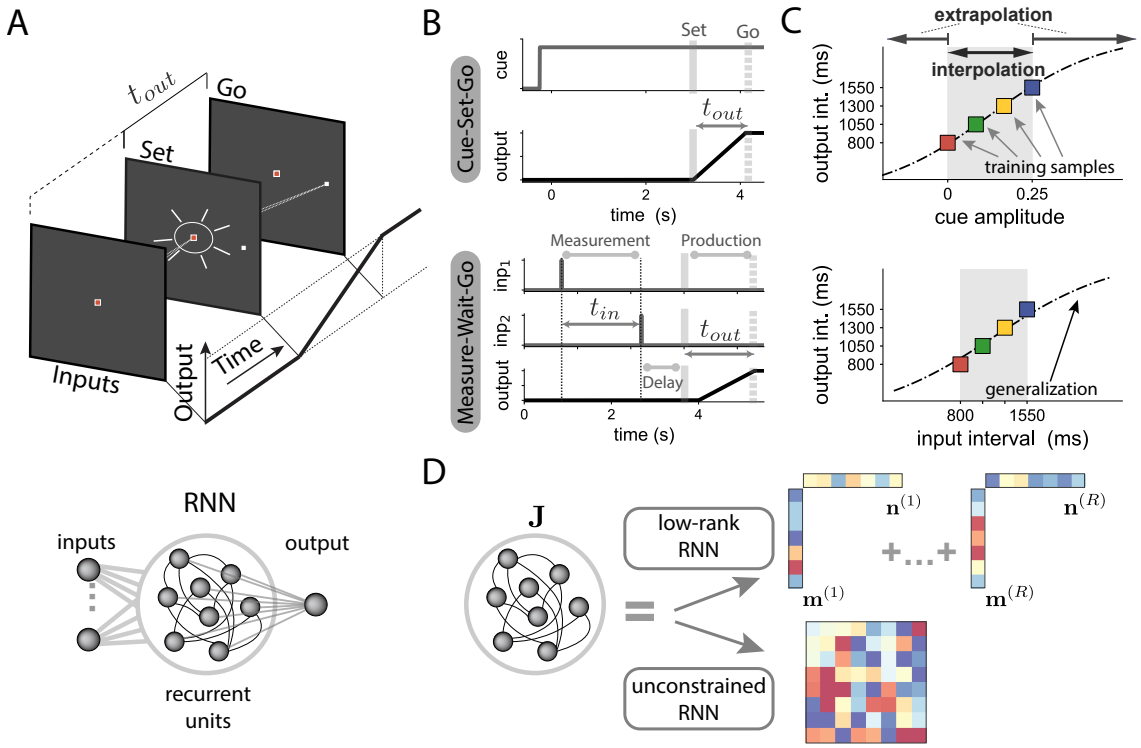


Figure 1: Investigating generalization in flexible timing tasks using low-rank recurrent neural networks (RNNs). **A** Flexible timing tasks. Top: On each trial, a series of timing inputs that depend on each task are presented to the network. After a random delay, a 'Set' signal is flashed, instructing the network to start producing a ramping output. The output time interval t_{out} is defined as the time elapsed between the 'Set' signal and the time point when the output ramp reaches a predetermined threshold value. Bottom: We trained recurrent neural networks (RNNs) where task-relevant inputs (left) are fed to the recurrently connected units. The network output is defined as a linear combination of the activity of recurrent units. **B** Structure of tasks. We considered two tasks: Cue-Set-Go (CSG) and Measure-Wait-Go (MWG). In the CSG task (top), the amplitude of a tonic input cue (grey line), present during the whole trial duration, indicates the time interval to be produced. In the MWG task (bottom), two input pulses are fed to the network ("inp 1" and "inp 2"), followed by a random delay. The input time interval t_{inp} between the two pulses indicates the target output interval t_{out}. **C** Characterizing generalization in terms of input-output functions. We trained networks on a set of training samples (colored squares), so that when learning converges, a specific cue amplitude (top) or input interval t_{inp} (bottom) corresponds to an output interval t_{out}. We then tested how trained networks generalized their outputs to inputs not seen during training, both within (interpolation, grey shaded region) and beyond (extrapolation) the training range, as illustrated with the dotted line. **D** To examine the influence of dimensionality on generalization, we compared two types of networks trained on each task. Low-rank networks had a connectivity matrix $\boldsymbol{J}$ of fixed rank R, which directly constrained the dimensionality of the dynamics (Dubreuil et al., 2020). Such networks were generated by parametrizing the connectivity matrix as in Eq. (2), and training only the entries of the connectivity vectors $\boldsymbol{m}^{(r)}$ and $\boldsymbol{n}^{(r)}$ for $r = 1 \dots R$. Unconstrained networks instead had full-rank connectivity matrices, where all the matrix entries were trained.

(Fig. 2 C, see Fig. S2 for additional examples of similarly trained networks), while networks without

rank constraint instead stopped producing a ramping signal altogether (see Fig. S3).

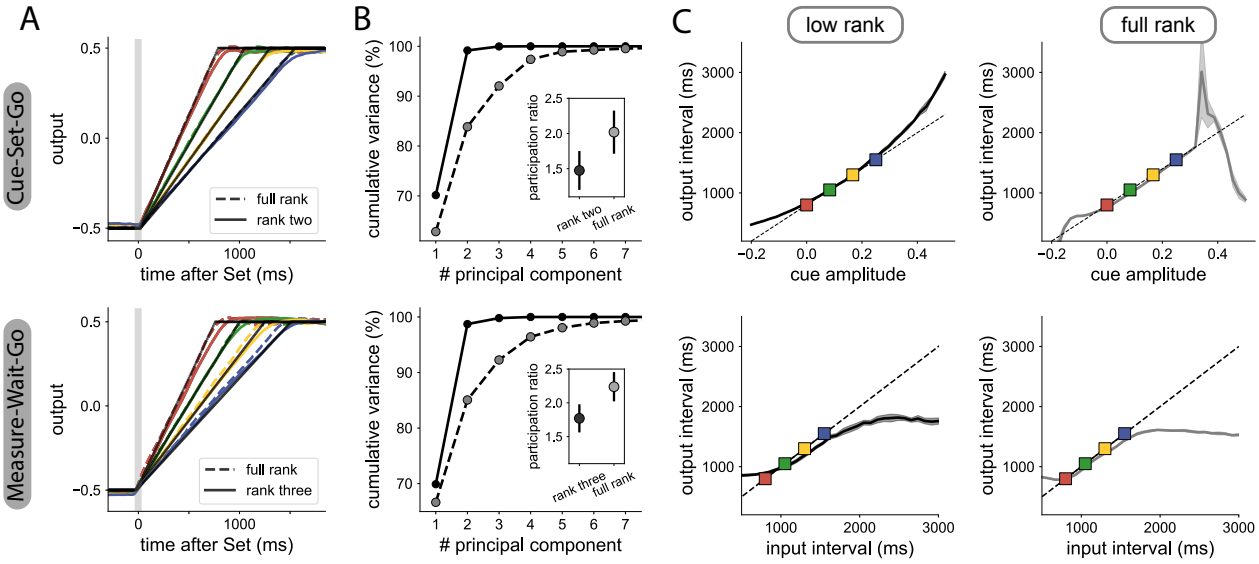


Figure 2: Influence of the dimensionality of network dynamics on generalization. Top: CSG task; bottom: MWG task. **A** Network output of RNNs trained on four different time intervals. RNNs with minimal rank $R = 2$ (solid colored lines) are compared with unconstrained RNNs (dashed colored lines). The thin black lines represent the target ramping output. **B** Dimensionality of neural dynamics following the 'Set' signal. Cumulative percentage of explained variance as a function of the number of principal components, for rank-two networks (solid line) and unconstrained networks (dashed line). Inset: participation ratio of activity (See Methods), with mean and SD over 6 trained RNNs. **C** Generalization of the trained RNNs to novel inputs within and beyond the training range. Low-rank (left) and unconstrained (right) RNNs. Lines and the shaded area respectively indicate the average output interval and the standard deviation estimated over ten trials per cue amplitude. In the CSG task (top row), the rank-two network interpolates and extrapolates, while the unconstrained network does not extrapolate. In the MWG task (bottom row), both types of networks fail to extrapolate to input intervals beyond the training set. The longest output interval that each network can generate is close to the longest output interval learned during training. The delay was randomized during training but the performance plots are based on a fixed delay of 1500ms (see S4 for other delays). See Figs. S2 and S5 for additional trained networks.

We next turned to a more complex task, Measure-Wait-Go (MWG), in which the desired output was indicated by the time interval between two identical pulse inputs (Fig. 1 B bottom). The network was required to measure that interval, and keep it in memory during a delay period of random duration, until the 'Set' signal triggered the output. This task is more demanding for two reasons. First, since the input interval is not encoded as a tonic input, the network must estimate the desired interval by tracking time between the two input pulses. Second, because of the delay period introduced before 'Set', the network has to hold this estimate in working memory until the time of 'Set'.

For this task, we found that a minimal rank $R = 3$ was required for low-rank networks to match the performance of unconstrained networks (Fig. 2 A-B bottom, Fig. S1). We then examined how low-rank and unconstrained networks responded to input intervals unseen during training. Both types of networks interpolated well to input intervals in between those used for training, however neither the rank-constrained nor the unconstrained networks were able to extrapolate to intervals longer or

shorter than those in the training set (Fig. 2 C bottom, see Fig. S4 for the input-output curves for different delays and Fig. S5 for the same analysis on other trained networks). Instead, the maximal output interval saturated at a value close to the longest interval used for training, thereby leading to a sigmoid input-output function.

Low-rank networks therefore appeared to be able to extrapolate in the CSG task but not in the MWG task, while unconstrained networks did not extrapolate in either task.

2.2 Contextual inputs enable parametric control of extrapolation

What factors could underlie the performance difference associated with extrapolation between CSG and MWG tasks? Since both tasks involved producing an interval, we reasoned that the difference is not related to the computational demands after 'Set'. Moreover, the random delay before 'Set' is unlikely to be the culprit as both tasks included such delay. The key remaining difference between the two tasks was that the interval in the CSG was specified by a tonic input, whereas in the MWG task, it had to be estimated from two brief input pulses. Since low-rank networks were able to extrapolate in the CSG but not the MWG task, we hypothesized that the tonic input in CSG may act as a parametric control signal allowing extrapolation in rank-constrained networks.

To probe this hypothesis, we designed an extension of the MWG with a tonic contextual input, which we refer to as the Measure-Wait-Go with Context (MWG+Ctxt). The MWG+Ctxt task was identical to the original MWG task in that the network had to measure an interval from two pulses, remember the interval over a delay, and finally produce the interval after 'Set'. However, on every trial, the network received an additional tonic input that specified the range of input intervals the network had to process (Fig. 3 A). A low tonic input indicated a relatively shorter set of intervals (800-1550 ms), and a higher tonic input, a set of longer intervals (1600-3100 ms). Note that the functional role of this context input is different from that of the CSG; in CSG, the input specifies the actual interval, whereas in the MWG+Ctxt task, it only specifies the underlying distribution. The contextual input therefore provides additional information that is not strictly necessary for performing the task, but may facilitate it.

Similar to what we found for the basic MWG task, both rank-three and unconstrained networks were able to solve the MWG+Ctxt task, and reproduced input intervals sampled from both distributions (Fig. S6). We then examined how each type of network generalized to unseen input intervals when associated with one of the two contextual cues used during training (Fig. 3 B).

Building on what we had learned from extrapolation in the CSG task, we reasoned that low-rank networks performing the MWG+Ctxt task may be able to use the tonic input as a control signal enabling extrapolation to interval ranges outside those the networks were trained for. To test this hypothesis, we examined generalization to both unseen input intervals and unseen contextual cues, comparing as before low-rank and unconstrained networks.

As in the original MWG task, when the input intervals were increased beyond the training range, the output intervals saturated to a maximal value in both types of networks. The only notable difference was that, in low-rank networks, this maximal value depended on the contextual input (Fig. 3 B left), and was set by the longest interval of the corresponding interval distribution, while in unconstrained networks it did not depend strongly on context (Fig. 3 B right). More generally, in low-rank networks, the contextual inputs modulated the input-output function, and biased output intervals towards the mean of the corresponding input distribution. This was evident from the distinct outputs the network generated for identical intervals under the two contextual inputs (Fig. 3 B left). In contrast, unconstrained networks were only weakly sensitive to the contextual cue and reproduced all intervals within the joint support of the two distributions similarly (Fig. 3 B right).

We next probed generalization to different values of the contextual input. Strikingly, in low-rank

networks we found that novel contextual inputs parametrically controlled the input-output transform (Fig. 3 C). In particular, contextual inputs outside of the training range led to strong extrapolation to unseen values of input intervals. Indeed, contextual inputs larger than used for training expanded the range of output intervals beyond the training range and shifted its mid-point to longer values (Fig. 3 C, bottom left), up to a maximal value of the contextual cue that depended on the training instance (Fig. S7). Conversely, contextual cues smaller than used in training instead reduced the output range and shifted its mean to smaller intervals (Fig. 3 C, bottom right). In contrast, in networks with unconstrained rank, although there was a larger heterogeneity across training instances, varying contextual cues did not have a strong effect on the input-output transform, which remained mostly confined to the range of input intervals used for training (Fig. S8).

Altogether, our results indicate that, when acting on low-dimensional neural dynamics, contextual inputs played the role of control signals that parametrically modulated the input-output transform performed by the network, thereby allowing for successful extrapolation beyond the training range. In contrast, unconstrained networks learnt a more rigid input-output mapping that could not be malleably controlled by new contextual inputs.

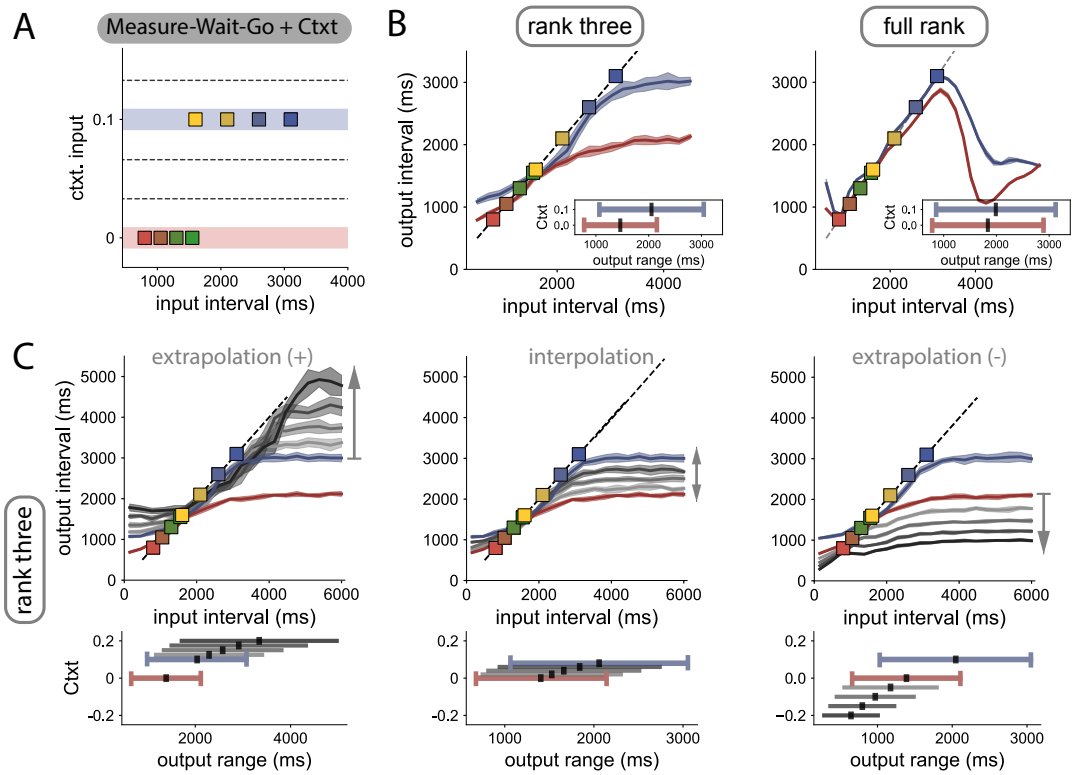


Figure 3: Control and extrapolation in the Measure-Wait-Go task with contextual cues. **A** Task design. Left: Input time intervals t_{inp} used for training (colored squares) are sampled from two distributions. A contextual cue, present during the whole trial duration, indicates from which distribution (red or blue lines) the input interval was drawn. After training, we probed how novel contextual cues (dashed lines) influence the input-output transform of the network. **B** Generalization to time intervals not seen during training, when using the two trained contextual cues (red and blue lines). Contextual cues modulate the input-output function in law-rank networks (left), but not in unconstrained networks (right). Lines and shading indicate average output interval t_{out} and standard deviation for independent trials. Inset: range of output intervals that the networks can produce in the different contexts. Black lines indicate the midpoint of the range. **C** Generalization in the low-rank network when using contextual cues not seen during training (context values shown in bottom insets). Top: Effects of contextual inputs above (left), within (center) and below (right) the training range. Bottom: Modulation of the range of output intervals as a function of the trained and novel contextual cues (colored and grey lines, respectively). See Fig. S7 for additional low-rank RNNs and Fig. S8 for unconstrained RNNs.

2.3 Non-linear activity manifolds underlie contextual control of extrapolation

To uncover the mechanisms by which low-rank networks implement parametric control for flexible timing, we examined the underlying low-dimensional dynamics (Sussillo and Barak, 2013; Vyas et al., 2020; Dubreuil et al., 2020). In low-rank networks, the connectivity constrains the activity to be embedded in a low-dimensional subspace (Mastrogiuseppe and Ostojic, 2018; Beiran et al., 2021; Dubreuil et al., 2020), allowing for a direct visualisation. Examining the resulting low-dimensional dynamics for networks trained on flexible timing tasks, in this section we show that, within the embedding subspace, the neural trajectories are attracted to non-linear manifolds along which they evolve slowly. The dynamics on, and structure of, these manifolds across conditions determine the

extrapolation properties of the trained networks. Here we summarize this reverse-engineering approach and the main results, and provide details in Methods.

Low-dimensional embedding of neural activity. The RNNs used to implement flexible timing tasks consisted of N units, with the dynamics for the activity x_i of the i-th unit given by:

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

Here $\phi\left(x\right) = \tanh\left(x\right)$ is the single neuron transfer function, J_{ij} is the strength of the recurrent connection from unit j to unit i, and $\eta_i\left(t\right)$ is a single-unit noise source. The networks received N_{in} task-dependent scalar inputs $u_s\left(t\right)$ for $s = 1, \ldots, N_{in}$, each along a set of feed-forward weights that defined an *input vector* $\boldsymbol{I}^{(s)} = \{I_i^{(s)}\}_{i=1\ldots N}$ across units. Inputs $u_s\left(t\right)$ were either delivered as brief pulses ('Set' signal, input interval pulses in the MWG task) or as tonic signals that were constant over the duration of a trial (cue input in the CSG task, contextual input in the MWG+Ctxt task).

The collective activity in the network can be described in terms of trajectories $\boldsymbol{x}(t) = \{x_i(t)\}_{i=1\ldots N}$ in the N-dimensional state space, where each axis corresponds to the activity of one unit (Fig. 4 A). In low-rank networks, the connectivity constrains the trajectories to reside in a low-dimensional linear subspace of the state space (Mastrogiuseppe and Ostojic, 2018; Beiran et al., 2021; Dubreuil et al., 2020), that we refer to as the *embedding space* (Jazayeri and Ostojic, 2021). In a rank-R network, the recurrent connectivity can be expressed in terms of R pairs of *connectivity vectors* $\boldsymbol{m}^{(r)} = \{m_i^{(r)}\}_{i=1\ldots N}$ and $\boldsymbol{n}^{(r)} = \{n_i^{(r)}\}_{i=1\ldots N}$ for $r = 1 \ldots R$, which together specify the recurrent connections through

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

The connectivity vectors, as well as the feed-forward input vectors $\boldsymbol{I}^{(s)}$, define specific directions within the N-dimensional state space that determine the network dynamics. In particular, the trajectories are confined to the embedding space spanned by the recurrent connectivity vectors $\boldsymbol{m}^{(r)}$ and the feed-forward input vectors $\boldsymbol{I}^{(s)}$ (Fig. 4 A). The trajectories of activity $\boldsymbol{x}(t)$ can therefore be parameterized in terms of Cartesian coordinates along the basis formed by the vectors $\boldsymbol{m}^{(r)}$ and $\boldsymbol{I}^{(s)}$ (Mastrogiuseppe and Ostojic, 2018; Beiran et al., 2021; Dubreuil et al., 2020):

$$\boldsymbol{x}\left(t\right) = \sum_{r=1}^{R} \kappa_r\left(t\right) \boldsymbol{m}^{(r)} + \sum_{s=1}^{N_{in}} v_s\left(t\right) \boldsymbol{I}^{(s)}. \tag{3}$$

The variables $\boldsymbol{\kappa} = \{\kappa_r\}_{r=1\ldots R}$ and $\boldsymbol{v} = \{v_s\}_{s=1\ldots N_{in}}$ represent respectively activity along the *recurrent* and input-driven subspaces of the embedding space (Wang et al., 2018). The dimensionality of the embedding space is therefore $R + N_{in}$, where R dimensions correspond to the recurrent subspace, and N_{in} to the input-driven subspace. The evolution of the activity in the network can then be described in terms of a dynamical system for the recurrent variables $\boldsymbol{\kappa}$ driven by inputs $\boldsymbol{v}$ (Beiran et al., 2021; Dubreuil et al., 2020):

$$\tau \frac{d\boldsymbol{\kappa}}{dt}\left(t\right) = \boldsymbol{F}\left(\boldsymbol{\kappa}, \boldsymbol{v}\right). \tag{4}$$

In Eq. (4), for constant inputs the non-linear function $\boldsymbol{F}$ describes the dynamical landscape that determines the *flow* of the low-dimensional activity in the recurrent subspace (Fig. 4 A). Pulse-like inputs instantaneously shift the position of activity in the embedding space, and the transient dynamics

in between them are determined by this dynamical landscape. Tonic cue inputs instead shift the recurrent subspace within the embedding space and thereby modify the full dynamical landscape (Fig. 4 B). The low-dimensional embedding of neural trajectories can therefore be leveraged to explore the dynamics of trained recurrent neural networks, by directly visualizing the dynamical landscape and the flow of trajectories in the recurrent subspace instead of considering the full, high-dimensional state space.

Non-linear manifolds within the embedding space. In low-rank networks, the trajectories of activity reside in the low-dimensional embedding space, but they do not necessarily explore that space uniformly. Examining networks trained on flexible timing tasks, we found that neural trajectories quickly converged to lower-dimensional, non-linear regions within the embedding space. We refer to these regions as *non-linear manifolds* (Jazayeri and Ostojic, 2021), and we devised methods to identify them from network dynamics (Methods and Fig. S9). We next describe these manifolds and their influence on dynamics and computations for networks trained on individual tasks.

Cue-Set-Go task. For rank-two networks that implemented the CSG task, the recurrent subspace was of dimension two and was parameterized by Cartesian coordinates κ_1 and κ_2 (Fig. 4 C). In a given trial, corresponding to a fixed cue amplitude, we found that the dynamical landscape on this recurrent subspace displayed an attractive ring-like manifold on which the dynamics were slow (Fig. 4 C, red line). This manifold connected two stable fixed points through two intermediate saddle points (Fig. 4 C, red and white dots respectively). At trial onset, the neural activity started from the first stable fixed point, and was then pushed by the 'Set' pulse above a saddle point. This generated a neural trajectory in the recurrent subspace (Fig. 4 C, black line) that was quickly attracted to the ring-like manifold (Figs. S9 and S10), and subsequently followed slow dynamics along this manifold towards the second stable fixed point. The position on this non-linear manifold therefore represented the time since the 'Set' pulse, and was directly transformed into a ramping output by the readout of the network.

Different trials in the CSG task correspond to different amplitudes of the tonic input cue that shift the position of the recurrent subspace within the embedding space and modulate the dynamical landscape on it (Fig. 4 D). For each value of the cue amplitude, we found that trajectories evolved along parallel ring-like manifolds, which together formed a two-dimensional cylinder when visualised in the three-dimensional embedding space defined by κ_1, κ_2 and the input cue as a third coordinate (Fig. 4 D right). The shape of the ring-like manifolds and the position of fixed points were largely invariant, but the amplitude of the cue controlled the speed of the dynamics along each ring (Fig. 4 E), and thereby the slope of the ramping output that determined the output interval (Fig. 2 A). Because of the cylindrical geometry, extending cue amplitudes to values outside of the training range preserved the overall structure of the manifold (Fig. 4 D blacked dashed line), and thereby ensured lawful extrapolation of the required outputs. The modulation of speed along a low-dimensional manifold of invariant geometry therefore subserves the contextual control of extrapolation in the CSG task.

Measure-Wait-Go task. For rank three networks that implemented the MWG task, the recurrent subspace was three-dimensional. Within that subspace, we found that the trajectories were attracted to a spherically-shaped manifold (Fig. S9), along which they evolved at a much lower speed (Fig. 4 F). The dynamics of activity underlying the basic MWG task could therefore be described in terms of trajectories along a non-linear manifold, in a manner analogous to individual trials in the CSG task. During the measurement period, the two input pulses that defined the input time interval led to trajectories that quickly converged to a localized region on the spherical manifold where the speed was minimal (Fig. S10). This region played the role of a line attractor, the position along which encoded the input time interval during the delay period. The subsequent 'Set' input then generated trajectories that evolved towards the other side of the sphere, at speeds set by the initial condition on the line attractor, therefore generating ramp signals with varying slopes (Fig. S3). Crucially, the line attractor

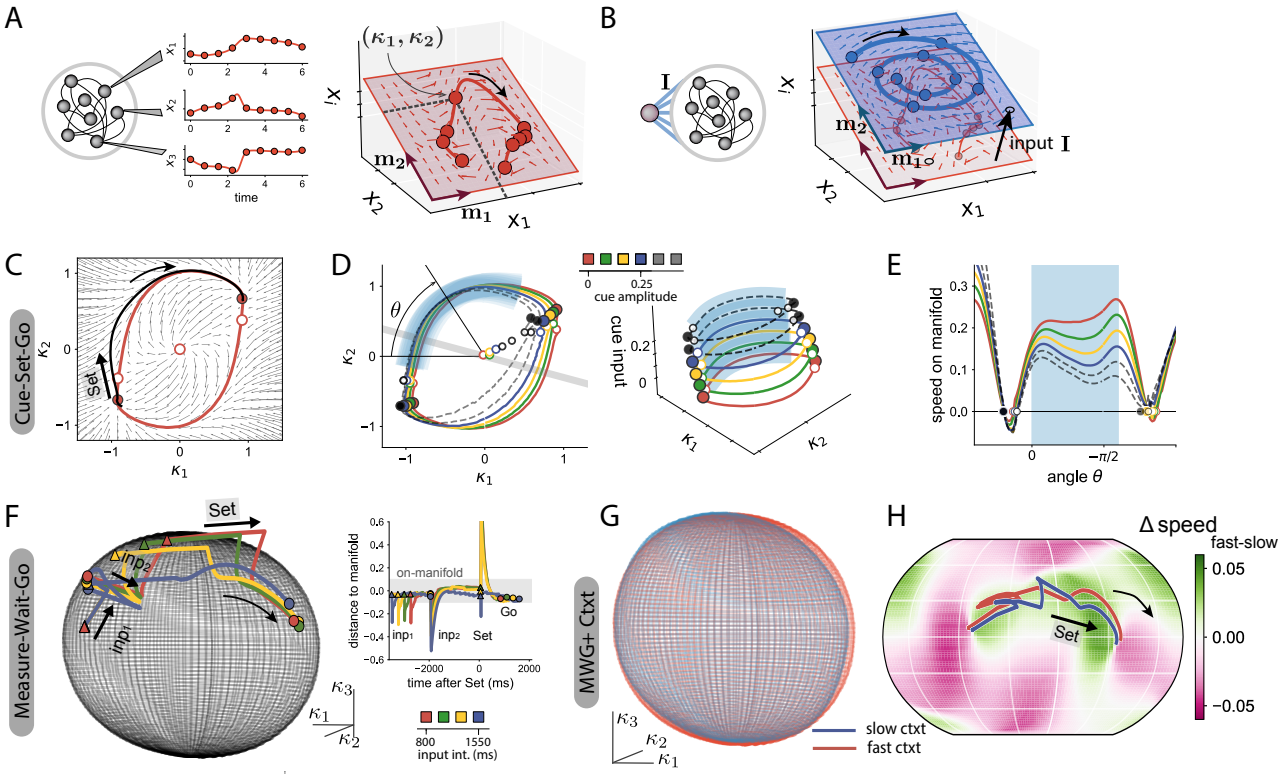


Figure 4: Reverse-engineering RNNs trained on flexible timing tasks. **A-B** Low-dimensional description of activity dynamics. **A** The temporal activity of different units in the network (left) can be represented as a trajectory in state space (right, thick red line), where each axis corresponds to the activity x_i of one unit. In low-rank networks, trajectories are embedded in a lower dimensional space (red shaded plane, see Eq. (3)). For instance, in a rank-two network, in absence of input, this embedding space is spanned by connectivity vectors $\boldsymbol{m}^{(1)}$ and $\boldsymbol{m}^{(2)}$, so that neural trajectories can be parametrized by two recurrent variables κ_1 and κ_2, and their dynamics represented in terms of a flow field on the embedding space (arrows, see Eq. (4)). **B** External inputs increase the dimensionality of the embedding space. In this illustration, a single input is added, so that the embedding subspace is now three-dimensional: two dimensions for the recurrent subspace, spanned by $\boldsymbol{m}^{(1)}$ and $\boldsymbol{m}^{(2)}$, and one additional dimension for the input vector $\boldsymbol{I}$. If the input is tonic, it shifts the recurrent subspace ($\boldsymbol{m}^{(1)}$ - $\boldsymbol{m}^{(2)}$) along the input direction (black arrow, from the red to the blue plane) and modifies the flow fields. **C-E** RNN trained on the CSG task. Color-filled dots represent stable fixed points, and white dots represent unstable fixed points. **C** Dynamical landscape in the embedding subspace of trained rank two network (small arrows), and neural trajectory for one trial (black line, $u_{cue} = 0$). The red line represents the non-linear manifold to which dynamics converge from arbitrary initial conditions (Fig. S9). The trial starts at the bottom left stable fixed point. The 'Set' pulse initiates a trajectory towards the opposite stable fixed point, which quickly converges to the slow manifold (see Fig. S10 A), and evolves along it. **D** Manifolds generated by a trained RNN for different amplitudes of the cue. Colored lines correspond to the different cues used for training and black dashed lines correspond to amplitudes beyond the training range. Left: Two-dimensional projections of the manifolds onto the recurrent subspace spanned by vectors $\boldsymbol{m}^{(1)}$ and $\boldsymbol{m}^{(2)}$. The grey line is the projection of the readout vector on the recurrent subspace. Right: Three dimensional visualization of the manifolds in the full embedding subspace, spanned by $\boldsymbol{m}^{(1)}$, $\boldsymbol{m}^{(2)}$ and $\boldsymbol{I_{cue}}$. The shaded blue region indicates the section of the manifolds along which neural trajectories evolve when performing the task. Any state on the manifold can be determined by the polar coordinate θ. Increasing the cue amplitude, even to values beyond the training range (black dashed lines), keeps the geometry of manifolds invariant.

Figure 4 *(previous page)*: **E** Speed along the manifold as a function of the polar angle θ, that parametrizes all on-manifold states. The speed along the manifold is scaled by the cue amplitude, even beyond the training range. Shaded blue region as in **D**. **F** RNN trained on the MWG task. Left: The dynamics in the embedding subspace of the trained rank-three network generated a spherical manifold to which trajectories were quickly attracted after fast transient responses to input pulses (Fig. S9). Right: distance of trajectories solving the task to the manifold's surface. **G** RNN trained on the MWG+Ctxt task. Spherical manifolds on the recurrent subspace for two different contextual cues (red: fast context, $u_{ctxt} = 0$, blue: slow context, $u_{ctxt} = 0.1$). **H** The contextual cue modulates the speed of dynamics along the manifold. The colormap shows the difference in speed on the manifold's surface between the two contexts. In the region of the manifold where trajectories evolve (red and blue lines), the speed is lower for stronger contextual cues (slow context). A stronger contextual amplitude slows down the non-linear manifold in the region of interest (see Fig. S10 C for novel contextual amplitudes).

that encoded the input interval occupied a bounded region on the sphere, so that any input interval outside of the training range converged to one of the extremities of the attractor. The finite limits of the line attractor therefore lead to a sigmoidal input-output function, where the output intervals that corresponded to the bounds of the training range as seen in Fig. 2 C.

For the extended MWG task with a contextual cue, the additional tonic contextual inputs modified the flow of the dynamics in the three-dimensional recurrent space. In a manner analogous to the CSG task, increasing the amplitude of the contextual input largely preserved the shape and position of the attractive spherical manifold (Fig. 4 G), while scaling the speed of the dynamics on it (Fig. 4 H). The contextual input thereby controlled the speed of trajectories of activity and parametrically modulated the input-output transform performed by the network. This effect extended to values of contextual inputs well beyond the trained region (Fig. S9), and thereby controlled extrapolation to previously unseen values for both input intervals and contextual inputs.

In summary, reverse-engineering low-rank networks trained on the CSG and MWG tasks revealed that in both tasks extrapolation beyond the training range relied on a mechanism based on an invariant geometry of underlying neural activity manifolds, the dynamics along which were parametrically controlled by tonic inputs.

2.4 Controlling the geometry and dynamics on non-linear manifolds

Our reverse-engineering analysis revealed that parametric control of extrapolation in trained low-rank networks relied on modulating dynamics over non-linear manifolds of invariant geometry. To further unravel how the properties of recurrent connectivity and tonic inputs respectively contribute to controlling the geometry of, and dynamics on non-linear activity manifolds, we next investigated simplified, mathematically tractable networks. In such networks, a mathematical analysis allows us to directly infer dynamics from the connectivity and input parameters and to synthesize networks that perform specific computations (Mastrogiuseppe and Ostojic, 2018; Schuessler et al., 2020a; Beiran et al., 2021; Dubreuil et al., 2020; Pollock and Jazayeri, 2020), thereby demonstrating that the mechanisms identified through reverse-engineering are sufficient to implement generalization in flexible timing tasks.

We considered the restricted class of Gaussian low-rank RNNs (Mastrogiuseppe and Ostojic, 2018; Beiran et al., 2021; Dubreuil et al., 2020) where for each neuron i, the set of components $\left\{m_i^{(r)}, n_i^{(r)}\right\}_{r=1...R}$ and $\left\{I_i^{(s)}\right\}_{s=1...N_{in}}$ along connectivity and input vectors are drawn randomly and independently from a zero-mean multivariate Gaussian distribution (Fig. 5 A). Unlike trained low-

rank networks, which in principle depend on a high number of parameters (order N) these simplified networks are fully specified by a few parameters (order $(2R + N_{in})^2$) that correspond to the entries of the covariance matrix of the underlying Gaussian distribution, or equivalently the matrix of pairwise *overlaps* between connectivity and input vectors (Fig. 5 A right, Methods). We therefore investigated how this overlap structure determines the dynamics that underlie computations.

Generating slow manifolds through recurrent connectivity. We first focused on the recurrent interactions, and investigated which type of overlap structure between connectivity vectors generates slow attractive manifolds as seen in trained networks. In particular, we analyzed how the position of the fixed points and speed along manifolds depend on the overlap matrix. As reported in previous studies (Mastrogiuseppe and Ostojic, 2018; Pereira and Brunel, 2018; Beiran et al., 2021) and detailed in the Methods, a mean-field analysis shows that in the limit of large networks, attractive manifolds arise when the overlap matrix is diagonal and its non-zero entries are equal to each other and sufficiently strong (Fig. 5 B left). In Gaussian low-rank RNNs, such an overlap matrix, proportional to the identity, implies that the dynamics induced by recurrent connectivity are symmetric along the different directions of the recurrent subspace spanned by the connectivity vectors $\boldsymbol{m}^{(r)}$ for $r = 1 \dots R$. In general, for a rank R network, such a symmetry induces an attractive R-dimensional spherical manifold in neural space where each point on the manifold is an attractive fixed point of the dynamics (see Methods). Accordingly, for rank-two networks this structure leads to a continuous ring attractor, embedded in the plane defined by the connectivity vectors $\boldsymbol{m}^{(1)}$ and $\boldsymbol{m}^{(2)}$ (Fig. 5 B), and for rank-three networks to a spherical attractor.

The mean-field analysis formally holds in the limit of infinitely large networks. Dynamics in networks of finite size can be described by adding random perturbations to the overlap matrix describing the connectivity, which is therefore never perfectly diagonal. Perturbations away from a diagonal overlap structure break up the continuous attractor so that only a small number of points on it remain exact fixed points of the dynamics (Fig. 5 C, see Methods for details). On the remainder of the original continuous attractor, the flow of the dynamics however stays very slow even for relatively large deviations from a diagonal overlap structure. For rank two connectivity, the continuous attractor predicted by the mean-field analysis therefore leads to a ring-like manifold on which slow dynamics connect stable fixed points and saddles (Fig. 5 C, middle), as seen in trained low-rank networks (Fig. 4 C-D). Specific deviations from a diagonal overlap matrix moreover determine the precise position of the fixed points on the manifold. In particular, a weak off-diagonal component, corresponding to a non-zero overlap between connectivity vectors of different rank-one structures, rotates the position of saddle points and brings them closer to stable fixed points (Fig. 5 D). This type of structure leads to long transient trajectories from a saddle to a fixed point, analogous to those underlying ramping signals in trained networks (Fig. 4 C).

Controlling dynamics through tonic inputs. We next examined how a tonic cue, i.e. a constant external input along an input vector $\boldsymbol{I}$, impacts the geometry of recurrently-generated manifolds and the dynamics on them. Our mean-field analysis showed that the geometrical arrangement between the input vector $\boldsymbol{I}$ and recurrent vectors, as quantified by their overlaps, was the key factor that determined the effect of the input on the manifold. We therefore distinguished between *non-specific inputs* for which the input vector was orthogonal to all connectivity vectors, and *subspace-specific inputs*, for which the input vector was correlated with the recurrent connectivity structure.

Non-specific inputs modify both the dynamics on the manifold, and its geometry in neural state space (Fig. 6 A-B, top row). In particular, increasing the amplitude of non-specific inputs shrinks the radius of the activity manifold, until it eventually collapses. In contrast, subspace-specific inputs leave approximately invariant the geometry of the manifold, and the position of fixed points on it (Fig. 6 A bottom). Varying the amplitude of subspace-specific inputs however strongly modulates the speed of dynamics along the manifold (Fig. 6 B bottom). Subspace-specific inputs therefore reproduce

the mechanism of speed modulation on invariant manifolds found when reverse-engineering trained networks.

Based on the mechanism identified using the mean-field analysis, we hypothesized that the input components required to produce flexible timing behavior are those specific to the recurrent subspace. We tested this prediction on the low-rank neural networks trained on the Cue-Set-Go task and Measure-Wait-Go task with context. We perturbed the trained input vectors in two ways; we either kept only input components specific to the connectivity subspace and removed all others, or kept only input components orthogonal to the connectivity subspace. As predicted, keeping only the non-specific components of the input vectors completely hindered the computation (Fig. 6 C top). In contrast, removing all non subspace-specific components from the input vectors in the trained networks solving the timing tasks, without strongly affecting performance (Fig. 6 C bottom). These results show that subspace-specific input vectors were required to solve timing tasks.

The mean-field analysis of simplified, Gaussian low-rank networks, allowed us to synthesize networks in which tonic inputs control dynamics over non-linear manifolds of invariant geometry. Going one step further, we next capitalized on these insights to directly design networks that perform the Cue-Set-Go and Measure-Wait-Go tasks based on this mechanism, by setting connectivity parameters without training (Fig. S11). Such minimal network models demonstrate that the mechanisms identified by reverse-engineering trained networks are indeed sufficient for implementing the flexible timing tasks over a large range of inputs.

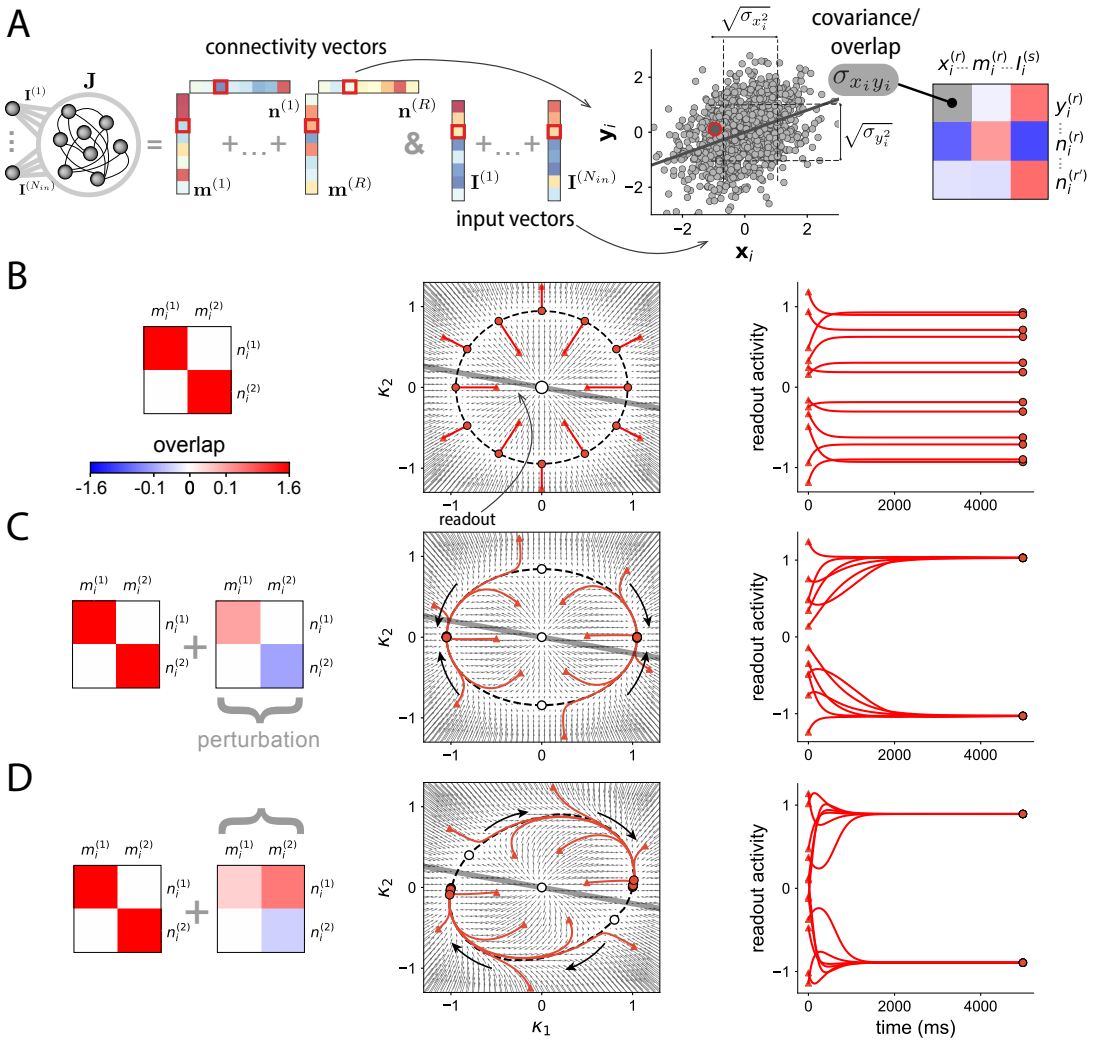


Figure 5: Shaping manifolds by the recurrent connectivity in low-rank networks. **A** Mean-field framework. We assume that the entries of the recurrent connectivity and input vectors are randomly drawn from a zero-mean multivariate Gaussian distribution. The parameters that define the connectivity are therefore the variances $\sigma_{x_i^2}$ and covariances $\sigma_{x_i y_i}$, for $\{x_i, y_i\} \in \left\{m_i^{(r)}, n_i^{(r)}, I_i^{(s)}\right\}$, or equivalently the pairwise overlaps between vectors $\boldsymbol{m}^{(r)}, \boldsymbol{n}^{(r)}, \boldsymbol{I}^{(s)}$. These pairwise overlaps can be represented as entries of a matrix (right) whose values determine the dynamics in the limit of large networks. **B** Ring-attractor dynamics of a rank-two network with overlap structure proportional to the identity (left). Middle: In the mean-field limit, the recurrent connectivity generates a ring attractor in the recurrent subspace. Red lines correspond to trajectories from different initial conditions, that converge towards fixed points. The grey line indicates the readout direction. Right: Temporal trajectories of activity projected along the readout dimension. The trajectories correspond to the mean-field dynamics, given by Eq. (21) (see Methods). **C** Analogous to **B**, in a rank-two network where the overlap structure is perturbed (left, second term). The overlap matrix is still diagonal, but the diagonal entries are different from each other. Middle: This connectivity structure leads to two stable fixed points and two saddle points (white dots) located on the original ring attractor which now forms a slow manifold. Trajectories of activity quickly converge to the ring manifold, and then evolve slowly towards one of the stable fixed points (right). **D** Analogous to **B**, in a rank-two network where the perturbation matrix is not diagonal (left). Middle: The off-diagonal component of the overlap matrix rotates the saddle points closer to the stable fixed points on the slow manifold. Right: The readout activity produces slow ramping signals, as trajectories evolve from one side of the manifold to the other.

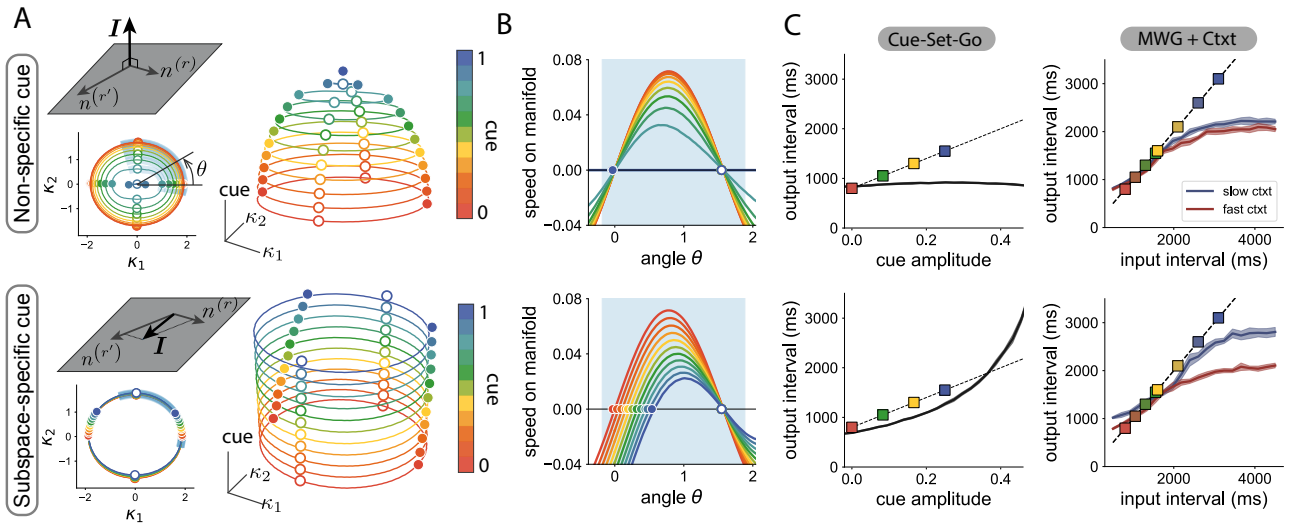


Figure 6: Effect of tonic cue inputs on the geometry of, and dynamics on, manifolds. Top: non-specific cue orthogonal to the recurrent connectivity vectors. Bottom: specific cue input along the subspace spanned by recurrent connectivity vectors. **A** Projection of manifolds on the recurrent subspace (bottom left inset), and on the three-dimensional embedding subspace (right). Different colors correspond to different cue amplitudes. Solid lines indicate the manifold location. Colored dots are stable fixed points, white dots are saddle points. **B** Speed along the manifold, parametrized by the polar angle, along the shaded blue section indicated in **A**. **C** Performance in the CSG (left) and MWG+Ctxt tasks (right), when either only non-specific (top) or only subspace-specific (bottom) components of the cue of the trained RNN are kept after training. Parameters: A, $\sigma_{m_1 n_1} = 2.6, \sigma_{m_2,n_2} = 2.4$, $\sigma_{I^2} = 1.4 \cdot \text{cue}$, B: $\boldsymbol{\sigma_{nI}} = (0, 0.1) \cdot \text{cue}$, $\sigma_{I^2} = 0.02 \cdot \text{cue}^2$.

2.5 Signatures of the computational mechanism in neural data

Our analyses of network models revealed the dynamical mechanisms underlying parametric control of extrapolation. We next sought to test whether the key signatures of these mechanisms were present in neural activity recorded during flexible timing behavior. More specifically, we examined the predictions recapitulated in Fig. 7 (top) in a low-rank network trained on the MWG+Ctxt task. First, contextual cues translate the manifolds of activity along a contextual axis in state space that is orthogonal to the subspace in which trajectories evolve in individual contexts (Fig. 7 A top). Second, when projected on the subspace orthogonal to the context axis, the geometry of neural manifolds is invariant across contextual inputs (Fig. 7 B top). Third, the strength of contextual inputs, as quantified by the projection on the context axis, modulates the speed of the dynamics along manifolds (Fig. 7 C top).

To test for the presence of these signatures in the brain, we analyzed neural activity recorded in the dorsomedial frontal cortex (DMFC) of two monkeys performing a time interval reproduction task analogous to the MWG+Ctxt task but without a delay period (Sohn et al., 2019; Meirhaeghe et al., 2021). Specifically, animals had to measure and immediately afterward reproduce different time intervals. The feature that made this task comparable to the MWG+Ctxt was that the time interval on each trial was sampled from one of two distributions (i.e. two contexts), and the color of the fixation spot throughout the trial explicitly cued the relevant distribution thereby providing a tonic cue. While the network model allowed us to analyze the responses to any contextual input, the neural data analysis is limited to the two contexts. We focused on the measurement period between the two input pulses, as this period was strictly identical across tasks. Combining a simple principal component analysis with a decoding analysis of context showed that neural trajectories were separated along a contextual axis in state space throughout the measurement period (Fig. 7 A bottom). A different projection of the activity, along a subspace orthogonal to the context axis, revealed that activity evolved along manifolds that were invariant across contexts (Fig. 7 B bottom). Finally, the projection on the contextual axis was strongly correlated with the speed of the dynamics in the individual trajectories (Fig. 7 C bottom).

Altogether, neural activity exhibited the key signatures of parametric control predicted from computational modeling and theory, suggesting that the dynamics of neural trajectories in the cortex may optimize the capacity to generalize to previously unseen inputs.

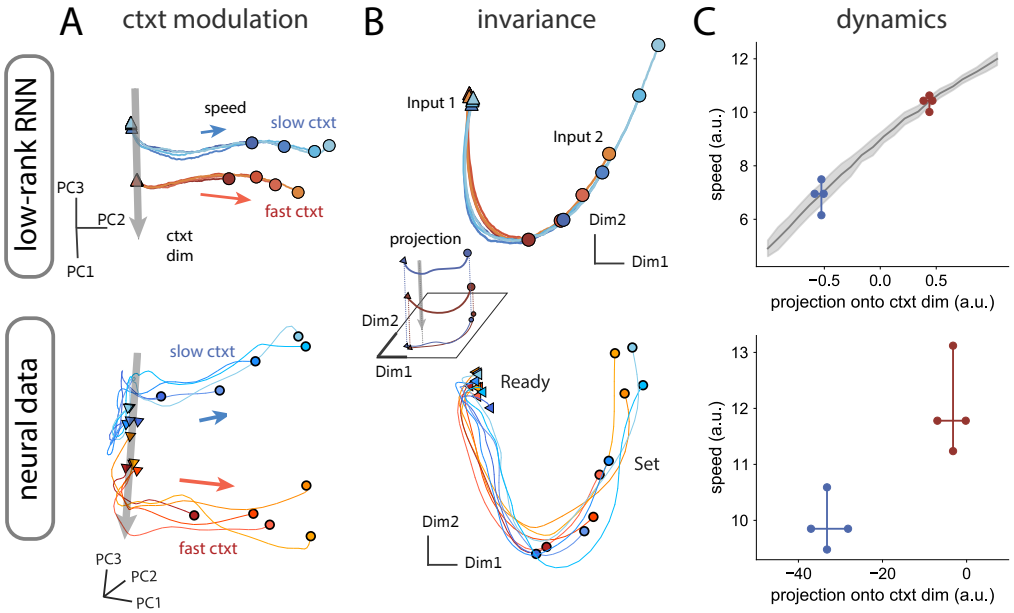


Figure 7: Signatures of the computational mechanism in neural data. Comparison between a low-rank network trained on the MWG+Ctxt task (top) and neural data in a time-reproduction task (Sohn et al., 2019; Meirhaeghe et al., 2021) (bottom). **A** -**B** Projections on the first principal components of the activity during the measurement epoch between the two input pulses. Different trajectories correspond to different input intervals; colors denote the two contexts. **A** Trajectories for different contexts are separated along a context axis (grey arrow) determined using a decoding analysis (Methods), and evolve along an orthogonal subspace (colored arrows). **B** In a two-dimensional projection orthogonal to the context axis, trajectories evolve along a non-linear manifold that is invariant across contexts. **C** Projection of neural activity on the context dimension before measurement vs the speed of neural trajectories during the measurement epoch (see Methods). The projection on the contextual axis controls the speed of neural trajectories along the manifold. Blue and red crosses indicate the 99% confidence interval for the estimated speed and projection on context dimension for the slow and fast contexts. The grey line on top indicates the average speed and projection for contextual cues not used during training (error bars indicate the standard deviation in speed across different input intervals).

3 Discussion

The ongoing progress in recording technologies has enabled an unprecedented view of large-scale neural activity underlying cognitive tasks (Urai et al., 2021). An overarching finding across a large number of experimental studies has been that the complexity of population activity does not increase with the number of recorded neurons, but instead displays an intricate collective structure (Saxena and Cunningham, 2019; Chung and Abbott, 2021). In particular, trajectories of neural activity, when represented in neural state space where each axis corresponds to the activity of one recorded neuron, appear to often be embedded within a much lower dimensional subspace (Gao et al., 2015; Gallego et al., 2017; Jazayeri and Ostojic, 2021). One possibility is that this low-dimensionality is an artifact of focusing on overly simple behavioral tasks (Gao et al., 2017). An alternative hypothesis is that the observed low dimensionality plays an active computational role. Indeed, studies of stimulus categorization have proposed that low-dimensional representations facilitate the generalization to previously unseen stimuli of a given class (DiCarlo and Cox, 2007; DiCarlo et al., 2012; Chung et al., 2018; Flesch

et al., 2021), but whether and how that idea extends to other types of tasks has been an open question that is challenging to address directly on neural data.

Examining recurrent neural networks trained on a class of flexible timing tasks, in this study we show that controlling low-dimensional dynamics with contextual inputs enables extrapolation to inputs well beyond the training range. Reverse-engineering and theoretical analyses of the recurrent networks demonstrated that the underlying mechanism of extrapolation relied on a specific geometry of collective dynamics. Within a given condition, collective dynamics evolved along non-linear manifolds, while across conditions, contextual cues modulated the manifolds along an orthogonal direction. This modulation parametrically controlled the speed of dynamics on the manifolds while leaving their geometry invariant.

The geometric and dynamical signatures of the mechanisms underlying extrapolation are consistent with a number of properties of neural activity recorded in flexible timing tasks. Previous analyses of recordings in the dorsomedial frontal cortex while monkeys performed the CSG task revealed a temporal scaling of neural responses along a projected trajectory that was invariant across conditions (Wang et al., 2018). This finding is fully consistent with our prediction of dynamics along an invariant manifold in the recurrent subspace orthogonal to the input cue (Fig. 4 D). Here we additionally re-examined neural activity in the time interval reproduction task (Sohn et al., 2019; Meirhaeghe et al., 2021) directly comparable to the MWG+Ctxt task. We showed that the geometric and dynamical organization of neural trajectories was highly similar to model predictions (Fig. 7). Our analyses in particular uncovered an axis in state space that separated trajectories across contexts, while projecting on the subspace orthogonal to this axis revealed an invariant manifold. The projection of activity along the contextual axis predicted previously reported speed modulations (Meirhaeghe et al., 2021). Altogether, our computational and theoretical results therefore connect a variety of previous experimental observations into a coherent picture which suggests that the organization of neural activity may optimize generalization to previously unseen inputs.

In studies of flexible behavior, contextual cues typically play a key role by specifying for instance the currently relevant stimulus feature (Mante et al., 2013), the relevant stimulus-response association (Rigotti et al., 2010; Saez et al., 2015; Bouchacourt et al., 2020) or the prior distribution in a noisy environment (Jazayeri and Shadlen, 2010, 2015; Sohn et al., 2019; Meirhaeghe et al., 2021). In contrast, in our implementation of the MWG+Ctxt task, contextual cues did not bring essential information needed to perform the task, as on each trial the expected response was indicated unambiguously and in a noise-free manner by input pulses. Contextual cues indicated the global distribution the stimuli were drawn from, but the task could in principle be performed optimally on trained stimuli even when ignoring that information. Our central finding is that taking into account the summary statistics encoded by the contextual cue enabled low-rank networks to extrapolate to untrained stimuli. Our results therefore uncovered a novel computational role of contextual inputs for generalization. This generalization relied on continuous, parametric control by the contextual cue, a mechanism related to recent reports of parametric control of non-linear dynamics in RNNs (Klos et al., 2020; Kim et al., 2021). These studies did not focus on the role of dimensionality, but interestingly relied on specific training schemes that implicitly induce a low-rank structure (Sussillo and Abbott, 2009).

In this study, we enforced low-dimensional dynamics in RNNs by constraining their connectivity matrix to be of a minimal rank compatible with the examined task (Dubreuil et al., 2020). In addition to providing a principled approach for generating low-dimensional dynamics, this class of low-rank RNNs has the added benefit of being mathematically tractable (Mastrogiuseppe and Ostojic, 2018; Schuessler et al., 2020a; Beiran et al., 2021), thereby facilitating the interpretation of underlying dynamical and computational mechanisms (Dubreuil et al., 2020). We contrasted these low-rank networks with generic RNNs in which the connectivity matrix was not explicitly constrained. A prominent observation is that training unconstrained RNNs on timing tasks leads to a broad variety

of solutions (Turner et al., 2021) (Fig. S8). Overall, we found that unconstrained RNNs developed higher dimensional dynamics than low-rank RNNs (Fig. 2 B) and did not extrapolate outside of their training range (Fig. 2 C, Fig. S8). While higher than in low-rank networks, the dimensionality of dynamics in unconstrained networks was still relatively low, and training such networks in fact often leads to low-rank structure in the connectivity (Schuessler et al., 2020b). It is therefore possible that variations in the training details, and in particular specific types of regularization (Sussillo et al., 2015), may lead to lower-dimensional dimensional dynamics that induce better generalization in full rank networks. From that point of view, constraining the networks to be of minimal rank can be seen as a particular type of regularization, and our results show that such a regularization enables extrapolation by enforcing a smooth activity manifold across conditions.

Given a set of examples, how far artificial neural networks extrapolate out of the training distribution depends on a number of factors such as the details of the architecture, loss function and regularization. In contrast, humans and other animals readily generalize from a small number of examples (Lake et al., 2015). Biological brains are therefore believed to implement specific *inductive biases*, i.e. additional constraints that favor fast learning and generalization (Sinz et al., 2019; Bordelon and Pehlevan, 2021). These inductive biases may originate from developmental constraints optimized over evolution (Zador, 2019), learning multiple tasks in the same network (Freedman and Assad, 2006; Driscoll et al., 2017; Yang et al., 2019) or other imperatives, but their precise form remains to be fully determined. Our results bring forward the possibility that low-dimensional neural dynamics may form one type of implicit bias that facilitates extrapolation in flexible timing, a fundamental aspect of behavior (Paton and Buonomano, 2018).

Generalization is an extremely broad phenomenon, and an overarching proposal is that it can be conceptualized in terms of learning internal models of the environment and other individuals (Lake et al., 2017). Our analyses suggest that low-dimensional networks learn simple parametric models of the dynamics, where the relevant parameter is controlled by a contextual cue. We expect these results to extend to other parametric forms of generalization (Rajalingham et al., 2021), and provide a fundamental building block for neural networks that implement more complex internal models of the external world.

Acknowledgements

MB, MJ and SO were supported by the CRCNS project PIND funded through the National Institute of Health (NIMH: 1R01MH122025-01) and French Agence Nationale de la Recherche (ANR-19-NEUC-0001-01). MB was supported by the Ecole de Neurosciences de Paris. NM was supported by a Whitaker Health Sciences Fund Fellowship. HS was supported by a BBRF Young Investigator grant. SO was supported by the program "Ecoles Universitaires de Recherche" ANR-17-EURE-0017. MJ was supported by the Simons Foundation, the McKnight-Endowment Fund for Neuroscience and the McGovern Institute. The authors would like to thank Adrian Valente for discussions and the software library for training recurrent neural networks, developed initially for Dubreuil et al. (2020).

Code availability

Code and trained networks will be made available upon publication.

References

Abbott, L. F., Rajan, K., and Sompolinsky, H. (2011). Interactions between Intrinsic and Stimulus-Evoked Activity in Recurrent Neural Networks. In *The Dynamic Brain: An Exploration of Neuronal Variability and Its Functional Significance*. Oxford University Press.

Badre, D., Bhandari, A., Keglovits, H., and Kikumoto, A. (2021). The dimensionality of neural representations for control. *Current Opinion in Behavioral Sciences*, 38:20–28.

Beiran, M., Dubreuil, A., Valente, A., Mastrogiuseppe, F., and Ostojic, S. (2021). Shaping Dynamics With Multiple Populations in Low-Rank Recurrent Networks. *Neural Computation*, 33(6):1572–1615.

Bernardi, S., Benna, M. K., Rigotti, M., Munuera, J., Fusi, S., and Salzman, C. D. (2020). The Geometry of Abstraction in the Hippocampus and Prefrontal Cortex. *Cell*, 183(4):954–967.e21.

Bi, Z. and Zhou, C. (2020). Understanding the computation of time using neural network models. *Proceedings of the National Academy of Sciences of the United States of America*, 117(19):10530–10540.

Bordelon, B. and Pehlevan, C. (2021). Population Codes Enable Learning from Few Examples By Shaping Inductive Bias. *bioRxiv*, page 2021.03.30.437743.

Bouchacourt, F., Palminteri, S., Koechlin, E., and Ostojic, S. (2020). Temporal chunking as a mechanism for unsupervised learning of task-sets. *eLife*, 9.

Cayco-Gajic, N. A. and Silver, R. A. (2019). Re-evaluating Circuit Mechanisms Underlying Pattern Separation. *Neuron*, 101(4):584–602.

Chung, S. and Abbott, L. F. (2021). Neural population geometry: An approach for understanding biological and artificial neural networks. *arXiv*, 2104.07059.

Chung, S., Lee, D. D., and Sompolinsky, H. (2018). Classification and Geometry of General Perceptual Manifolds. *Physical Review X*, 8(3).

Cueva, C. J., Saez, A., Marcos, E., Genovesio, A., Jazayeri, M., Romo, R., Salzman, C. D., Shadlen, M. N., and Fusi, S. (2020). Low-dimensional dynamics for working memory and time encoding. *Proceedings of the National Academy of Sciences of the United States of America*, 117(37):23021–23032.

Darshan, R. and Rivkind, A. (2021). Learning to represent continuous variables in heterogeneous neural networks. *bioRxiv*, page 2021.06.01.446635.

DiCarlo, J. J. and Cox, D. D. (2007). Untangling invariant object recognition. *Trends in Cognitive Sciences*, 11(8):333–341.

DiCarlo, J. J., Zoccolan, D., and Rust, N. C. (2012). How does the brain solve visual object recognition? *Neuron*, 73(3):415–434.

Driscoll, L. N., Pettit, N. L., Minderer, M., Chettih, S. N., and Harvey, C. D. (2017). Dynamic Reorganization of Neuronal Activity Patterns in Parietal Cortex. *Cell*, 170(5):986–999.e16.

Dubreuil, A., Valente, A., Beiran, M., Mastrogiuseppe, F., and Ostojic, S. (2020). Complementary roles of dimensionality and population structure in neural computations. *bioRxiv*, page 2020.07.03.185942.

Egger, S. W., Remington, E. D., Chang, C. J., and Jazayeri, M. (2019). Internal models of sensorimotor integration regulate cortical dynamics. *Nature Neuroscience*, 22(11):1871–1882.

Flesch, T., Juechems, K., Dumbalska, T., and Saxe, A. (2021). Rich and lazy learning of task representations in brains and neural networks. *Bioarxiv*, page 2021.04.23.441128.

Freedman, D. J. and Assad, J. A. (2006). Experience-dependent representation of visual categories in parietal cortex. *Nature*, 443(7107):85–88.

Fusi, S., Miller, E. K., and Rigotti, M. (2016). Why neurons mix: High dimensionality for higher cognition. *Current Opinion in Neurobiology*, 37:66–74.

Gallego, J. A., Perich, M. G., Miller, L. E., and Solla, S. A. (2017). Neural Manifolds for the Control of Movement. *Neuron*, 94(5):978–984.

Gámez, J., Mendoza, G., Prado, L., Betancourt, A., and Merchant, H. (2019). The amplitude in periodic neural state trajectories underlies the tempo of rhythmic tapping. *PLoS Biology*, 17(4):e3000054.

Gao, P., Ganguli, S., Battaglia, F. P., and Schnitzer, M. J. (2015). On simplicity and complexity in the brave new world of large-scale neuroscience This review comes from a themed issue on Large-scale recording technology. *Current Opinion in Neurobiology*, 32:148–155.

Gao, P., Trautmann, E., Yu, B., Santhanam, G., Ryu, S., Shenoy, K., and Ganguli, S. (2017). A theory of multineuronal dimensionality, dynamics and measurement. *bioRxiv*, page 214262.

Jazayeri, M. and Ostojic, S. (2021). Interpreting neural computations by examining intrinsic and embedding dimensionality of neural activity. *Current Opinion in Neurobiology*, 70:113–120.

Jazayeri, M. and Shadlen, M. N. (2010). Temporal context calibrates interval timing. *Nature Neuroscience*, 13(8):1020–1026.

Jazayeri, M. and Shadlen, M. N. (2015). A Neural Mechanism for Sensing and Reproducing a Time Interval. *Current Biology*, 25(20):2599–2609.

Kim, J. Z., Lu, Z., Nozari, E., Pappas, G. J., and Bassett, D. S. (2021). Teaching recurrent neural networks to infer global temporal structure from local examples. *Nature Machine Intelligence*, 3(4):316–323.

Kingma, D. P. and Ba, J. L. (2015). Adam: A method for stochastic optimization. *arXiv*, 1412.6980.

Klos, C., Kalle Kossio, Y. F., Goedeke, S., Gilra, A., and Memmesheimer, R. M. (2020). Dynamical Learning of Dynamics. *Physical Review Letters*, 125(8).

Laje, R. and Buonomano, D. V. (2013). Robust timing and motor patterns by taming chaos in recurrent neural networks. *Nature Neuroscience*, 16(7):925–933.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40.

Litwin-Kumar, A., Harris, K. D., Axel, R., Sompolinsky, H., and Abbott, L. F. (2017). Optimal Degrees of Synaptic Connectivity. *Neuron*, 93(5):1153–1164.e7.

Mante, V., Sussillo, D., Shenoy, K. V., and Newsome, W. T. (2013). Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, 503(7474):78–84.

Markman, E. M. (1989). *Categorization and Naming in Children*. MIT Press.

Mastrogiuseppe, F. and Ostojic, S. (2018). Linking Connectivity, Dynamics, and Computations in Low-Rank Recurrent Neural Networks. *Neuron*, 99(3):609–623.e29.

Meirhaeghe, N., Sohn, H., and Jazayeri, M. (2021). A precise and adaptive neural mechanism for predictive temporal processing in the frontal cortex. *Neuron*, 109(18):2995–3011.e5.

Mello, G. B., Soares, S., and Paton, J. J. (2015). A scalable population code for time in the striatum. *Current Biology*, 25(9):1113–1122.

Merchant, H. and Averbeck, B. B. (2017). The computational and neural basis of rhythmic timing in medial premotor cortex. *Journal of Neuroscience*, 37(17):4552–4564.

Nogueira, R., Rodgers, C. C., Bruno, R. M., and Fusi, S. (2021). The geometry of cortical representations of touch in rodents. *bioRxiv*, page 2021.02.11.430704.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., Facebook, Z. D., Research, A. I., Lin, Z., Desmaison, A., Antiga, L., Srl, O., and Lerer, A. (2017). Automatic differentiation in PyTorch. In *Advances in Neural Information Processing Systems*, pages 8024–8035.

Paton, J. J. and Buonomano, D. V. (2018). The Neural Basis of Timing: Distributed Mechanisms for Diverse Functions. *Neuron*, 98(4):687–705.

Pereira, U. and Brunel, N. (2018). Attractor Dynamics in Networks with Learning Rules Inferred from In Vivo Data. *Neuron*, 99(1):227–238.e4.

Pollock, E. and Jazayeri, M. (2020). Engineering recurrent neural networks from task-relevant manifolds and dynamics. *PLoS Computational Biology*, 16(8 August):e1008128.

Rabinovich, M., Huerta, R., and Laurent, G. (2008a). Transient dynamics for neural processing. *Science*, pages 48–50.

Rabinovich, M. I., Huerta, R., Varona, P., and Afraimovich, V. S. (2008b). Transient cognitive dynamics, metastability, and decision making. *PLoS Computational Biology*, 4(5):e1000072.

Rajalingham, R., Piccato, A., and Jazayeri, M. (2021). The role of mental simulation in primate physical inference abilities. *bioRxiv*, page 2021.01.14.426741.

Remington, E. D., Egger, S. W., Narain, D., Wang, J., and Jazayeri, M. (2018a). A Dynamical Systems Perspective on Flexible Motor Timing. *Trends in Cognitive Sciences*, 22(10):938–952.

Remington, E. D., Narain, D., Hosseini, E. A., and Jazayeri, M. (2018b). Flexible Sensorimotor Computations through Rapid Reconfiguration of Cortical Dynamics. *Neuron*, 98(5):1005—-1019.e5.

Rigotti, M., Barak, O., Warden, M. R., Wang, X. J., Daw, N. D., Miller, E. K., and Fusi, S. (2013). The importance of mixed selectivity in complex cognitive tasks. *Nature*, 497(7451):585–590.

Rigotti, M., Ben Dayan Rubin, D., Morrison, S. E., Salzman, C. D., and Fusi, S. (2010). Attractor concretion as a mechanism for the formation of context representations. *NeuroImage*, 52(3):833–847.

Saez, A., Rigotti, M., Ostojic, S., Fusi, S., and Salzman, C. D. (2015). Abstract Context Representations in Primate Amygdala and Prefrontal Cortex. *Neuron*, 87(4):869–881.

Saxe, A., Nelli, S., and Summerfield, C. (2021). If deep learning is the answer, what is the question? *Nature Reviews Neuroscience*, 22(1):55–67.

Saxena, S. and Cunningham, J. P. (2019). Towards the neural population doctrine. *Current Opinion in Neurobiology*, 55:103–111.

Schuessler, F., Dubreuil, A., Mastrogiuseppe, F., Ostojic, S., and Barak, O. (2020a). Dynamics of random recurrent networks with correlated low-rank structure. *Physical Review Research*, 2(1):013111.

Schuessler, F., Mastrogiuseppe, F., Dubreuil, A., Ostojic, S., and Barak, O. (2020b). The interplay between randomness and structure during learning in RNNs. In *Advances in Neural Information Processing Systems*, volume 33.

Sinz, F. H., Pitkow, X., Reimer, J., Bethge, M., and Tolias, A. S. (2019). Engineering a less artificial intelligence. *Neuron*, 103(6):967–979.

Sohn, H., Narain, D., Meirhaeghe, N., and Jazayeri, M. (2019). Bayesian Computation through Cortical Latent Dynamics. *Neuron*, 103(5):934—-947.e5.

Susman, L., Mastrogiuseppe, F., Brenner, N., and Barak, O. (2021). Quality of internal representation shapes learning performance in feedback neural networks. *Physical Review Research*, 3(1):013176.

Sussillo, D. and Abbott, L. (2009). Generating Coherent Patterns of Activity from Chaotic Neural Networks. *Neuron*, 63(4):544–557.

Sussillo, D. and Barak, O. (2013). Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Computation*, 25(3):626–649.

Sussillo, D., Churchland, M. M., Kaufman, M. T., and Shenoy, K. V. (2015). A neural network that finds a naturalistic solution for the production of muscle activity. *Nature Neuroscience*, 18(7):1025–1033.

Turner, E., Dabholkar, K. V., and Barak, O. (2021). Charting and navigating the space of solutions for recurrent neural networks. In *Advances in Neural Information Processing Systems*.

Urai, A. E., Doiron, B., Leifer, A. M., and Churchland, A. K. (2021). Large-scale neural recordings call for new insights to link brain and behavior. *arXiv*, 2103.14662.

Vyas, S., Golub, M. D., Sussillo, D., and Shenoy, K. (2020). Computation Through Neural Population Dynamics. *Annual Review of Neuroscience*, 43:249–275.

Wang, J., Narain, D., Hosseini, E. A., and Jazayeri, M. (2018). Flexible timing by temporal scaling of cortical responses. *Nature Neuroscience*, 21(1):102–112.

Werbos, P. J. (1990). Backpropagation Through Time: What It Does and How to Do It. *Proceedings of the IEEE*, 78(10):1550–1560.

Yang, G. R., Cole, M. W., and Rajan, K. (2019). How to study the neural mechanisms of multiple tasks. *Current Opinion in Behavioral Sciences*, 29:134–143.

Zador, A. M. (2019). A critique of pure learning and what artificial neural networks can learn from animal brains. *Nature Communications*, 10(1):1–7.

4 Methods

Recurrent neural network dynamics We trained recurrent neural networks (RNNs) consisting of $N = 1000$ units with dynamics given by Eq. (1). We simulated the network dynamics by applying Euler's method with a discrete time step Δt. The noise source $\eta_i(t)$ was generated by drawing values from a zero-mean Gaussian distribution at every time step.

The readout of the network was defined as

$$z(t) = \sum_{i=1}^{N} w_i \phi(x_i(t)), \tag{5}$$

a linear combination of the firing rates of all network units, along the vector $\boldsymbol{w} = \{w_i\}_{i=1\ldots N}$.

We considered networks with constrained low-rank connectivity as well as unconstrained networks. In networks of constrained rank R, the connectivity matrix was defined as the sum of R rank-one matrices

$$J_{ij} = \frac{1}{N} \sum_{r=1}^{R} m_i^{(r)} n_j^{(r)}. \tag{6}$$

We refer to vectors $\boldsymbol{m}^{(r)} = \{m_i^{(r)}\}_{i=1\ldots N}$ and $\boldsymbol{n}^{(r)} = \{n_i^{(r)}\}_{i=1\ldots N}$ as the $r-$th left and right connectivity vectors for $r = 1 \ldots R$.

Training Networks were trained using backpropagation-through-time (Werbos, 1990) to minimize the loss function defined by the squared difference between the readout $z_q(t)$ of the network on trial q and the target output $\hat{z}_q(t)$ for that trial. The loss function was written as

$$\mathcal{L} = \sum_q \sum_{t_1^{(q)} < t < t_2^{(q)}} \left(z_q(t) - \hat{z}_q(t)\right)^2 \tag{7}$$

where q runs over different trials, and $t_1^{(q)}$ and $t_2^{(q)}$ correspond to the time boundaries taken into account for computing the loss function (specified in task definitions below).

The network parameters trained by the algorithm were the components of input vectors $\boldsymbol{I^{(s)}}$, the readout vector $\boldsymbol{w}$, the initial network state at the beginning of each trial $\boldsymbol{x}(t = 0)$, and the connectivity. In networks with constrained rank R, we directly trained the components of the connectivity vectors $\boldsymbol{m}^{(r)}$ and $\boldsymbol{n}^{(r)}$, i.e. a total of $2R \times N$ parameters. In networks with unconstrained rank, the N^2 connectivity strengths J_{ij} were trained.

We used 500 trials for each training set, and 100 trials for each test set. Following Dubreuil et al. (2020), we used the ADAM optimizer (Kingma and Ba, 2015) in pytorch (Paszke et al., 2017) with decay rates of the first and second moments of 0.9 and 0.999, and learning rates varying between 10^{-4} and 10^{-2}. The remaining parameters for training RNNs are listed in Table 1. The training code was based on the software library developed for training low-rank networks in Dubreuil et al. (2020).

In networks with constrained rank, we initialized the connectivity vectors using random Gaussian variables of unit variance and zero-mean. The covariance between components of different connectivity vectors at the beginning of training was defined as

$$\sigma_{m_r n_s} = \sigma_0 \delta_{rs}, \tag{8}$$

where $\sigma_0 = 0.8$ and δ_{rs} is the Kronecker delta function. These initial covariances between connectivity vectors generated collective activity effectively that was slower than the membrane time constant,

number of units N	1000
single-unit τ	100 ms
standard deviation η_i	0.08
integration step Δt	10 ms
initial g_0	0.8
initial σ_0	0.8

Table 1: Parameters for trained RNNs

which was useful to propagate errors back in time during learning (Schuessler et al., 2020b). In networks where the rank was not constrained, the initial connectivity strengths J_{ij} were drawn from a Gaussian distribution with zero mean and variance g_0/N^2. The input and readout vectors were initialized as random vectors with mean zero and unit variance, randomly correlated with the connectivity vectors.

In the MWG task with contextual cue, networks were initialized using solutions from RNNs trained on the MWG task for initialization. We implemented a two-step method for training RNNs on the MWG+Ctxt task. First, only input and output weights were trained. In a second step, we trained both inputs and output together with the recurrent weights.

In networks with constrained rank, the rank R was treated as a hyperparameter of the model. We trained networks with increasing fixed rank, starting from $R = 1$ (see Fig. S1). The minimal rank is defined as the lowest rank R for which the loss is comparable to the loss after training a full-rank network (Dubreuil et al., 2020).

Flexible timing tasks We considered three flexible timing tasks, Cue-Set-Go (CSG, Wang et al. (2018)), Measure-Wait-Go (MWG) and Measure-Wait-Go with context (MWG+ctx). All tasks required producing a time interval t_{out} after a brief input pulse which we denote as 'Set'. In the three tasks, the input pulse 'Set' was defined as an instantaneous pulse along the vector $\boldsymbol{I}_{set}$, and indicated the beginning of the output time interval. The target output interval t_{out} depended on other inputs given to the network, specific to each task and detailed below. The target output in the loss function was designed as a linear ramp (Wang et al., 2018), that started at value -0.5 when the 'Set' signal is received, and grew until the threshold value $+0.5$ (Fig. 1 A-B, Fig. 2 A). The output interval t_{out} was defined as the time elapsed from the time of 'Set' until the time the output reached the threshold value.

The considered time window in the loss function (Eq. 7) included the ramping epoch as well as the 300 ms that preceded and followed the ramp, where the target output was clamped to the initial and final values. For training, we used four target intervals ranging between 800 ms and 1550 ms, about one order of magnitude longer than the membrane time constant of single units. In a small fraction of trials, $p = 0.1$, we omitted the 'Set' signal. In that case, the target output of the network remained at the initial value -0.5.

Cue-Set-Go task. The target output interval t_{out} was indicated by the amplitude of a 'Cue' input presented before the 'Set' signal. The 'Cue' input was constant for each trial and present throughout the whole trial duration, along the spatial vector $\boldsymbol{I}_{cue}$. In each trial, the 'Set' signal was presented at a random time ranging between 400 and 800 ms after trial onset. For training, we used four different cue amplitudes ranging from 0, corresponding to the shortest interval, to 0.25, for the longest interval.

Measure-Wait-Go task. The target output interval t_{out} was indicated by the temporal interval between two pulse inputs along vectors $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$. Following a random delay ranging between 200 and 1500 ms after the second input, the 'Set' input indicated the beginning of the production epoch. Four different input intervals, ranging from 800 ms to 1550 ms were used for training.

Measure-Wait-Go with context. We added to the MWG task a tonic contextual input along $\boldsymbol{I}_{ctxt}$, that was present during the whole trial duration and covaried with the average duration of the target interval. For this variant of the task, we used eight target intervals for training. Four of them ranged between 800 ms and 1550 ms. In trials with those target intervals, the amplitude of the contextual input was $u_{ctxt} = 0$. The other four ranged between 1600 ms and 3100 ms, and the associated amplitude of the contextual input was $u_{ctxt} = 0.1$.

In all tasks the first input pulse was fed to the network at a random point in time between 100 and 600 ms after the beginning of each trial.

Performance measure in timing tasks We summarized the performance in a timing task by the output time interval t_{out} generated by the network in each trial. Networks trained to produce a ramping output from -0.5 to 0.5 do not always reach exactly the target endpoint 0.5, but stay close to this value. To avoid inaccuracies due to this variability, we estimated the produced time interval by setting a threshold slightly lower than the endpoint, at a value of $v = 0.3$. We determined the time t_v elapsed between the 'Set' input and the threshold crossing. The produced time interval in each trial was then estimated as $t_{out} = t_v/v$. In trained networks where the readout activity did not reach the threshold (e.g., Fig. S3), the threshold crossing was estimated as the time in which the readout was the closest value to threshold.

Dimensionality of neural activity For trained networks, we assessed dimensionality (Fig. 2 B) using two complementary approaches: the variance explained by the first principal components, and the participation ratio. For the analysis, we focused on the production epoch, common to all tasks, and defined as the time window between the 'Set' pulse and the threshold crossing. We subsampled the time points for each trial condition so that each of them contributes with the same number of time points.

We applied principal component analysis to the firing rates $\phi(x_i\,(t))$ of the recurrent units for every different trial in a given task. The principal component decomposition quantifies the percentage of variance in the neural signal explained along orthogonal patterns of network activity. Due to the presence of single-unit noise in the RNNs, all principal components explain a fixed fraction of variance in the neural signal. The dimensionality can be defined in practice as the number of principal components necessary to account for a given percentage of the neural signal. Alternatively, the dimensionality of two different RNNs can be compared by comparing the distribution of explained variance across the first principal components as shown in Fig. 2 B.

Additionally, we quantified the dimensionality by means of the *participation ratio* (Abbott et al., 2011; Litwin-Kumar et al., 2017; Susman et al., 2021), defined as:

$$P = \frac{\left(\sum_{i=1}^{N} \lambda_i\right)^2}{\sum_{i=1}^{N} \lambda_i^2}, \tag{9}$$

where λ_i correspond to the eigenvalues of the covariance matrix of the neural signal $\phi(x_i\,(t))$,

$$C_{ij} = \left[\phi(x_i)\phi(x_j)\right]. \tag{10}$$

The square brackets denote the time-average and trial-average over the time window of interest. The participation ratio is an index that not only takes into account the dimensionality of neural trajectories, but also weighs each dimension by the fraction of signal variance explained.

Analysis of trained RNNs For the analysis of trained networks, the dynamics of RNNs with constrained low-rank were reduced to a low-dimensional dynamical system (Eq. 4), as detailed here. For any rank-R RNN, the connectivity matrix $\boldsymbol{J}$ can be decomposed uniquely using singular value decomposition as the sum of R rank-one terms (Eq. 6) where the left (resp. right) connectivity vectors $\left\{\boldsymbol{m}^{(r)}\right\}_{r=1\ldots R}$ (resp. $\left\{\boldsymbol{n}^{(r)}\right\}_{r=1\ldots R}$) are orthogonal to each other.

The dynamics of the network (Eq. 1), written in vector notation, read:

$$\tau\frac{d\boldsymbol{x}}{dt} = -\boldsymbol{x} + \frac{1}{N}\sum_{r=1}^{R}\boldsymbol{m}^{(r)}\boldsymbol{n}^{(r)T}\phi\left(\boldsymbol{x}\right) + \sum_{s=1}^{N_{in}}\boldsymbol{I}^{(s)}u_s\left(t\right) + \boldsymbol{\eta}\left(t\right). \tag{11}$$

We decompose each input vector $\boldsymbol{I}^{(s)}$ into the orthogonal and parallel components to the left connectivity vectors:

$$\boldsymbol{I}^{(s)} = \alpha_{\perp}^{(s)}\boldsymbol{I}_{\perp}^{(s)} + \sum_{r=1}^{R}\alpha_{\parallel,r}\boldsymbol{I}_{\parallel,r}^{(s)}. \tag{12}$$

where the constants $\alpha_{\perp}^{(s)}$ and $\alpha_{\parallel,r}^{(s)}$ for $r = 1,\ldots,R$ and $s = 1,\ldots,N_{in}$ indicate the fraction of the input pattern that correspond to each basis vector.

The vector of collective activity $\boldsymbol{x}\left(t\right)$ was then expressed in the basis given by the left connectivity vectors $\left\{\boldsymbol{m}^{(r)}\right\}_{r=1\ldots R}$ and the orthogonal input vectors components $\boldsymbol{I}_{\perp}^{(s)}$ (Beiran et al., 2021; Dubreuil et al., 2020):

$$\boldsymbol{x}\left(t\right) = \sum_{r=1}^{R}\kappa_r\left(t\right)\boldsymbol{m}^{(r)} + \sum_{s=1}^{N_{in}}v_s\left(t\right)\boldsymbol{I}_{\perp}^{(s)}. \tag{13}$$

The time-dependent variables $\boldsymbol{\kappa} = \left\{\kappa_r\right\}_{r=1\ldots R}$ represent the projection of the activity along the *recurrent subspace* spanned by the recurrent connectivity vectors $\left\{\boldsymbol{m}^{(r)}\right\}_{r=1\ldots R}$, and $\boldsymbol{v} = \left\{v_s\right\}_{s=1\ldots N_{in}}$ represent the projection of the activity along the *input-driven subspace*. Altogether, we refer to subspace spanned by the left connectivity vectors and orthogonal inputs as the *embedding subspace*. The projection of the activity $\boldsymbol{x}\left(t\right)$ along the connectivity vector $\boldsymbol{m}^{(r)}$ was in practice calculated as:

$$\kappa_r\left(t\right) = \frac{\boldsymbol{m}^{(r)T}\boldsymbol{x}\left(t\right)}{\boldsymbol{m}^{(r)T}\boldsymbol{m}^{(r)}}, \tag{14}$$

and similarly for the input variables.

Inserting Eq. (13) in Eq. (11), and separating each term along orthogonal vectors of the embedding space, we obtain a set of differential equations for the recurrent and input-driven variables:

$$\begin{aligned}
\tau\frac{d\kappa_r}{dt} &= -\kappa_r + \frac{1}{N}\boldsymbol{n}^{(r)T}\phi\left(\sum_{r'=1}^{R}\kappa_{r'}\left(t\right)\boldsymbol{m}^{(r')} + \sum_{s=1}^{N_{in}}v_s\left(t\right)\boldsymbol{I}_{\perp}^{(s)}\right) + \sum_{s=1}^{N_{in}}u_s\alpha_{\parallel,r}^{(s)}, \\
\tau\frac{dv_s}{dt} &= -v_s + u_s\alpha_{\perp}^{(s)}.
\end{aligned} \tag{15}$$

This analysis effectively reduces a high-dimensional dynamical system (Eq. 11, N variables) to a lower dimensional dynamical system (Eq. 15, $R + N_{in}$ variables) based on the fact that the recurrent connectivity is rank R.

Note that the input-driven variables $\boldsymbol{v}$ are a temporally filtered version of the input variables $\boldsymbol{u}$ at the single unit time constant τ (Eq. 15). Therefore, pulse-like inputs produce a change in the recurrent variables $\boldsymbol{\kappa}$ at the timescale given by τ. For constant inputs $u\boldsymbol{I}$ with variable amplitude u from trial

to trial, as the cue in the CSG task and context in the MWG+Ctxt task, the effect on the dynamics is twofold. First, varying the amplitude is equivalent to shifting the location of the recurrent subspace to a parallel plane in the embedding subspace, because the input-driven variable v is different for each trial. Secondly, the dynamics of the recurrent variables are also affected by changes in the amplitude. They read:

$$\tau \frac{d\kappa_r}{dt} = -\kappa_r + \frac{1}{N} \boldsymbol{n}^{(r)T} \phi \left(\sum_{r=1}^{R} \kappa^{(r)} m_i^{(r)} + u I_i \right) + u \alpha_{\parallel,r}. \tag{16}$$

To study the dynamical landscape of low-dimensional activity, we define the speed q (Sussillo and Barak, 2013) at a given neural state as a scalar function

$$q = \sqrt{\sum_{i=1}^{N} \left(\frac{dx_i}{dt} \right)^2}. \tag{17}$$

The speed q indicates how fast trajectories evolve at a given point in state space. States $\boldsymbol{\kappa}$ where the speed is zero correspond to fixed points of the RNN.

In full-rank networks, it is a priori not possible to fully describe the trajectories using only a few collective variables. The speed of the dynamics can however still be calculated as in Eq. (17).

Non-linear manifolds A useful approach to analyze the dynamics of low-rank RNNs is to initialize the network at arbitrary initial conditions and visualize the dynamics of the variables $\boldsymbol{\kappa}$ in the recurrent subspace, and as a function of time (Fig. S9). We found that before reaching a stable state trajectories with random initial conditions in trained networks appear to converge to non-linear regions of the recurrent subspace, that we refer to as *neural manifolds*.

We therefore devised methods to identify these non-linear manifolds: one exact method, that we used in practice for rank-two networks, and an approximate method, used for rank $R > 2$. The first method consists of initializing trajectories close to all saddle points of the dynamics. In rank-two networks trained on the CSG task, for instance, there are two saddle points, and initializing two trajectories nearby the two opposite saddle points led to a closed curve to which random trajectories converge (solid red line, Fig. S9 A). The manifolds obtained through this method are closely related to the concept of heteroclinic orbits (Rabinovich et al., 2008b,a).

In rank-three networks trained on the MWG task, randomly initialized trajectories converged to a sphere-like manifold (Fig. S3 A). Determining the manifold starting from non-trivial saddle points would require computing a large number of trajectories, to sample all the possible trajectories on the surface of the sphere-like manifold. We instead used an approximate method for determining the manifold. This method consisted in sampling each radial direction of the recurrent subspace, parametrized by the polar and azimuth angles θ and ϕ, and localizing the non-zero radial distance where the dynamics had minimal speed as defined in Eq. (17) (grey surface, Fig. S9 A; in practice we set a threshold for a minimum distance). Once the manifold was identified, the dynamical landscape on its surface was calculated by locally projecting in two dimensions the vector field on the plane perpendicular to each manifold state. In rank-two networks, the approximate method to determine the manifold led to a curve (red dashed line, Fig. S9 A) which was close to the manifold determined by the first method, in particular at the fixed points, and on the portions of the manifold where the dynamics were slow.

To confirm that the identified manifolds correspond to slow manifolds of the dynamics, we computed the distance of randomly initialized trajectories to the manifold. We found that the distance to the manifold decayed to zero at a timescale given by the membrane time constant (Fig. S9 C).

In contrast, projecting the recurrent variables onto a linear readout, we find that trajectories took a much longer time to converge to a stable fixed point of the dynamics (Fig. S9 B).

Networks of rank R do not necessarily generate manifolds. As a counter-example, Fig. S9 D-F display the dynamics of a rank-two network, that led to only two non-trivial stable fixed point, and no saddle points. In this case, initializing trajectories randomly led to trajectories that approach the stable fixed points along different curves. The dynamics of the projected activity along the readout in this case converged quickly to the stable fixed points.

Mean-field low-rank networks Dynamics in low-rank networks become mathematically tractable in the limit of large networks when the connectivity components of every unit are randomly drawn from a multivariate probability distribution. Here we assumed that the components of connectivity and input vectors were drawn from a zero-mean multivariate Gaussian (Mastrogiuseppe and Ostojic, 2018; Schuessler et al., 2020a; Beiran et al., 2021)

$$m_i^{(1)}, \dots, m_i^{(R)}, n_i^{(1)}, \dots, n_i^{(R)}, I_i^{(1)}, \dots, I_i^{(S)} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{\Sigma}\right), \tag{18}$$

where $\mathbf{\Sigma}$ is the $(2R + N_{in}) \times (2R + N_{in})$ covariance matrix. We introduce the notation $P\left(\underline{m}, \underline{n}, \underline{I}\right) = P\left(m^{(1)}, \dots, m^{(R)}, n^{(1)}, \dots, n^{(R)}, I^{(1)}, \dots, I^{(s)}\right)$ to refer to the joint probability distribution of vector components. Without loss of generality, we fixed the variance of the right connectivity vectors and input vectors to unity; $\sigma^2_{m^{(r)}} = \sigma^2_{I^{(s)}} = 1$. We further assumed that the external input vectors are orthogonal to the left connectivity vectors $\boldsymbol{m}^{(r)}$.

The overlap between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ was defined as their empirical covariance:

$$\hat{\sigma}_{xy} = \frac{1}{N} \sum_{i=1}^{N} \left(x_i - \bar{x}\right)\left(y_i - \bar{y}\right) \tag{19}$$

Here $\bar{x}$ indicates the average value of the components, set to zero in mean-field networks, so that the overlap between two vectors was equivalent to their scalar product. In the large N limit, the overlap between vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ therefore converges to the covariance σ_{xy} between their components. Importantly, as the full covariance matrix $\mathbf{\Sigma}$ needs to be positive-definite, not all its elements σ_{xy} are free parameters.

The parameters that determine the dynamics are then the covariances between left and right connectivity vectors and the covariances between input and left connectivity vectors, as we detail below (see also Beiran et al. (2021)). For that reason, we defined the overlap matrix $\boldsymbol{\sigma_{mn}}$ as the matrix with elements $\sigma_{m^{(r)}n^{(r')}}$, for $r, r' = 1, \dots, R$, and the covariance vector $\boldsymbol{\sigma_{In}}$ between input and right connectivity vectors as the vector with components $\sigma_{In^{(r)}}$. The overlap matrix $\boldsymbol{\sigma_{mn}}$ and vector $\boldsymbol{\sigma_{In}}$ correspond to different subsets of elements of the full covariance matrix $\mathbf{\Sigma}$.

Given these definitions and assuming just one constant input $v\mathbf{I}$, in the limit of large networks $N \to \infty$, the sum over N units in Eq. (15) can be replaced by the expected value over the Gaussian distribution of connectivity and input vectors. The dynamics of the recurrent variables then read:

$$\tau \frac{d\kappa_r}{dt} = -\kappa_r + \int d\underline{m}\, d\underline{n}\, d\underline{I} P\left(\underline{m}, \underline{n}, \underline{I}\right) n^{(r)} \phi\left(\sum_{r'=1}^{R} \kappa_{r'} m^{(r')} + \sum_{s=1}^{N_{in}} v_s I^{(s)}\right). \tag{20}$$

The Gaussian integral in Eq. (20) can be further expressed in terms of the covariances of the probability distribution as (Schuessler et al., 2020a; Beiran et al., 2021; Dubreuil et al., 2020)

$$\tau \frac{d\kappa_r}{dt} = -\kappa_r + \left\langle \phi' \right\rangle \left(\sum_{r'=1}^{R} \sigma_{m^{(r')}n^{(r)}} \kappa_{r'} + \sum_{s=1}^{N_{in}} \sigma_{I^{(s)}n^{(r)}} v_s\right) \tag{21}$$

where the state-dependent gain factor $\langle \phi' \rangle$ was defined as

$$\langle \phi' \rangle = \int \frac{dx}{\sqrt{2\pi}} \exp\left(-x^2/2\right) \phi' \left(x \sqrt{\sum_{r=1}^{R} \kappa_r^2 + \sum_{s=1}^{N_{in}} v_s^2} \right). \tag{22}$$

The dynamics plotted in Fig. 5 were generated directly from Eq. (21).

For constant inputs (as in Fig. 6) we define *non-specific inputs* as inputs that are uncorrelated with the connectivity vectors, so that $\sigma_{In^{(r)}} = 0$ for $r = 1, \ldots, R$. *Subspace-specific inputs* are defined as inputs that are correlated with at least one of the right connectivity vectors $\sum_{r=1}^{R} |\sigma_{In^{(r)}}| > 0$.

Analysis of dynamics in mean-field low-rank networks We first focus on fixed points of the dynamics in Eq. (21) in absence of inputs. The fixed points correspond to values of κ_r for which the r.h.s of Eq. (21) is zero, which leads to:

$$\kappa_r = \langle \phi' \rangle \sum_{r'=1}^{R} \sigma_{m^{(r')} n^{(r)}} \kappa_{r'}. \tag{23}$$

Since the gain $\langle \phi' \rangle$ is implicitly a function of κ_r (Eq. (22)), Eq. (23) is a set of R non-linear equations, the solutions of which depend only on the overlap matrix $\boldsymbol{\sigma_{mn}}$. Mathematical analyses show that non-zero fixed points correspond to states along the eigenvector directions of the overlap matrix $\boldsymbol{\sigma_{mn}}\boldsymbol{\kappa}_r = \lambda_r \boldsymbol{\kappa}_r$, for eigenvectors whose eigenvalues are larger than one, $\lambda_r > 1$ (Schuessler et al., 2020a; Beiran et al., 2021). Only the fixed points corresponding to the largest eigenvalue are stable, while all the other non-trivial fixed points are saddle points (Schuessler et al., 2020a; Beiran et al., 2021).

If the overlap matrix $\boldsymbol{\sigma_{mn}}$ has a degenerate eigenvalue λ_r, with at least two orthogonal eigenvectors $\boldsymbol{\kappa}_{r_1}$ and $\boldsymbol{\kappa}_{r_2}$ (as is the case when the overlap matrix $\boldsymbol{\sigma_{mn}}$ is proportional to the identity matrix), each possible linear combination of the two eigenvectors leads to a fixed point. Consequently, the system displays a continuous set of fixed points with identical stability, symmetrically located around the origin. In rank-two networks, such a degenerate overlap matrix leads to a ring attractor (Fig. 5 B) (Mastrogiuseppe and Ostojic, 2018; Beiran et al., 2021; Darshan and Rivkind, 2021). All fixed points on the ring are marginally stable, as the direction tangent to the ring corresponds to a zero eigenvalue. In rank-three networks, such symmetry in the connectivity leads to a spherical attractor.

In finite networks, however, the overlap matrix $\boldsymbol{\sigma_{mn}}$ is perturbed by the finite-size sampling, so that it is never exactly proportional to the identity matrix. Small perturbations of the overlap matrix break the continuous symmetry of the attractor, generally leading to a pair of stable fixed points and saddle points on the former attractor, but keeping the speed of the dynamics slow along its surface. The location of the fixed points are then given by Eq. (23). If the perturbation of the symmetric overlap matrix has orthogonal eigenvectors, two stable fixed points and two saddle points are generated on the manifold along orthogonal directions (Fig. 5 C). In contrast, if the perturbation has non-orthogonal eigenvectors, the stable fixed points and saddle points are arranged closer to each other along the manifold (Fig. 5 D).

We then studied the effects of the dynamics of mean-field low-rank networks receiving a constant input $v\mathbf{I}$ (Fig. 6). The fixed point equation with an external input reads:

$$\kappa_r = \langle \phi' \rangle \sum_{r'=1}^{R} \sigma_{m^{(r')} n^{(r)}} \kappa_{r'} + v\, \sigma_{In_r}. \tag{24}$$

Tonic inputs affect the dynamics of low-rank networks in two different ways. One effect is given by the second term in Eq. (24), which is additive and directed along the vector $\boldsymbol{\sigma_{In}}$. The second effect corresponds to changes in the gain $\langle \phi' \rangle$ which decreases monotonically towards zero when v is increased (Eq. 22). Such changes correspond to the multiplicative factor in Eq. (24), and depend only on the strength of the input, but not its direction. Non-specific tonic inputs lead only to the second effect because for them $\sigma_{In^{(r)}} = 0$ for $r = 1, \ldots, R$ (Fig. 6 A-B), while subspace-specific inputs combine both effects (Fig. 6 D-E).

Behavioral task Details about the behavioral task in monkeys can be found in Sohn et al. (2019) and Meirhaeghe et al. (2021). Briefly, two macaque monkeys (monkey G and H) performed a time interval reproduction task known as the 'Ready-Set-Go' task (Jazayeri and Shadlen, 2015; Sohn et al., 2019; Meirhaeghe et al., 2021) Each trial began with animals maintaining their gaze on a central fixation point (white circle: diameter 0.5 deg; fixation window: radius 3.5 deg) presented on a black screen. Upon successful fixation, and after a random delay (uniform hazard; mean: 750 ms, min: 500 ms), a peripheral target (white circle: diameter 0.5 deg) was presented 10 degrees left or right of the fixation point and stayed on throughout the trial. After another random delay (uniform hazard; mean: 500 ms, min: 250 ms), the 'Ready' and 'Set' cues (white annulus: outer diameter 2.2 deg; thickness: 0.1 deg; duration: 100 ms) were sequentially flashed around the fixation point. Following 'Set', the animal had to make a proactive saccade (self-initiated Go) toward the peripheral target so that the output interval (t_{out}, between 'Set' and 'Go') matched the input interval (t_{inp}, between 'Ready' and 'Set'). The sample interval, t_s, was sampled from one of two discrete uniform distributions, with 5 values each between 480–800 ms for Short, and 800–1200 ms for Long. The distributions alternated in short blocks of trials (min of 5 trials for G, 3 trials for H, plus a random sample from a geometric distribution with mean 3, capped at 25 trials for G, 20 trials for H), and were indicated to the animal by the color of the fixation point (context cue; red for Short, blue for Long).

The trial was rewarded if the relative error, $|t_{out} - t_{inp}|/t_{inp}$, was smaller than 0.2. If the trial was rewarded, the color of the target turned green, and the amount of juice delivered decreased linearly with the magnitude of the error. Otherwise, the color of the target turned red, and no juice was delivered. The trial was aborted if the animal broke fixation prematurely before 'Set' or did not acquire the target within 3 t_{inp} after 'Set'. After a fixed inter-trial interval, the fixation point was presented again to indicate the start of a new trial. To compensate for lower expected reward rate in the Long context due to longer duration trials (i.e. longer t_{inp} values), we set the inter-trial intervals of the Short and Long contexts to 1220 ms and 500 ms, respectively.

Neural data Details about the neural recordings can be found in Sohn et al. (2019) and Meirhaeghe et al. (2021). In summary, the data were obtained from acute recordings in the dorsomedial frontal cortex (DMFC; n=619 neurons in monkey G, n=542 in monkey H). For the neural analysis, the context dimension was defined as follows: we first computed the trial-averaged firing rates (bin size: 20 ms, Gaussian smoothing kernel SD: 40 ms) between 'Ready' and 'Set' (measurement epoch) for the Short and Long contexts separately. Because the animal did not know beforehand which input interval (t_{inp}) was presented on a given trial, we averaged trials irrespective of the t_{inp} value within each context. The context dimension was defined as the unit vector connecting the state at the time of Ready of the Long condition and the state at the time of Ready of the Short condition. We then projected the Ready state of the Short and Long contexts onto the context dimension (Fig. 7 C, bottom). To compute the neural speed in the measurement epoch for each context, we averaged the Euclidean distance between consecutive states between 'Ready' and 'Set'. We used standard bootstrapping (resampling trials with replacement; N=100 repeats) to generate confidence intervals.

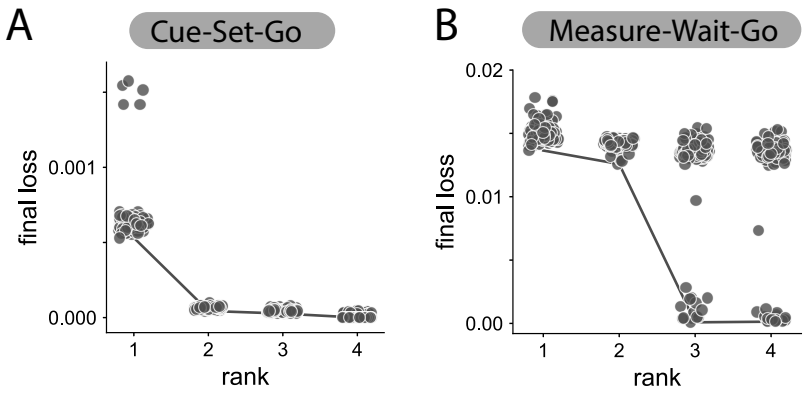
We performed principal component analysis (PCA) to visualize neural trajectories (Fig. 7 A-B, bottom). PC trajectories were obtained by gathering smoothed firing rates in a 2D data matrix where each column corresponded to a neuron, and each row corresponded to a given time point in the measurement epoch. To obtain a common set of principal components for the two contexts (Short and Long), we concatenated the average firing rates of the recorded neurons for the two contexts along the time dimension. We then applied principal component analysis and projected the original data onto the top 3 PCs, which explained about 75% of total variance. Fig. 7 A was obtained by projecting the trajectories onto the top three principal components. Fig. 7 B was obtained by projecting the trajectories onto the first three principal components and then onto the subspace orthogonal to the estimated contextual axis.

Linear decoders For Fig. 7, we used linear decoders based on cross-validated linear discriminant classification, to readout context from neural activity during the pre-stimulus period in the MWG task with contextual cue. The direction of the classifier is estimated as:

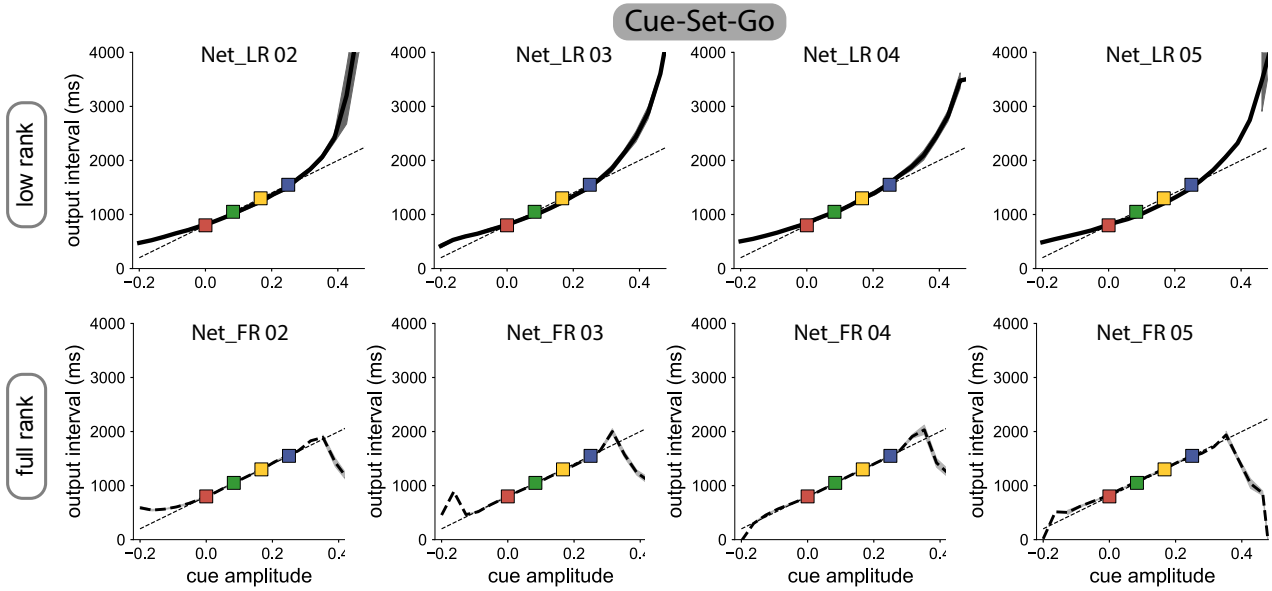
$$\boldsymbol{w_{Ctxt}} = [\boldsymbol{x}\,(t, \text{slow ctxt})] - [\boldsymbol{x}\,(t, \text{fast ctxt})] \tag{25}$$

where the square brackets indicate the average across time points and trials of a given context. In the network model, we set the bias term of the decoder so that the average projected activity of the training set is zero, and normalized it so that the two contexts correspond to projected values -0.5 and 0.5. After the estimation of the decoder, we projected the neural activity of an independent set of trials, different from the training set.

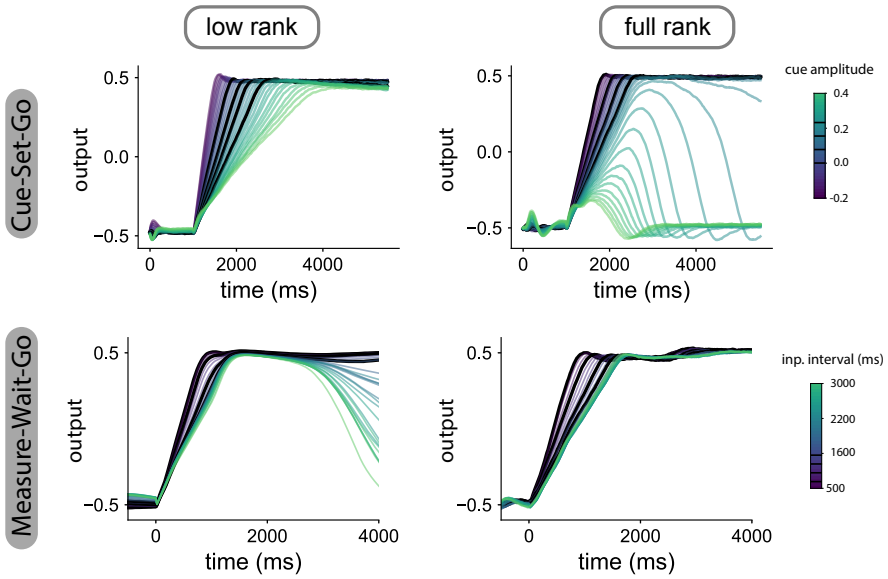
Supplementary Figures



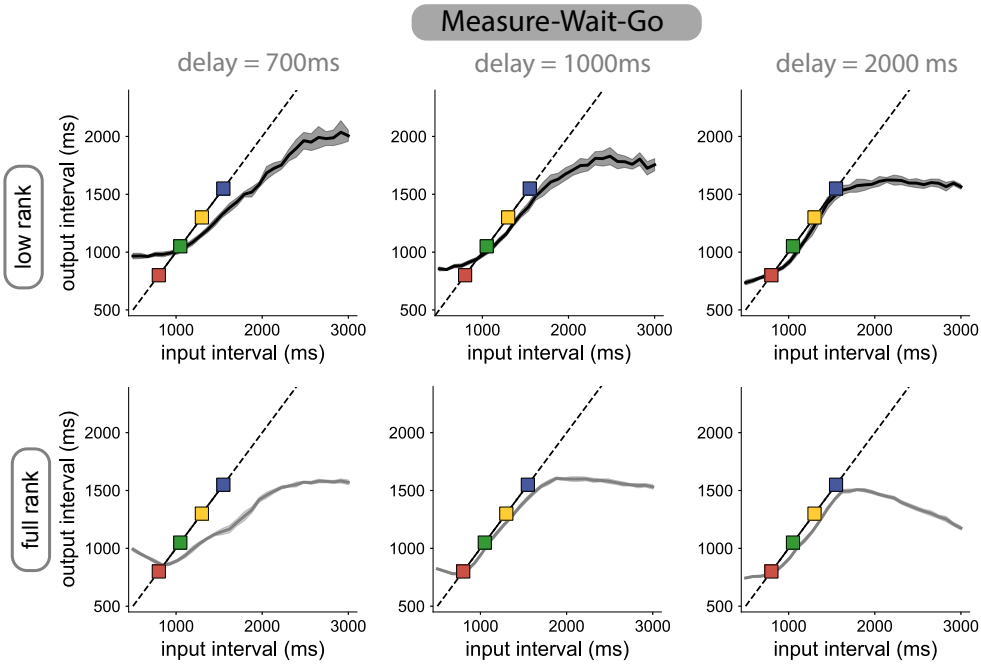
Supplementary Figure 1. Loss (Eq. 7) of trained low-rank networks as a function of the rank of the connectivity matrix. Dots correspond to 100 different RNNs, the line connects the networks with minimal loss. **A** CSG task. The minimal rank for this task is $R = 2$, for which the 100 trained RNNs learned the task. **B** MWG task. The minimal rank is $R = 3$, for which 14 networks learned the task. For rank $R = 4$, 18 networks learned the task. For ranks one and two, no network successfully learned the task.



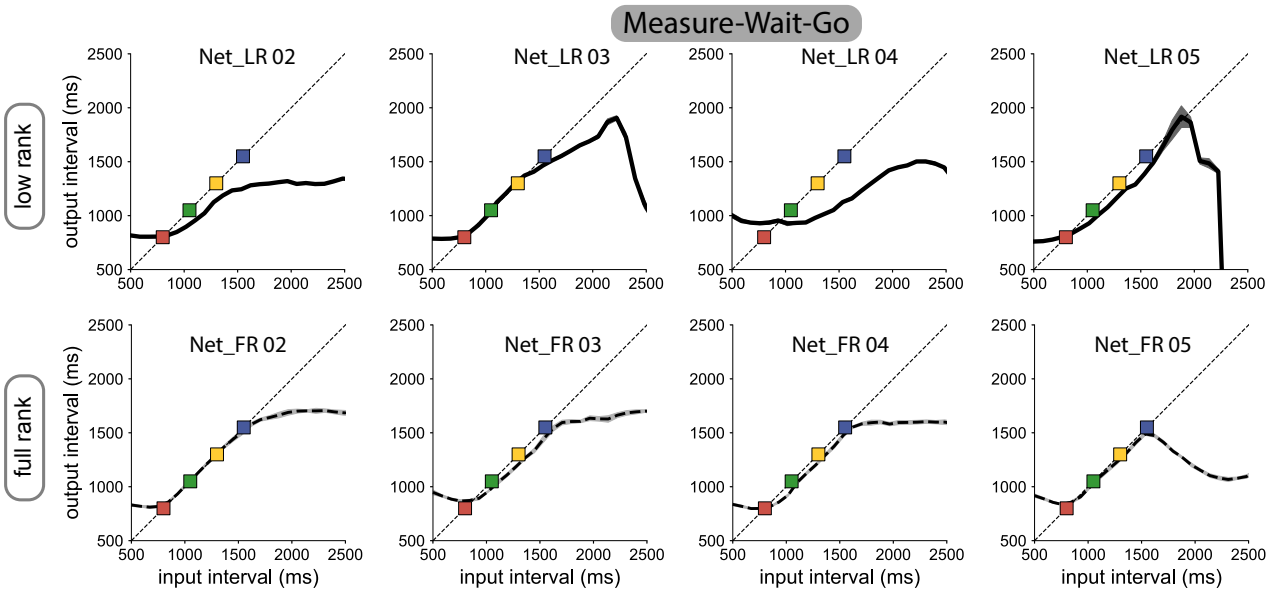
Supplementary Figure 2. Generalization on Cue-Set-Go task for four different trained RNNs, rank-two (top row) and unconstrained full-rank networks (bottom row), similar to Fig. 1 C. Low-rank networks generalize to higher and lower cue amplitudes, by producing longer and shorter intervals respectively, while unconstrained networks do not.



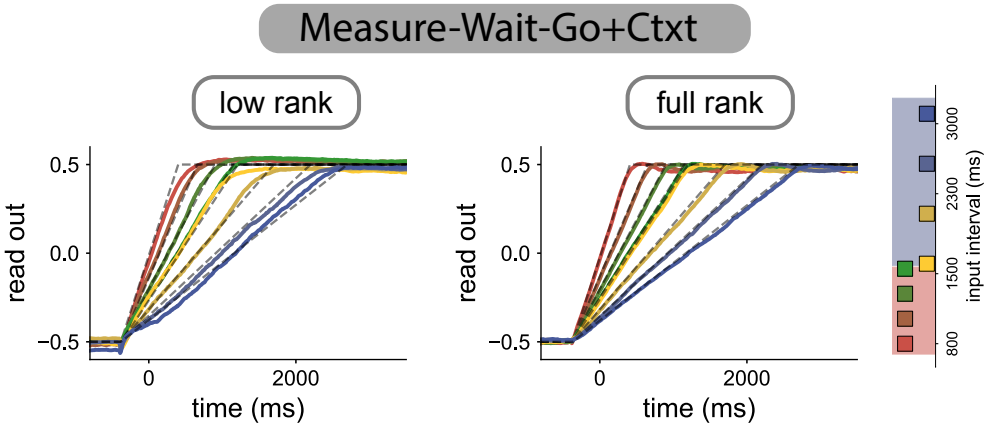
Supplementary Figure 3. Network output on CSG (top row) and MWG tasks (bottom row) for the networks shown in Fig. 2. Black lines correspond to trials used for training. Only low-rank networks trained on the CSG are able to generalize by modifying the slope of the ramping output (top left). Unconstrained networks trained on CSG do not generate a ramping output for stronger cue amplitudes (top right). In the MWG, both low-rank and unconstrained networks do not produce slower ramps than those used during training (bottom plots).



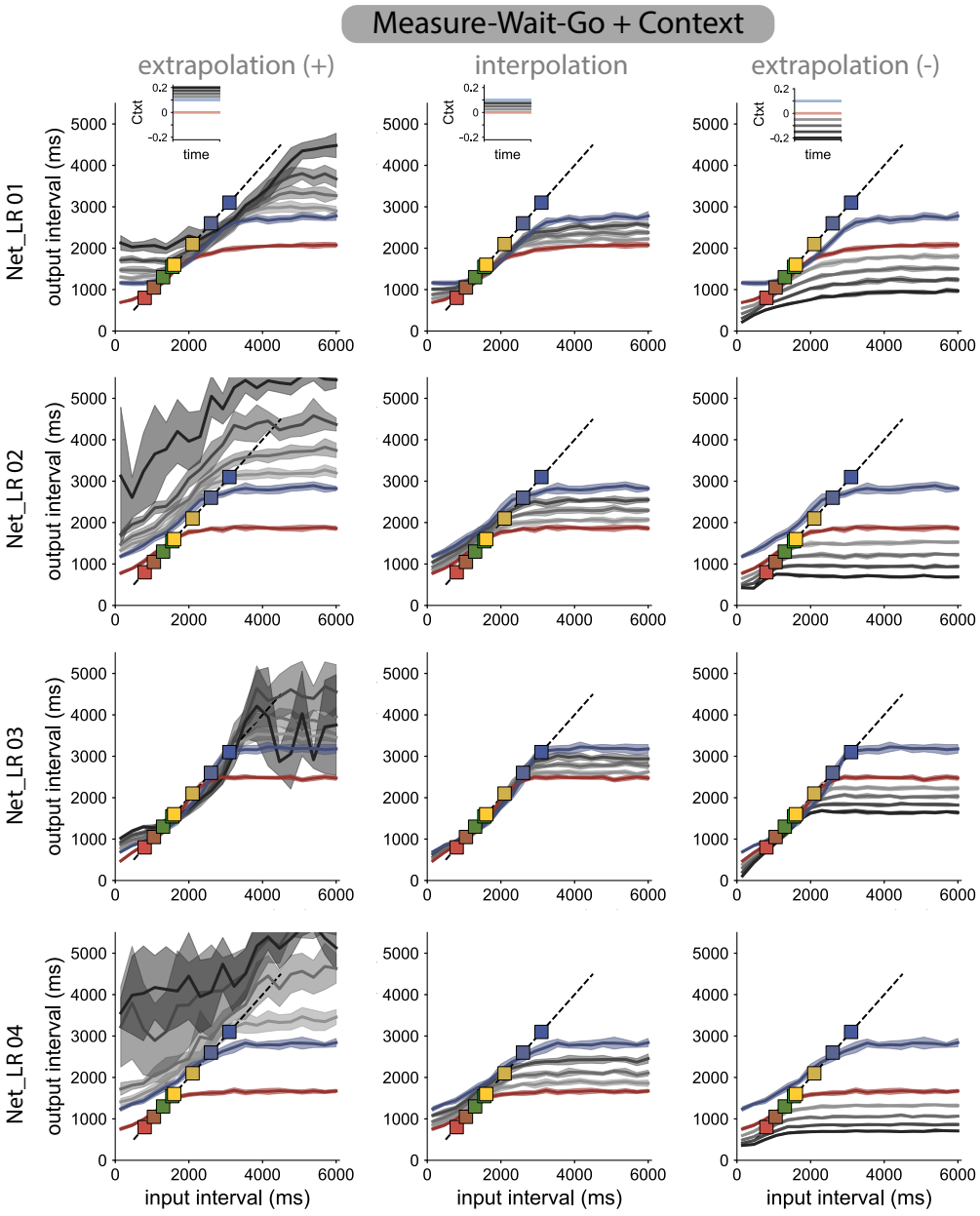
Supplementary Figure 4. Generalization on the Measure-Wait-Go task, as in Fig. 3, for three different lengths of the delay period. The input-output function is not strongly affected by the duration of the delay period.



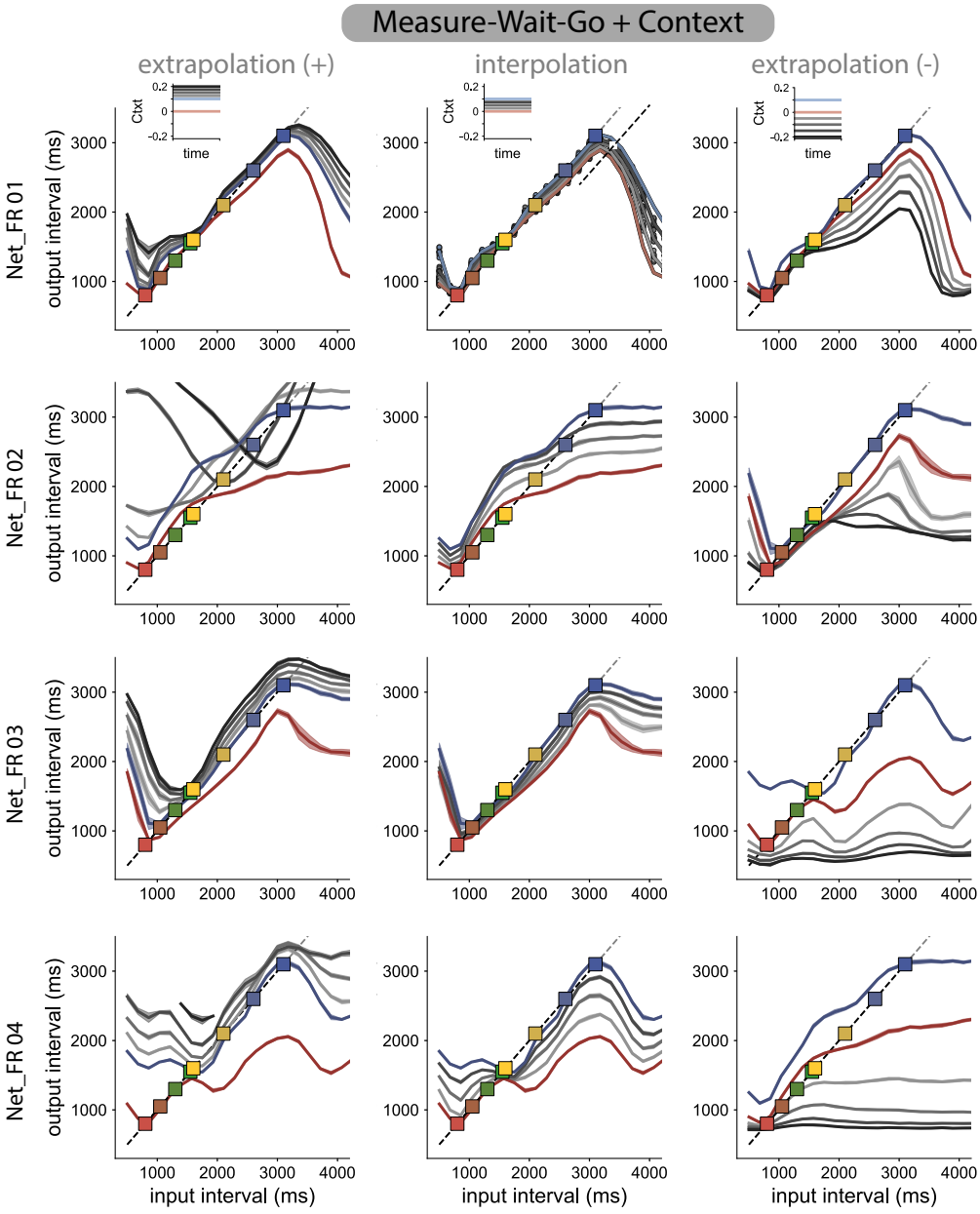
Supplementary Figure 5. Generalization on the Measure-Wait-Go task for four different trained RNNs, low-rank (top row) and unconstrained (bottom row), similar to Fig. 3. Neither low-rank nor unconstrained RNNs produce intervals much longer than those used in training. **A**



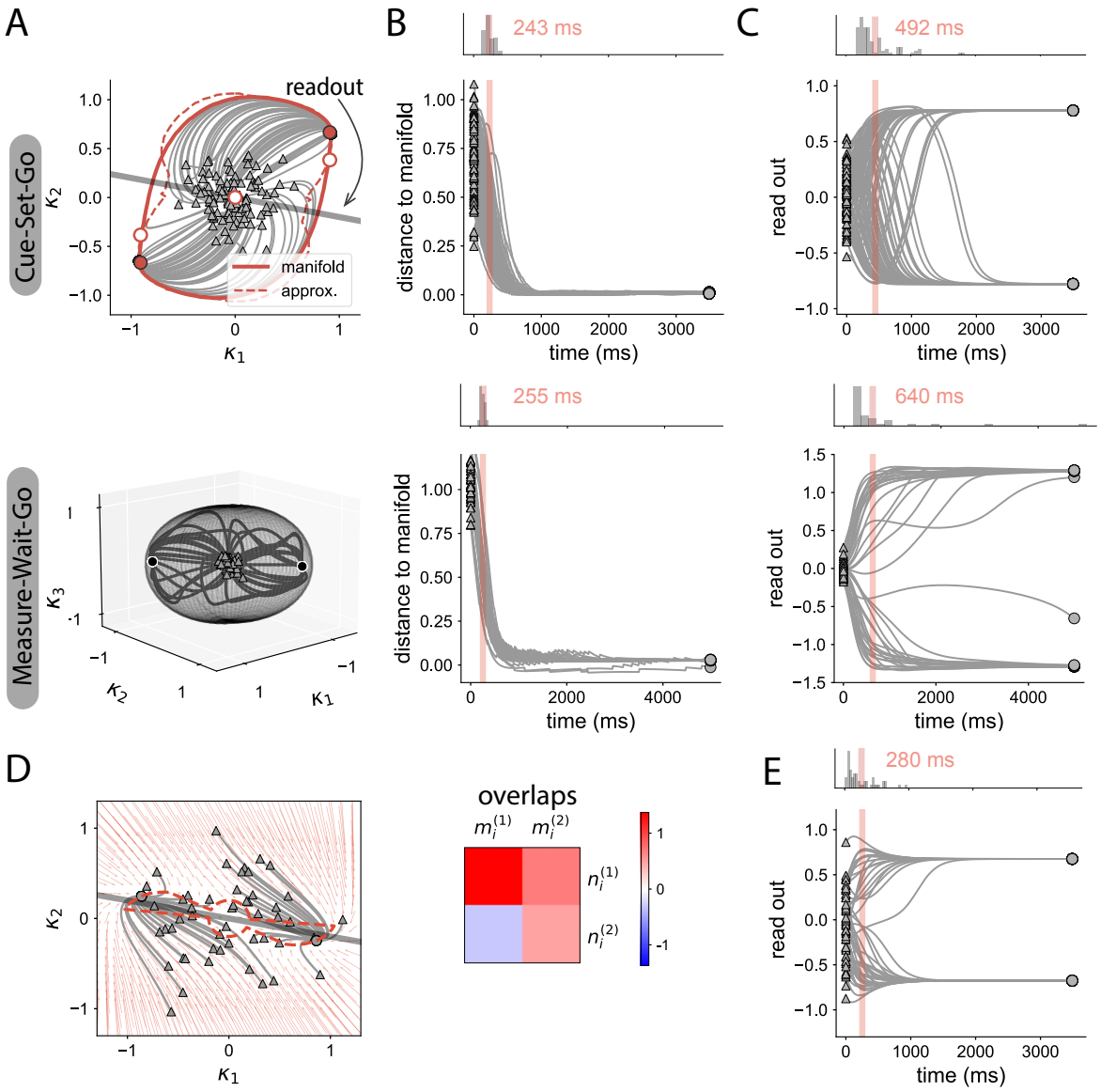
Supplementary Figure 6. Network output on the MWG+Ctxt task in rank-three (left) and unconstrained (right) RNNs. Both networks are able to produce the eight output intervals used during training, four corresponding to each context (right, red and blue colors).



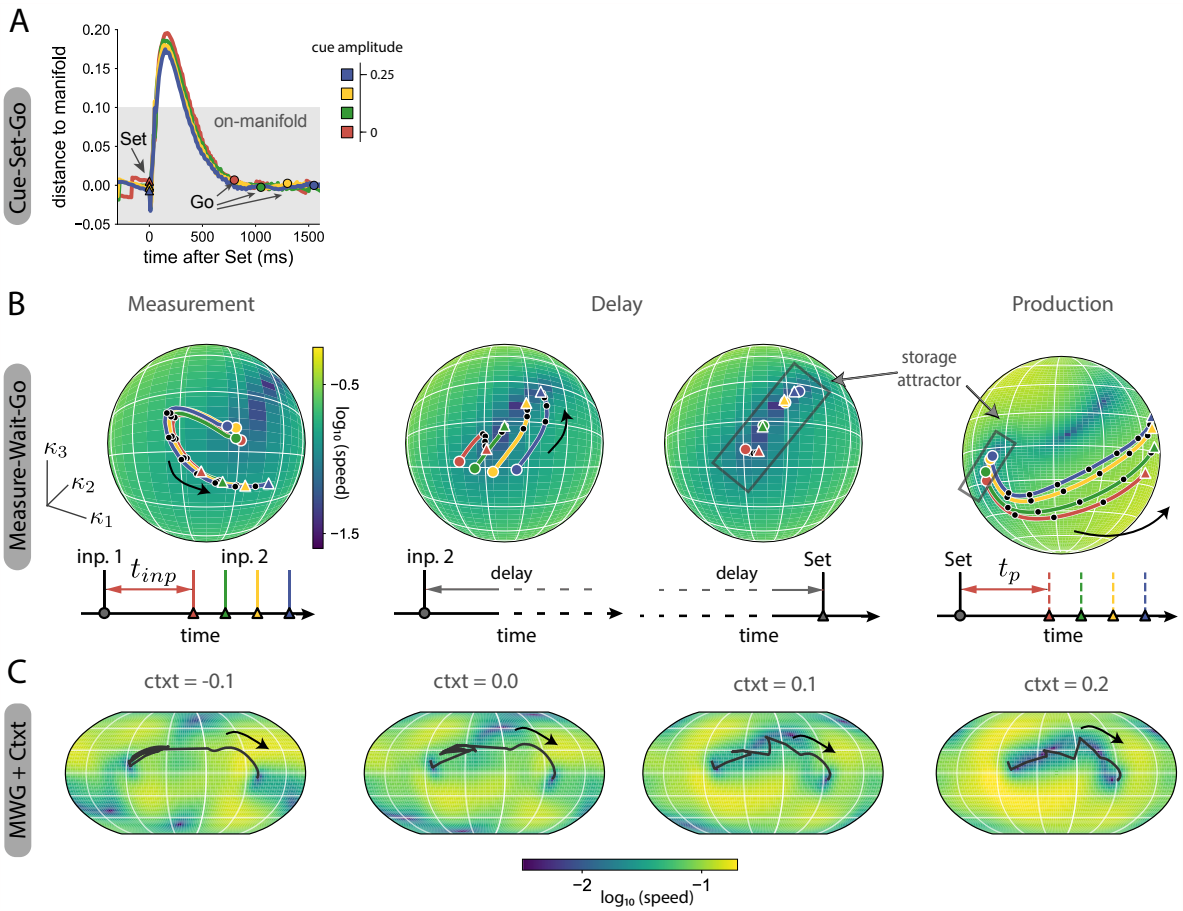
Supplementary Figure 7. Generalization of four trained rank-three RNNs on the MWG+Ctxt task to contexts not seen during training.



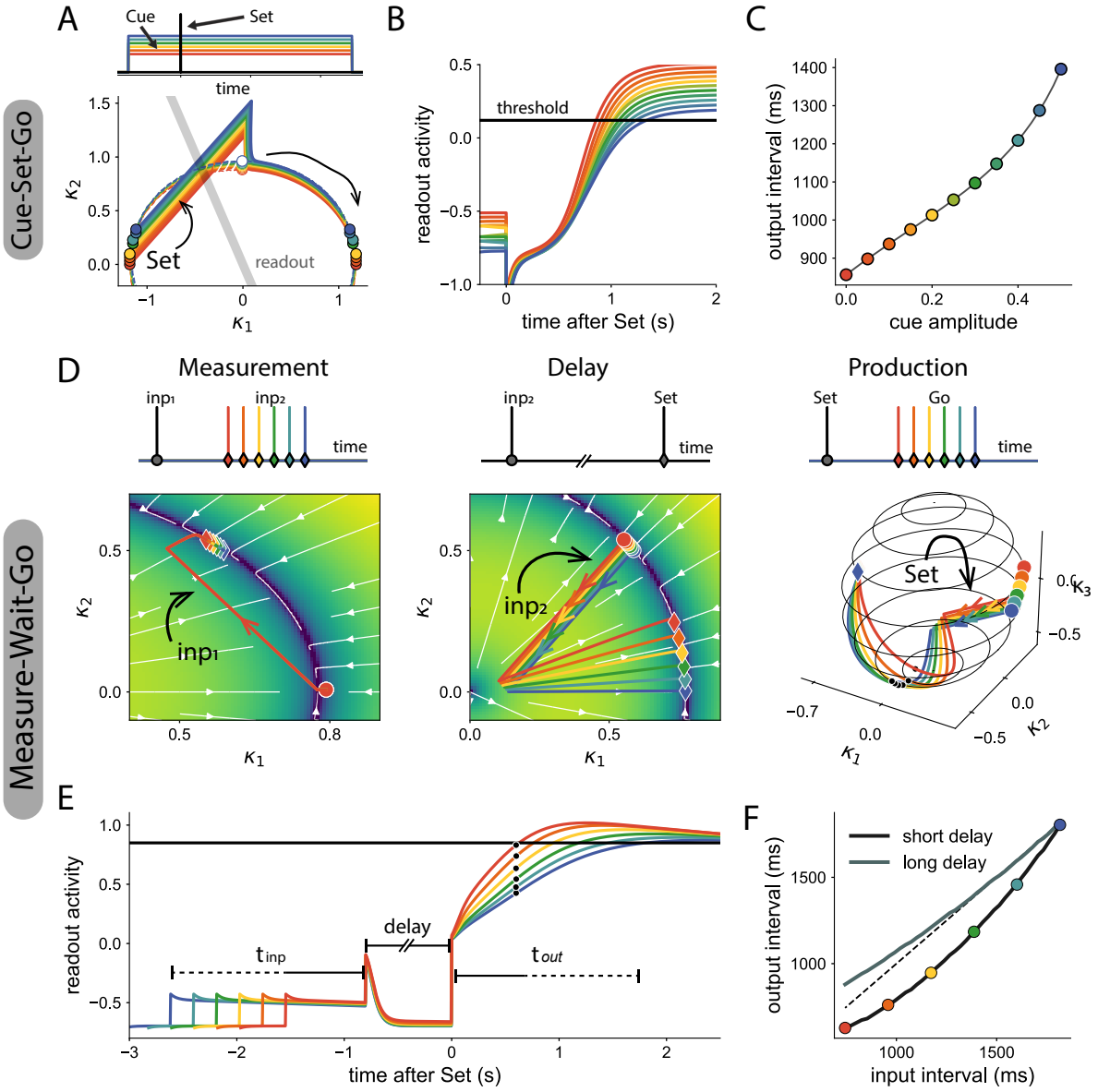
Supplementary Figure 8. Generalization of four trained full-rank RNNs on the MWG+Ctxt task to novel contexts. Although there is a larger variety of solutions, networks do not generalize smoothly to novel contexts. **A**



Supplementary Figure 9. **A** Neural manifolds in low-rank networks trained on the CSG task (top) and MWG task (bottom). Grey lines display trajectories initiated at random states indicated with triangles. White-filled dots are non-stable fixed points, and red/black filled dots are stable fixed points. Top: For the CSG task, the solid red line corresponds to the manifold found by tracking trajectories emanating from saddle points. The red dashed line displays the result of the approximate method for identifying the manifold (see Methods). Bottom: For the MWG task, the spherical-like manifold is shown in grey, determined using the approximate method. **B** Radial distance between randomly initiated trajectories and the manifold. Trajectories converge first to the manifold, and then evolve towards one of the stable fixed points. Red line indicates the average half-time decay from the initial state (see distribution on top inset). **C** Projection of the random trajectories along the readout direction (indicated by grey straight lines in A). Randomly-initialized trajectories converge slowly towards the stable fixed points. **D**-**E** Example of a rank-two network not generating a ring-like manifold. **D** Red arrows correspond to the vector field of the dynamics. Randomly initialized trajectories converge to the non-trivial stable fixed points following different curves. Right: overlap structure between connectivity patterns determining the dynamics. **E** Projection of the random trajectories along the readout direction. Trajectories quickly converge to the non-trivial stable fixed points.



Supplementary Figure 10. **A** Radial distance to the neural manifold of trajectories in a low-rank network trained the CSG task. The 'Set' pulse sends trajectories away from the manifold, but they quickly decay back to it before reaching the final state. **B** Low-dimensional description of neural trajectories in a rank-there network trained on the MWG task. Colored lines are the projections of trajectories on the spherical manifold, corresponding to different input intervals. The colormap on the sphere indicates the speed of the dynamics on the surface of the manifold (blue: slow dynamics, yellow: fast dynamics). Colored dots indicate the state at the beginning of the considered time window, triangles indicate the end of the epoch. Black dots are placed every 200ms. Left: Measurement epoch between the two input pulses. Trajectories are initialized at a non-trivial stable fixed point. The first pulse, indicating the beginning of the measurement epoch, triggers a transient response. The second pulse is received at different states along the evoked transient trajectory, so that the network effectively maps different input intervals to different states on the manifold. Middle panels: activity during the first half (left) and second half (right) of the delay following the second input pulse. Trajectories quickly converge to a slow region of the manifold, where different input intervals are stored in working memory during the remaining delay. The slow region works as a storage attractor: the smallest and largest input intervals stored (red and blue lines) correspond to the extremes of the attractor. Right: Production epoch. The 'Set' pulse sends trajectories from the storage attractor towards the other side of the spherical manifold, along parallel trajectories at different speeds. **C** Speed on the surface of the non-linear manifold for different contextual inputs in the rank-three RNN trained on the MWG+Ctxt task. The black lines corresponds to a trajectory solving the task (input interval of 1500 ms). The region explored by the trajectory of the manifold becomes slower as the context input is increased.



Supplementary Figure 11. Mean-field low-rank RNNs designed for solving the CSG (**A** - **C**) and MWG (**D** - **F**) tasks. The overlaps between connectivity vectors, and the overlaps of input vectors are chosen by hand to reproduce the mechanisms observed in trained networks. **A** Trajectories solving the CSG task in a rank-two network. The amplitude of the input cue modulates the speed on an invariant manifold (colored dashed lines). Trajectories are initialized at one stable fixed point on the manifold (left, colored dots). The 'Set' pulse sends trajectories beyond the saddle point, so that trajectories, after quickly converging to the manifold, evolve at different speeds towards the stable fixed point. Grey line: readout direction. **B** The projection of the different trajectories on the output direction generates a ramping signal with different speeds. The output time interval corresponds to the crossing of a fixed threshold. **C** Relation between cue amplitudes and output intervals in the designed model. Different cue amplitudes effectively modulate the output interval, as required by the task.

Figure S11 *(previous page)*: **D** Trajectories of the designed solution for the MWG task. During the measurement and delay epochs (left, middle), the trajectories stay on the $\kappa_1-\kappa_2$ plane. The rank-three connectivity generates a sphere-like manifold. Within the ring of the manifold on the $\kappa_1 - \kappa_2$ plane, there are stable fixed points with very slow dynamics in their surroundings, that can be used for storing the signal. Left: during the measurement epoch, the first input initiates a slow transient trajectory. Middle: At the beginning of the delay, the trajectories corresponding to different input intervals decay to different states in the vicinity of the stable fixed point, and remain mostly unchanged during the rest of the delay period. Right: the 'Set' pulse sends trajectories towards the other side of the manifold. The trajectories evolve at different speeds depending on their location on the manifold during the delay period. **E** Readout activity for the different input intervals. A ramping signal with different slopes is produced after the 'Set' pulse. The output time interval corresponds to the crossing of a fixed threshold. **F** Relation between input interval and output interval for two different delays. The network produces different intervals based on the estimated input interval. Parameters are available in shared code.