Neuron

Parametric control of flexible timing through low-dimensional neural manifolds

Highlights

- Dimensionality-constrained neural networks generalize to novel stimuli in timing tasks
- Tonic inputs adjust speed of dynamics while preserving low-dimensional structure
- Input control explains geometry of neural population activity in frontal cortex
- Flexible adjustment of control inputs enables fast adaptation to new environments

Authors

Manuel Beiran, Nicolas Meirhaeghe, Hansem Sohn, Mehrdad Jazayeri, Srdjan Ostojic

Correspondence

mjaz@mit.edu (M.J.), srdjan.ostojic@ens.psl.eu (S.O.)

In brief

Beiran et al. investigate how neuronal networks adapt to novel stimuli and changing environments in flexible timing tasks. By reverse-engineering trained RNNs, they show that combining low-dimensional activity and tonic control signals enables generalization and fast adaptation. They further identify signatures of this mechanism in the frontal cortex of monkeys.

Beiran et al., 2023, Neuron *111*, 739–753
March 1, 2023
https://doi.org/10.1016/j.neuron.2022.12.016

CellPress

Neuron

CellPress

Article

Parametric control of flexible timing through low-dimensional neural manifolds

Manuel Beiran,[1,2,8] Nicolas Meirhaeghe,[3,4,8] Hansem Sohn,[5,7] Mehrdad Jazayeri,[5,6,*] and Srdjan Ostojic[1,9,*]

[1]Laboratoire de Neurosciences Cognitives et Computationnelles, INSERM U960, Ecole Normale Superieure - PSL University, 75005 Paris, France

[2]Zuckerman Mind Brain Behavior Institute, Columbia University, New York, NY 10027, USA

[3]Harvard-MIT Division of Health Sciences and Technology, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

[4]Institut de Neurosciences de la Timone (INT), UMR 7289, CNRS, Aix-Marseille Université, Marseille 13005, France

[5]McGovern Institute for Brain Research, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

[6]Department of Brain & Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

[7]Present address: Center for Neuroscience Imaging Research, Institute for Basic Science, Department of Biomedical Engineering, Sungkyunkwan University, Suwon, Republic of Korea

[8]These authors contributed equally

[9]Lead contact

*Correspondence: mjaz@mit.edu (M.J.), srdjan.ostojic@ens.psl.eu (S.O.)

https://doi.org/10.1016/j.neuron.2022.12.016

SUMMARY

Biological brains possess an unparalleled ability to adapt behavioral responses to changing stimuli and environments. How neural processes enable this capacity is a fundamental open question. Previous works have identified two candidate mechanisms: a low-dimensional organization of neural activity and a modulation by contextual inputs. We hypothesized that combining the two might facilitate generalization and adaptation in complex tasks. We tested this hypothesis in 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, enabling generalization to novel inputs and adaptation to changing conditions. Reverse-engineering and theoretical analyses demonstrated that this parametric control relies on a mechanism where tonic inputs modulate the dynamics along non-linear manifolds while preserving their geometry. Comparisons with data from behaving monkeys confirmed the behavioral and neural signatures of this mechanism.

INTRODUCTION

Humans and animals can readily adapt and generalize their behavioral responses to new environmental conditions.[1–5] This capacity has been particularly challenging to realize in artificial systems,[6–8] raising the question of what biological mechanism might enable it. Two mechanistic components have been put forward. A first proposed component is a low-dimensional organization of neural activity, a ubiquitous experimental observation across a large variety of behavioral paradigms.[9–12] Indeed, theoretical analyses have argued that, while high dimensionality promotes stimulus discrimination, low dimensionality instead facilitates generalization to previously unseen stimuli and conditions.[13–20] A second suggested component is tonic inputs that can help cortical networks modulate their outputs to stimuli that appear in distinct behavioral context[19,21–28] and thereby generalize their responses to new circumstances. These two mechanisms have so far been investigated separately in different sets of simplified tasks, but they are not mutually exclusive and could in principle have complementary functions. Whether and how an interplay between low-dimensional dynamics and tonic inputs might enable generalization and adaptation in more complex cognitive tasks remains an open question.

Here, we address this question within the framework of timing tasks that demand flexible control of the temporal dynamics of neural activity.[24,25,29–40] Previous studies have demonstrated that a low-dimensional organization is a prominent feature of neural activity recorded during such tasks.[24,33,34,36,37] Several of those studies, moreover, found that tonic inputs that provide contextual information allow networks to flexibly adjust their output to identical stimuli.[34,36] However, so far, the computational roles of low-dimensional dynamics and contextual inputs were only probed within the range on which the network was trained. Going beyond previous studies, we instead hypothesized that combining the two mechanisms by pairing contextual input signals with low-dimensional dynamics might facilitate generalization to novel inputs and adaptation to changing environments.

We investigate this hypothesis using a multidisciplinary approach spanning network modeling, theory, and analyses of neural and behavioral data in non-human primates. 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.[33,34,36] 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.[41–44] We then compared the performance, generalization capabilities, and neural dynamics of these low-rank RNNs with networks trained in the absence of any constraint on the connectivity and the dimensionality of dynamics. We similarly contrasted networks with and without tonic inputs.

We first show that low-rank RNNs, unlike unconstrained networks, are able to generalize to novel stimuli well beyond their training range but only when paired with tonic inputs that provide contextual information. Specifically, we found that, in low-rank networks, tonic inputs parametrically control the input-output transform and its extrapolation. Reverse-engineering the trained networks revealed that tonic inputs endowed the system with generalization by modulating the dynamics along low-dimensional non-linear manifolds while leaving their geometry invariant. Population analyses of neural data recorded in behaving monkeys performing a time-interval reproduction task[36,40] confirmed the key signatures of this geometry-preserving mechanism. In a second step, we show how the identified mechanism allows networks to quickly adapt their outputs to changing conditions by adjusting the tonic input based on the temporal statistics. Direct confrontation of newly collected behavioral and neural data with low-rank networks supported the idea that contextual inputs are key to adaptation. All together, our work brings forth evidence for the computational benefits of controlling low-dimensional activity through tonic inputs and provides a mechanistic framework for understanding the role of resulting neural manifolds in flexible timing tasks.

RESULTS

Generalization in flexible timing tasks

We trained 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 (Figures 1A and 1B). In all tasks, the network output was a weighted sum of the activity of its units (see STAR 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.[33,34,36] 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 (Figure 1B). Our goal was to study which properties of RNNs favor generalization, i.e., extend their outputs to inputs not seen during training. We therefore examined the full input-output transform performed by the trained networks (Figure 1C), and we 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[44] (Figure 1D).

We first examined the role of dimensionality for generalization when a tonic input is present. We considered the Cue-Set-Go (CSG) task,[33] where the goal is to produce an output time interval associated with a tonic input cue, the level of which varies from trial to trial (Figure 1B, top). Correctly performing this task required learning the association between each cue and target output interval. We found that networks with rank-two connectivity solved this task with similar performance to unconstrained networks (Figures 2A and S1) but led to lower-dimensional dynamics (Figures 2A and 2B, top; see STAR Methods). During training, we used four different input amplitudes that were linearly related to their corresponding output intervals (four different color lines in Figure 2A). 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 (Figure 2C, top). However, only low-rank networks showed extrapolation (i.e., a smooth monotonic input-output mapping) when networks were probed with cue amplitudes beyond the training range. Specifically, as the amplitude of the cue input was increased, low-rank networks generated increasing output time intervals (Figure 2C, see Figure S2 for additional examples of similarly trained networks), while unconstrained networks instead stopped producing a ramping signal altogether (see Figure S1).

We next asked whether low-dimensional dynamics enable generalization in absence of tonic inputs. We therefore 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 (Figure 1B, bottom). The network was required to measure that interval, keep it in memory during a delay period of random duration, and then reproduce it after the Set signal.

In this task, because 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. We found that a minimal rank $R = 3$ was required for low-rank networks to match the performance of unconstrained networks (Figures 2A, 2B, bottom, and 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 low-rank nor the unconstrained networks were able to extrapolate to intervals longer or shorter than those in the training set (Figure 2C, bottom; see Figure S1 for the input-output curves for different delays and Figure S2 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.

Altogether, these results indicate that generalization does not emerge from low dimensionality alone (MWG task) but depends on the presence of tonic inputs (CSG task).

Combining contextual inputs in low-rank networks enables parametric control of extrapolation

The key difference between the CSG and MWG tasks is 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. Because 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 low-rank networks.

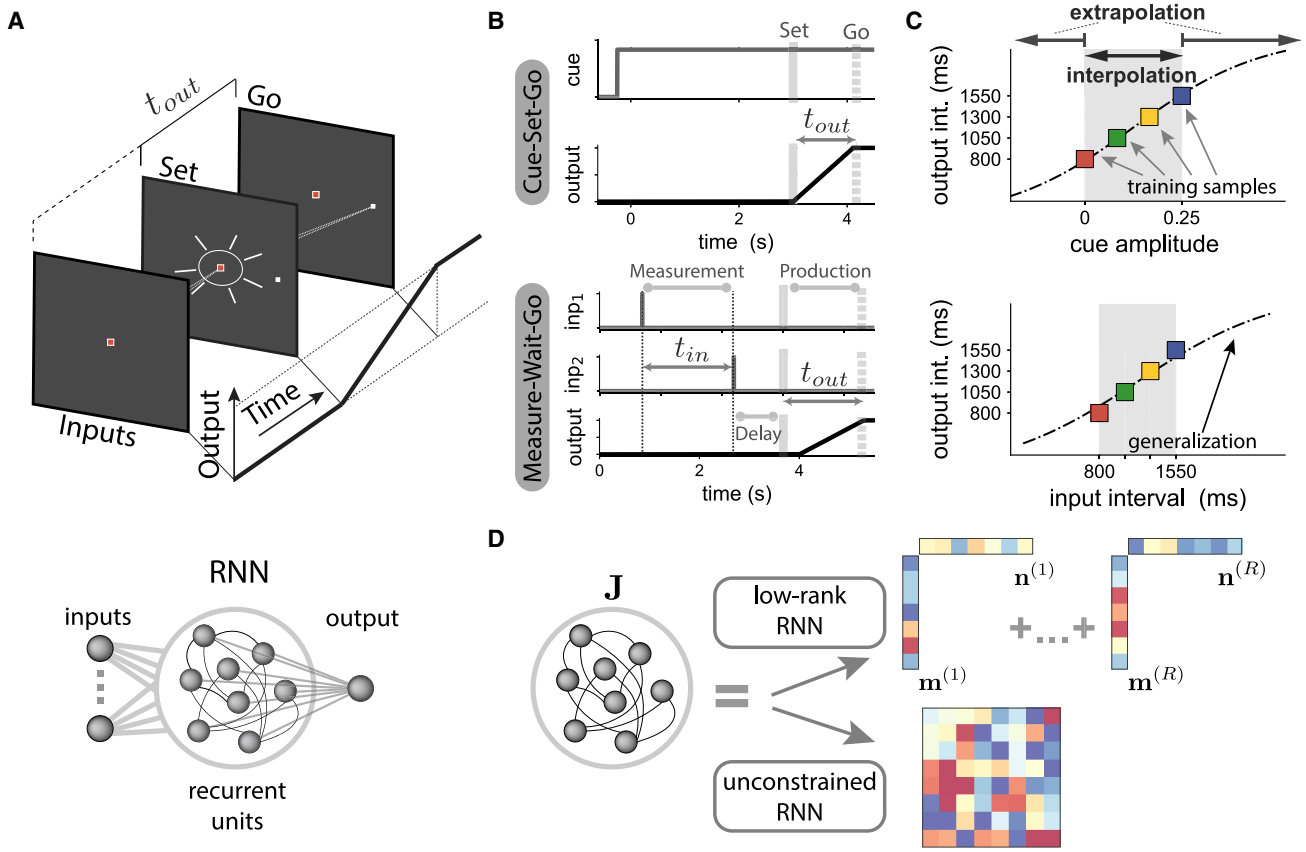

Figure 1. Investigating generalization in flexible timing tasks using low-rank recurrent neural networks (RNNs)

(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 CSG (top), the amplitude of a tonic input cue (gray line), present during the whole trial duration, indicates the time interval to be produced. In MWG (bottom), two input pulses are fed to the network ("inp 1" and "inp 2"), followed by a random delay. The input time interval t_{in} 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) such that a specific cue amplitude (top, CSG task) or input interval t_{in} (bottom, MWG task) 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, gray shaded region) and beyond (extrapolation) the training range, as illustrated with the dotted line.

(D) To examine the influence of dimensionality, we compared two types of trained RNNs. Low-rank networks had a connectivity matrix $\boldsymbol{J}$ of fixed rank R, which directly constrained the dimensionality of the dynamics.[44] Such networks were generated by parametrizing the connectivity matrix as in Equation 2 and training only the entries of the connectivity vectors $\boldsymbol{m}^{(r)}$ and $\boldsymbol{n}^{(r)}$ for $r = 1 \ldots R$. Unconstrained networks instead had full-rank connectivity matrices, where all the matrix entries were trained.

To test 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 (Figure 3A). A low-amplitude tonic input indicated a relatively shorter set of intervals (800–1,550 ms) and a higher tonic input, a set of longer intervals (1,600–3,100 ms). Note that the functional role of this tonic 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 MWG task, both rank-three and unconstrained networks were able to solve the MWG + Ctxt task and reproduced input intervals sampled from both distributions (Figure S3). 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 generalization to interval ranges outside those the networks were trained for. To test this hypothesis, we examined generalization to both unseen input

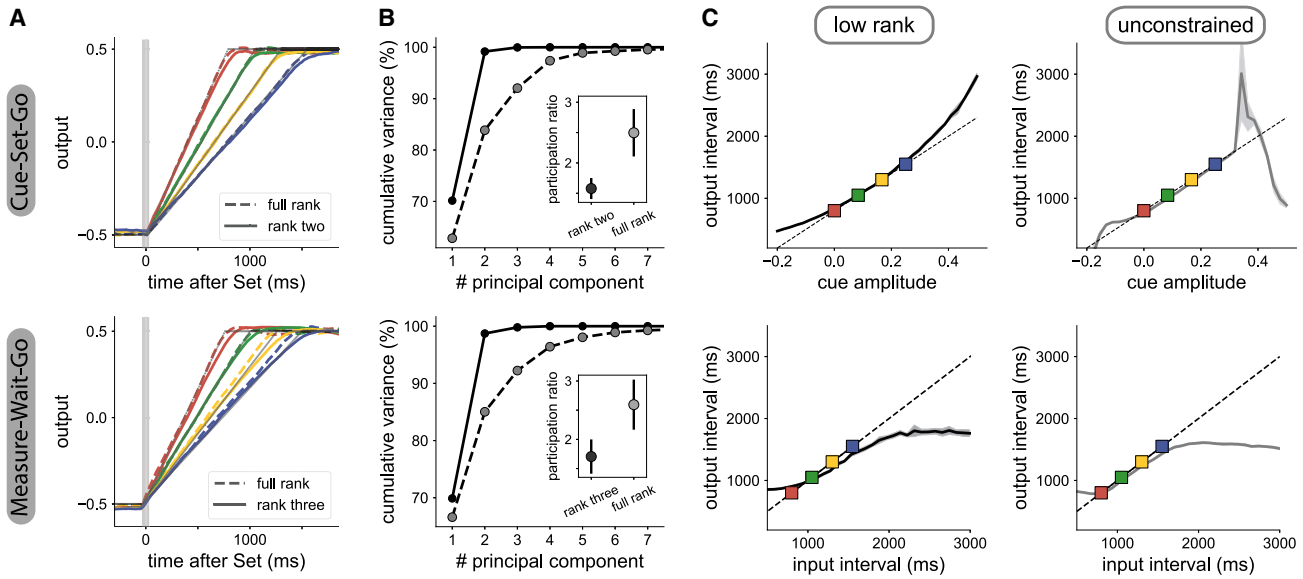

Figure 2. Influence of the dimensionality of network dynamics on generalization

Top: CSG task; bottom: MWG task.

(A) 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. Note that the dashed colored lines overlap with the solid ones.

(B) Dimensionality of neural dynamics following the Set signal. Cumulative percentage of explained variance during the measurement epoch is shown 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 during the whole trial duration (see STAR Methods), with mean and SD over 5 trained RNNs.

(C) Generalization of the trained RNNs to novel inputs in 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. The longest output interval that each network can generate is close to the longest output interval learned during training. For visualization, the delay is fixed to 1500 ms. See Figure S1 for additional delays and trained networks.

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 (Figure 3B, left) and was set by the longest interval of the corresponding interval distribution, while in unconstrained networks it did not depend strongly on context (Figure 3B, right). More generally, in low-rank networks, the contextual inputs modulated the input-output function and biased output intervals toward 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 (Figure 3B, left), reminiscent of what has been observed in human and monkey behavior in similar tasks.[36,45,46] 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 (Figure 3B, right).

We next probed generalization to values of the contextual input that were never presented during training. Strikingly, in low-rank networks, we found that novel contextual inputs parametrically controlled the input-output transform, therefore generalizing across context values (Figure 3C). 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 midpoint to longer values (Figure 3C, bottom left), up to a maximal value of the contextual cue that depended on the training instance (Figure S3). Conversely, contextual cues smaller than used in training instead reduced the output range and shifted its mean to smaller intervals (Figure 3C, bottom right). In contrast, in networks with unconstrained rank, 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 (Figure S3).

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

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

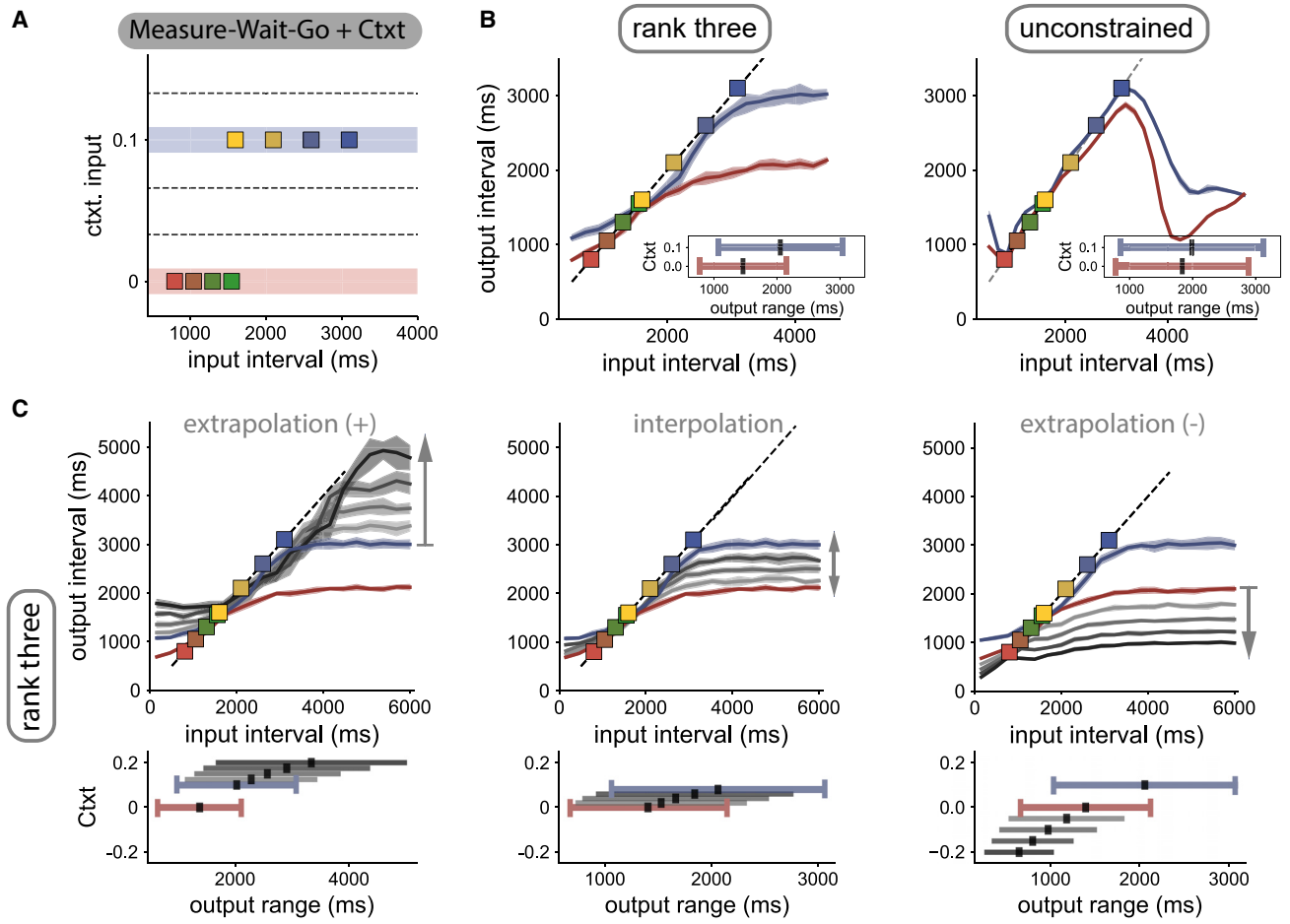

Figure 3. Control and extrapolation in the Measure-Wait-Go task with contextual cues

(A) Task design. Input time intervals t_{in} 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 novel contextual cues (dashed lines).

(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.

(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 gray lines, respectively). See Figure S2 for additional RNNs.

underlying low-dimensional dynamics.[44,47,48] In low-rank networks, the connectivity constrains the activity to be embedded in a low-dimensional subspace,[41,43,44] allowing for a direct visualization. Examining the resulting low-dimensional dynamics for networks trained on flexible timing tasks, in this section we show that the neural trajectories are attracted to non-linear manifolds along which they slowly evolve. 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. Additional details are provided in the STAR Methods.

Low-dimensional embedding of neural activity

The RNNs used to implement flexible timing tasks consisted of N units, with the dynamics for the total input current x_i to the i-th unit given by:

$$\tau \frac{dx_i}{dt} = -x_i + \sum_{j=1}^{N} J_{ij}\varphi(x_j) + \sum_{s=1}^{N_{in}} I_i^{(s)} u_s(t) + \eta_i(t). \quad \text{(Equation 1)}$$

Here $\varphi(x) = \tanh(x)$ is the single neuron transfer function, J_{ij} is the strength of the recurrent connection from unit j to unit i, and $\eta_i(t)$ is a single-unit noise source. The networks received N_{in} task-dependent scalar inputs $u_s(t)$ 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 $u_s(t)$ 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

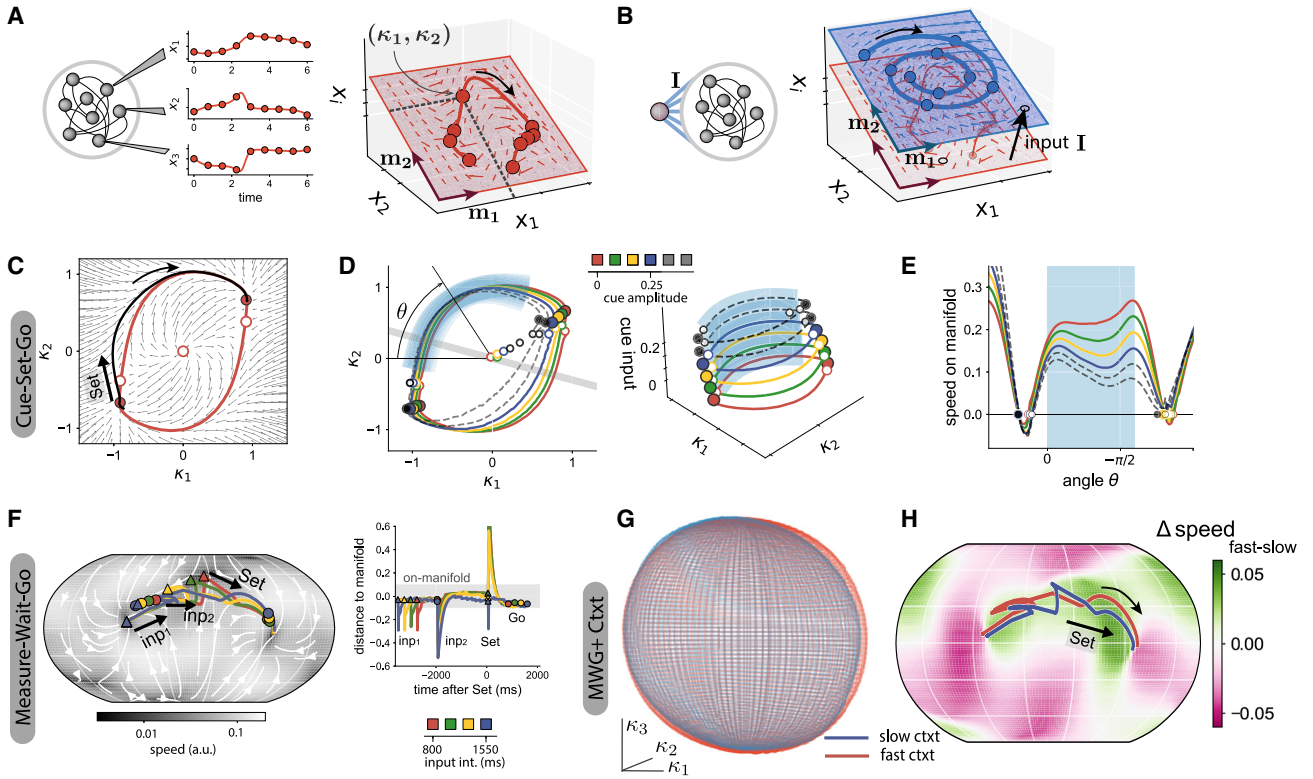

Figure 4. Reverse-engineering RNNs trained on flexible timing tasks

(A) The temporal activity of different units in the network (left) can be represented as a trajectory in state space (right, thick red line). In low-rank networks, trajectories are embedded in a lower dimensional space (red shaded plane, see Equation 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 (arrows, see Equation 4).

(B) External inputs increase the dimensionality of the embedding space. Here, a single input is added; the embedding subspace is now three-dimensional: two dimensions for the recurrent subspace and one dimension for the input $\boldsymbol{I}$. A tonic input (black arrow) shifts the recurrent subspace ($\boldsymbol{m}^{(1)}$–$\boldsymbol{m}^{(2)}$) along the input direction and modifies the flow field.

(C–E) RNN trained on the CSG task. Stable fixed points: color-filled dots; unstable fixed points, white dots.

(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 (Figure S4). The trial starts at the bottom left stable fixed point. The Set pulse initiates a trajectory toward the opposite stable fixed point, which quickly converges to the slow manifold (see Figure S5A) and evolves along it.

(D) Manifolds generated by a trained RNN for different amplitudes of the cue (colored lines: cues used for training, black dashed lines: cues beyond the training range). Left: 2D projections of the manifolds onto the recurrent subspace. The gray line is the projection of the readout vector on the recurrent subspace. Right: 3D visualization of the manifolds in the full embedding subspace, spanned by $\boldsymbol{m}^{(1)}$, $\boldsymbol{m}^{(2)}$, and $\boldsymbol{I}_{\boldsymbol{cue}}$. 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.

(E) Speed along the manifold as a function of the polar angle θ. The speed along the manifold is scaled by the cue amplitude, even beyond the training range. Shaded blue region is 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 (Figure S4). 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, $u_{ctxt} = 0.1$) are shown.

(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 also Figure S5).

space, where each axis corresponds to the total current received by one unit (Figure 4A). In low-rank networks, the connectivity constrains the trajectories to reside in a low-dimensional linear subspace of the state space[41,43,44] that we refer to as the *embedding space*.[12] 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

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

The connectivity vectors, as well as the feedforward 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 feedforward input vectors $\boldsymbol{I}^{(s)}$ (Figure 4A). 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 $\boldsymbol{I}^{(s)}$:[41,43,44]

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

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.[33] 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 $\boldsymbol{v}$:[43,44]

$$\tau\frac{d\boldsymbol{\kappa}}{dt}(t) = \boldsymbol{F}(\boldsymbol{\kappa}, \boldsymbol{v}). \qquad \text{(Equation 4)}$$

In Equation 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 (Figure 4A). Pulse-like inputs instantaneously shift the position of activity in the embedding space such that the transient dynamics in between pulses are determined by this dynamical landscape. Instead, tonic cue inputs, such as the signals used in the CSG and MWG + Ctxt tasks, shift the recurrent subspace within the embedding space and thereby modify the full dynamical landscape (Figure 4B). 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 neural trajectories 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*,[12] and we devised methods to identify them from network dynamics (STAR Methods and Figure S4). 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 (Figure 4C). 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 (Figure 4C, red line). This manifold connected two stable fixed points through two intermediate saddle points (Figure 4C, red and white dots, respectively). At trial onset, the neural activity started at one 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 (Figure 4C, black line) that was quickly attracted to the ring-like manifold (Figures S4 and S5) and subsequently followed slow dynamics along this manifold toward 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 (Figure 4D). 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 visualized in the three-dimensional embedding space defined by κ_1, κ_2, and the input cue as a third coordinate (Figure 4D, 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 (Figure 4E) and, thereby, the slope of the ramping output that determined the output interval (Figure 2A). Because of the cylindrical geometry, extending cue amplitudes to values outside of the training range preserved the overall structure of the manifold (Figure 4D, black dashed lines) 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, only diverging from its surface during the fast transient responses to the input pulses (Figures 4F and S4).

The neural dynamics 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 (Figure S5). 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 toward the other side of the sphere at speeds set by the initial condition on the line attractor, therefore leading to ramp signals with varying slopes (Figure S1). Crucially, the line attractor 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. Thus, the finite limits of the line attractor lead to a sigmoidal input-output function where the output intervals were saturated to the bounds of the training range, as seen in Figure 2C.

For the extended MWG + Ctxt task, 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 (Figure 4G) while scaling the speed of the dynamics on it (Figure 4H). 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 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.

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 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,[41–44,49] 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[41,43,44] where for each neuron i, the set of components $\{m_i^{(r)}, n_i^{(r)}\}_{r=1\ldots R}$ and $\{I_i^{(s)}\}_{s=1\ldots N_{in}}$ along connectivity and input vectors are drawn randomly and independently from a zero-mean multivariate Gaussian distribution (Figure S6A). 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 (STAR Methods). We therefore investigated how this overlap structure determines the dynamics that underlie computations.

Generating slow manifolds through recurrent connectivity

We first focused solely on 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[41,43,50] and detailed in the STAR 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 (Figure S6B). More precisely, for a rank R network, such a symmetry in the overlap matrix induces an attractive R-dimensional spherical manifold in neural space where each point on the manifold is an attractive fixed point (see STAR Methods). Accordingly, for rank-two networks, this structure leads to a continuous ring attractor embedded in a plane, and for rank-three networks it leads 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 such that only a small number of points on it remain exact fixed points of the dynamics (Figure S6C). 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 (Figure S6C, middle), as seen in trained low-rank networks (Figures 4C and 4D). 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 in the overlap matrix rotates the position of saddle points and brings them closer to stable fixed points (Figure S6D). 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 (Figure 4C).

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. In particular, we studied what properties of the input vectors and recurrent connectivity produce a change in dynamics such as the one observed in the CSG task (Figure 4D) landscape. 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 (Figures 5A and 5B, top row). In particular, increasing the amplitude of non-specific inputs shrinks the radius of the activity manifold until it eventually collapses. In contrast, for subspace-specific inputs, varying the input amplitude modulates the speed of the dynamics (similar to non-specific inputs, Figure 5B) but, importantly, keeps approximately invariant the geometry of the manifold and the position of fixed points on it (Figure 5A, 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 CSG task and MWG task with context. We perturbed the trained input vectors in two ways; we either

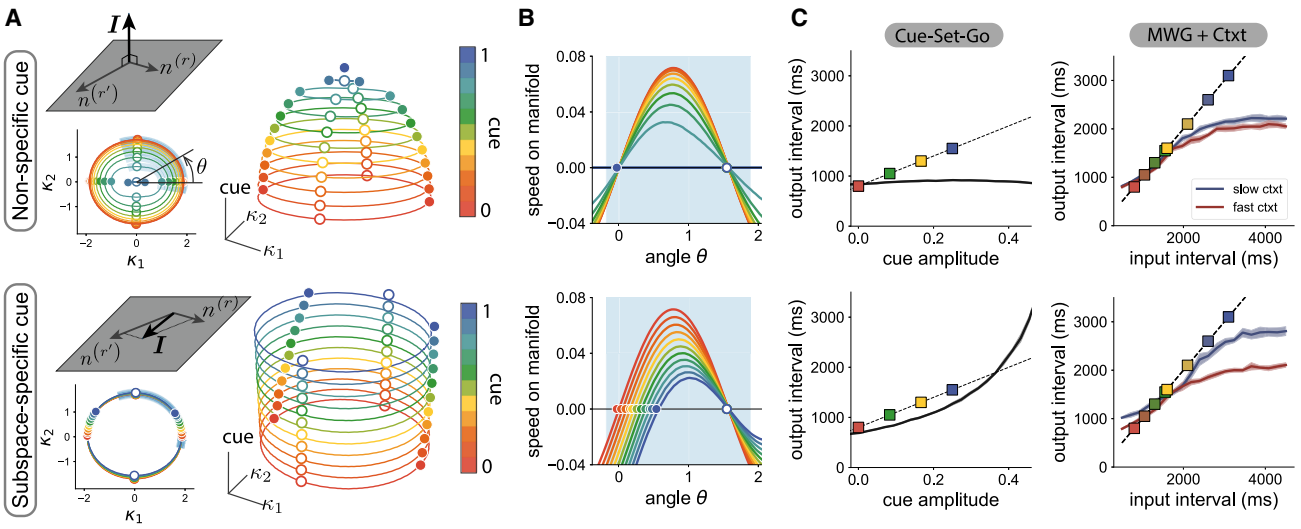

Figure 5. Effect of tonic cue inputs on the geometry of, and dynamics on, manifolds

Top: non-specific cue orthogonal to the recurrent connectivity vectors. Bottom: specific cue input within the subspace of recurrent connectivity vectors.

(A) Projection of manifolds on the recurrent subspace (bottom left inset) and on the 3D embedding subspace (right). Different colors correspond to different cue amplitudes. Solid lines indicate the manifold location (colored dots: stable fixed points, white dots: 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.

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 (Figure 5C, top). In contrast, removing all non-specific components from the input vectors in the trained networks did not strongly affect the performance of the network (Figure 5C, 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 CSG and MWG tasks based on this mechanism by setting connectivity parameters without training (Figure S7). 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.

Signatures of the computational mechanism in neural data

Our analyses of network models point to a putative mechanism for rapid generalization in timing tasks. Specifically, generalization emerges when inputs are paired with a low-dimensional contextual signal that is parametrically adjusted to the range of possible inputs. To test for the presence of this computational strategy in the brain, we turned to empirical data recorded during flexible timing behavior. We analyzed neural population activity in the dorsomedial frontal cortex (DMFC) of monkeys performing the "Ready-Set-Go" (RSG) task, a time interval reproduction task analogous to the MWG + Ctxt task but without a delay period.[36,40] In this task, animals had to measure and immediately reproduce different time intervals. The feature that made this task comparable to the MWG + Ctxt task was that the time interval on each trial was sampled from one of two distributions (i.e., a "fast" and a "slow" context), and the color of the fixation spot throughout the trial explicitly cued the relevant distribution, thereby providing a tonic contextual cue.

Previous studies have reported on several features of neural dynamics in this task.[36,40] In particular, it was found that during the measure of the intervals, population activity associated with each context evolves along two separate trajectories running at different speeds. While this observation is qualitatively consistent with the contextual speed modulations seen in our network models, it is by itself insufficient to validate the hypothesized mechanism, a tonic contextual input controlling the low-dimensional dynamics. To firmly establish the connection between our model and the neural data, we thus devised a new set of quantitative analyses aimed at directly comparing their dynamics.[24]

To guide our comparison between the model and the data, we focused on three main signatures of the low-rank networks. Our first observation was that the neural trajectories associated with the two contexts remain separated along a relatively constant dimension throughout the measurement epoch of the task. Indeed, as we have shown above, the contextual input translates the manifold of activity along a "context dimension" that is orthogonal to the subspace in which the trajectories evolve (Figure 6A, top left). This translation was also visually manifest in the data: the empirical trajectories associated with the two

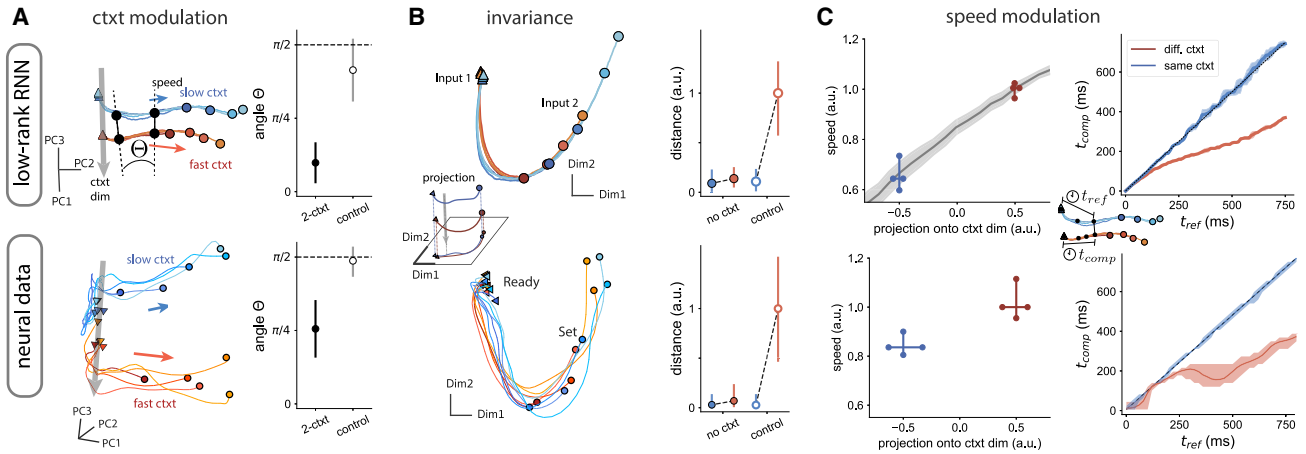

Figure 6. Signatures of the computational mechanism in neural data

Comparison between a low-rank network trained on the MWG + Ctxt task (top) and neural data in a time-reproduction task[36,40] (bottom).

(A) Left: projections on the first principal components of the activity during the measurement epoch. Different trajectories correspond to different input intervals; colors denote the two contexts. Right: mean (in absolute value) and standard deviation of the angle between vectors separating trajectories. Control corresponds to the same analysis with shuffled context categories.

(B) Left: 2D projection of activity orthogonal to the context axis. Trajectories evolve along a non-linear manifold that is invariant across contexts. Right: average distance between trajectories during the measurement epoch after removing the projection onto context dimension. The distance is computed between each average context trajectory (no ctxt; red: fast ctxt; blue: slow ctxt) and the reference trajectory (the slow ctxt). The estimate of the distance for the reference trajectory (blue) with itself uses different subsets of trials to quantify the noise. The control consists of removing the projection of the activity on a random direction (see STAR Methods).

(C) Left: average speed of trajectories during the measurement epoch as a function of the projection onto the context dimension. In the RNN, the grey line shows the same analysis for context amplitudes not seen during training. There is a monotonic relation between the speed of neural activity during measurement and the projection of the activity onto the context dimension. Right: speed estimation shown as the time elapsed until reaching a reference state (given by the average trajectory in the slow context). Trajectories from the context different from reference evolve faster after an initial transient at equal speeds. Error bars indicate 99% confidence interval.

contexts appeared to evolve in parallel, as shown by a 3D projection of the dynamics (Figure 6A, bottom left). To rigorously assess the degree of parallelism of the trajectories in higher dimensions, we computed the angle between the vectors separating the two trajectories at different time points (STAR Methods). We expected this angle to be small if the dimension separating the trajectories remained stable over time. Consistent with our initial observations, we indeed found that in the low-rank networks this angle remained close to zero (Figure 6A, top right). When we performed the same analysis on the neural data, we found a similar degree of parallelism between the trajectories (Figure 6A, bottom right): in both the model and the data, the angle remained far from 90° for at least 500 ms (Figure S8A), indicating that the context dimension was largely stable over time.

Our second observation was that the neural trajectories in the low-rank networks were invariant across contexts and overlapped in the recurrent subspace (Figure 6B, left). We visually found a similar invariance in the neural data after projecting out the context dimension (Figure 6B, bottom left). To quantify this effect, we calculated the distance between trajectories once the projection of the activity along the previously identified context dimension was removed (see STAR Methods). For comparison, we also calculated the distance between trajectories when removing the projection on a random dimension of the state space (see Figure S8B for the time-resolved distances). We found that, both in the networks and the data, the distance was largely reduced to zero once the context dimension was removed (Figure 6B, right), thus demonstrating a similar degree of invariance in the dynamics.

A final key signature of the low-rank networks was that the speed of dynamics during measurement depended on context: faster in the fast context, slower in the slow context. We measured speed by calculating the amount of change in neural activity in small periods of time. In parallel, we projected the activity onto the context dimension estimated at the beginning of measurement. We found that there was a monotonic relation between the activity along the context dimension and the speed of trajectories, based on the slow and fast contexts (blue and red crosses, Figure 6C, right). In addition, in the low-rank networks, we were able to establish the full functional relation between speed and projection on context for context amplitudes not seen during training (gray line, Figure 6C, right).

Direct neural speed estimates are, however, a measure sensitive to the noise level of the activity. To better compare neural speed in the network and the recordings, we used "kinematic analysis of neural trajectories" (KiNeT), which provides a finer-grain analysis based on relative time elapsed to reach a certain state (see STAR Methods[24]). Using this technique first on the networks we confirmed that there are strong speed modulations between different contexts. Furthermore, speed modulations only appeared once the transient signal from the beginning of measurement faded out, roughly 200 ms following Ready (Figure 6C, top). Applying the same technique to the neural data, we observed that the empirical trajectories also diverged in

terms of speed only after about 200 ms following the Ready signal (Figure 6C, bottom).

Altogether, these quantitative analyses show that neural activity exhibited the key signatures of parametric control by a low-dimensional input predicted from computational modeling and theory, suggesting that the dynamics of neural trajectories in the cortex may be optimized to facilitate generalization.

Adaptation to changing input statistics

Our comparison of the network model with neural data argues that flexible timing across contexts is controlled by a tonic input that flexibly modulates the network's dynamics. So far, we focused on the situation where the two contexts were explicitly cued and well-known to the animals, and we assumed that the corresponding level of the control input was instantaneously set by an unspecified mechanism. A key question is, however, how the value of this control input can be adaptively adjusted following uncued changes in input data. In this section, we propose a simple mechanism to dynamically update the contextual input based on the input interval statistics and compare the predictions of this augmented model with behavioral and neural data from a new monkey experiment.

To allow the model to infer changes in context from the recent history of measured intervals, we hypothesize that the network continuously represents and updates an estimate of the mean of the interval distribution. Indeed, previous work has shown that adjusting neural dynamics based on the mean interval may provide the substrate for adapting to new interval statistics.[40] Inspired by this result, we assume that the amplitude of the contextual input linearly encodes the mean input interval and is adjusted on every trial based on a "prediction error," i.e., the difference between the measured interval and the current estimate of the mean (Figure 7A). The adjustment of the contextual input can thus be formulated as a simple auto-regressive process that continuously tracks the statistics of the intervals (see STAR Methods).

A direct consequence of this mechanism is that the network becomes sensitive to local variations in the statistics of the intervals within a given context and adapts its output accordingly. In particular, the network's output on a given trial is biased toward the input interval of the previous trial (Figure 7B). Strikingly, we found a similar tendency in monkeys trained on a single context (Figure 7B, right): animals respond generally faster following a short interval and slower following a long interval. Note that this history effect is abolished in the networks when the updating of the control input is turned off (Figure 7B, middle), indicating that this effect specifically emerges from contextual input adjustments.

These behavioral results suggest that animals may rely on an updating mechanism similar to that implemented in the networks to remain tuned to changing stimulus statistics. To test the predictions of the model directly at the level of neural dynamics, we developed a novel experiment that combined context-based and adaptation-based timing. Each session was divided in two parts. In the first part, animals reproduced measured intervals under the same conditions as previously, i.e., they were exposed to two alternating interval distributions explicitly cued via the color of the fixation spot. In the second part of the experiment, following an uncued switch, the distribution corresponding to one of the two contexts was covertly changed to an intermediate distribution, while the color of the fixation point remained unchanged (Figure 7C, left). The specific predictions of our model were that: (i) the level of the pre-stimulus contextual input adapts to an intermediate value along the identified contextual axis; (ii) the speed of the dynamics following the Ready pulse adjusts according to the level of the contextual input.

We analyzed DMFC population activity in this task and compared it to the predictions of our low-rank networks, similar to Figure 6C, left. As expected, in the RNNs, after the switch, the network learned the new value of the contextual input, and this value was associated with an intermediate value of speed (Figure 7C, gray dots). The same key findings were observed in the neural data: we projected population activity post-switch along the context dimension defined pre-switch and found that this projection, as well as the speed of dynamics, was adjusted to intermediate values appropriate for the new distribution (Figure 7C, right). This result is particularly notable, since it demonstrates a non-trivial match between neural data and our low-rank networks constrained to adapt via changes in a tonic input that encodes context.

Overall, these analyses provide compelling evidence that a low-dimensional input control strategy provides an efficient way to promote both generalization and adaptability for flexible timing, and it is likely at play in the brain.

DISCUSSION

Examining recurrent neural networks trained on a set of flexible timing tasks, we show that controlling low-dimensional dynamics with tonic inputs enables a smooth extrapolation to inputs and outputs well beyond the training range. Reverse-engineering and theoretical analyses of the recurrent networks demonstrated that the underlying mechanism for generalization relied on a specific geometry of collective dynamics. Within a given condition, collective dynamics evolved along non-linear manifolds, while across conditions, tonic cues modulated the manifolds along an orthogonal direction. This modulation parametrically controlled the speed of dynamics on the manifolds while leaving their geometry largely invariant. We demonstrated that this mechanism leads to fast adapting responses in changing environments by adjusting the amplitude of the tonic input while reusing the same recurrent local network. Population analyses of neural activity recorded while monkeys adaptively solved time-interval reproduction tasks confirmed the key geometric and dynamic signatures of this mechanism.

At the algorithmic level, the RNNs that generalized to novel stimuli suggested a computational principle for dynamical tasks that makes a clear separation between recurrent dynamics and two types of inputs based on their function and timescale. First, the scaffold for time-varying dynamics is crafted by recurrent interactions that generate manifolds in neural state-space. Second, on the level of individual trials, fast inputs place neural trajectories at the suitable locations in state-space that allow time-varying responses to unfold. Third, tonic inputs, which are constant during the trial duration, provide the parametric control of the dynamics on the manifold by shifting trajectories to different regions of state-space. These tonic inputs can, however, vary at the timescale of trials

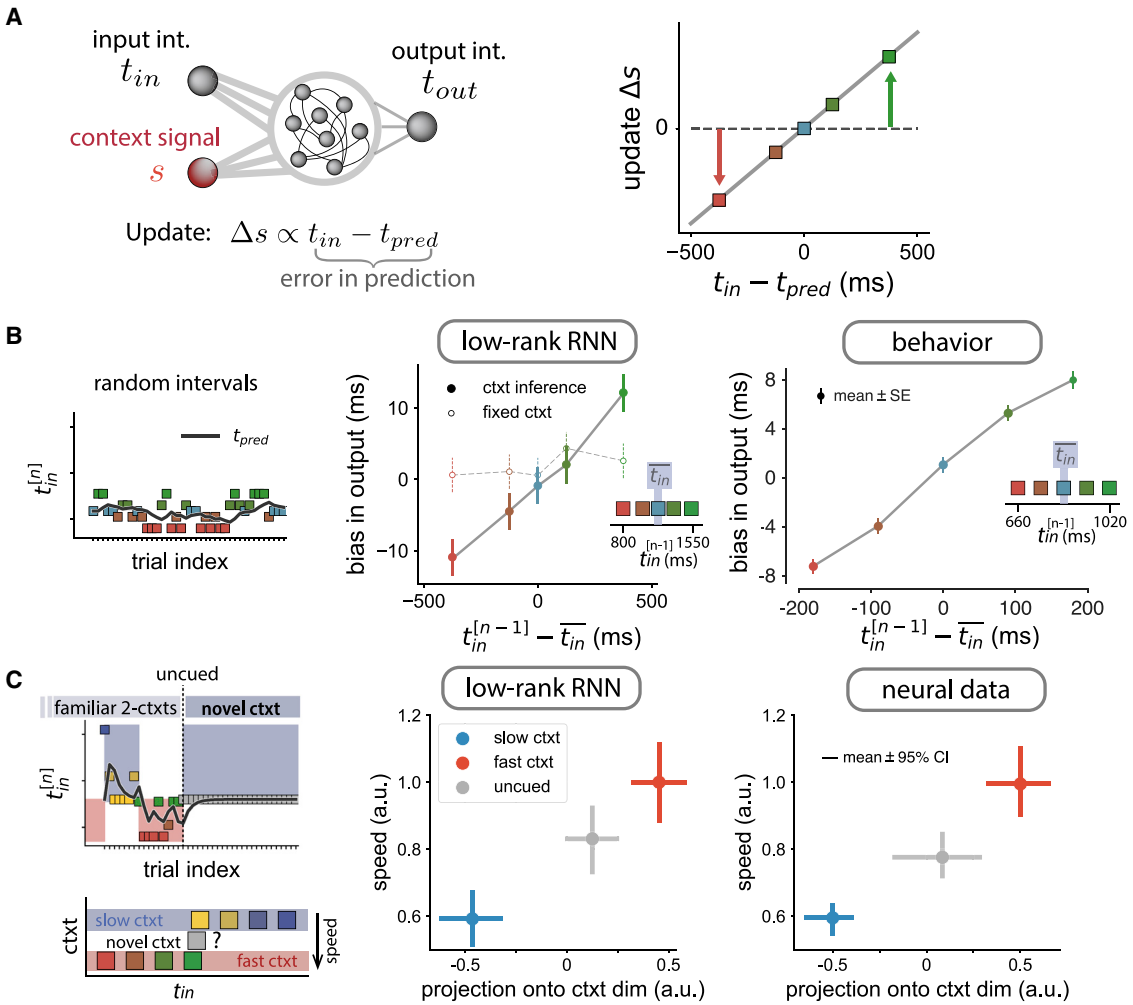

Figure 7. Adaptation to a change in statistics of input intervals

(A) Extended model, in which the context signal amplitude is updated in each trial proportionally to the difference between the input interval and the predicted input interval t_{pred} (see STAR Methods).

(B) Effect of contextual input adjustments in a single-context task. Left: task structure of a session: randomly interleaved trials from a single distribution. Black line indicates the predicted input interval at every trial. Center: bias in the output interval of a trained RNN as a function of the interval presented in the previous trial (colored dots). The control data correspond to the same trained RNN with constant context signal. Straight lines connecting the dots were added for visualization. Right: behavioral bias in monkeys performing a timing task. Insets show the distribution of input intervals, highlighting its mean value, t_{in}.

(C) Effect of contextual input adjustments in a two-context adaptation task. Left: task structure of a session: trials are structured into alternating blocks. In each block, trials are randomly drawn from different distributions (slow and fast contexts, bottom). At some point, there is an uncued transition to a novel distribution, consisting of a single input interval, ranging between the means of the slow and fast contexts. Center: speed of neural activity in the RNN vs. the projection of neural activity onto the estimated context dimension during the measurement epoch (see STAR Methods). Right: same analysis in the neural data. In the novel uncued context (grey dot), the speed and projection onto context are adjusted to values between the slow and fast contexts.

and encode the contextual information necessary to adapt to the changing statistics of the environment.[40] We expect that this principle of a separation of inputs along different behaviorally relevant timescales extends to other parametric forms of control,[6,51,52] and provides a fundamental building block for neural networks that implement more complex internal models of the external world.

A key objective of computational modeling is to make falsifiable predictions in experimental recordings; our work on RNNs supplies predictions for both generalization and adaptation. We performed experiments on primates that directly test some of the predictions on adaptation. We found a behavioral bias toward previous trials, compatible in our model with a predictive error signal required for adaptation. At the level of neural recordings, in line with the modeled tonic input controlling neural speed, we found that neural trajectories in dorsomedial frontal cortex adaptively varied along a contextual axis that determined the speed of neural dynamics. On the other hand, we did not have the data for validating our predictions on generalization (i.e., zero-shot learning). Nevertheless, based on the adaptation results, we expect that when animals generalize, their neural activity adjusts the tonic input within a single trial. This change in the input to

Table 1. Parameters for trained RNNs

Number of units N	1,000
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

the local network could be provided either through an external input present during the whole trial duration (e.g., an explicit instruction, or context cue as in Sohn et al.[36]) or some transient input that is mapped onto a tonic input in the brain (e.g., in reversal learning paradigms, where the contextual input reflects the inference of sudden switches in the environment[53,54]).

We have shown that low-dimensional recurrent dynamics can be flexibly controlled by a tonic input that varies along one dimension. It remains an open question how low-dimensional the dynamics need to be in order to be controllable with an external tonic input. In our case, we enforced the minimal dimensionality required to solve the task (up to three dimensions for the MWG task) and showed that networks generalize when provided with a tonic input. In the other extreme, trained networks without any dimensionality constraint did not generalize with a tonic input. These results suggest that flexible cognitive tasks that require more time to be learned are either high-dimensional and require additional mechanisms beyond a tonic input for generalization or that the extra time is needed to reduce the dimensionality of the network dynamics. These insights provide a test bed to develop more efficient learning curricula for RNNs and eventually animal training, not only in terms of generalization but also learning speed. The minimal dimensionality was achieved by directly constraining the rank of the connectivity matrix in each task.[44] The low-rank connectivity has the added advantage of providing an interpretable mathematical framework for the analysis of network dynamics.[41–43] However, it is possible that variations in the implementation of the RNNs, such as specific types of regularization in unconstrained networks[55] and other details of the learning algorithm or neural architecture may equally constrain neural activity to low-dimensional dynamics and induce comparable or better generalization. From that point of view, the minimal rank constraint used here can be seen as a particular type of inductive bias[7,56–58] for temporal tasks.

How and where tonic inputs are originated remains to be elucidated. The electrophysiology data in this study were recorded exclusively from dorsomedial frontal cortex, while the RNN model is agnostic to the origin of the tonic inputs. Thalamic activity has been found to be a candidate for the tonic input that flexibly controls cortical dynamics.[33,59,60] Nevertheless, when the tonic input provides contextual information, as in the MWG + Ctxt task, other cortical and subcortical brain areas are likely recruited to provide a signal that integrates statistics from past events.[39,61] Concurrently, contextual modulation of firing rates has been characterized in different areas of prefrontal cortex.[21,40,62] Whether this contextual information is integrated and broadcast in a localized cortical region or emerges from distributed interactions of multiple brain areas will need to be determined by causal perturbation experiments across brain regions.

STAR★METHODS

Detailed methods are provided in the online version of this paper and include the following:

- KEY RESOURCES TABLE
- RESOURCE AVAILABILITY
  - Lead contact
  - Materials availability
  - Data and code availability
- EXPERIMENTAL MODEL AND SUBJECT DETAILS
- METHOD DETAILS
  - Recurrent neural network dynamics
  - Training
  - Flexible timing tasks for RNNs
  - Performance measure in timing tasks
  - Dimensionality of neural activity
  - Analysis of trained RNNs
  - Non-linear manifolds
  - Mean-field low-rank networks
  - Analysis of dynamics in mean-field low-rank networks
  - Behavioral task
  - Neural data
  - Geometrical analyses of neural trajectories
  - Adaptation experiment in monkeys
  - Inference of context signal
  - Linear decoders

SUPPLEMENTAL INFORMATION

Supplemental information can be found online at https://doi.org/10.1016/j.neuron.2022.12.016.

ACKNOWLEDGMENTS

M.B., M.J., and S.O. were supported by the CRCNS project PIND, funded through the National Institutes of Health (NIMH: 1R01MH122025-01) and French Agence Nationale de la Recherche (ANR-19-NEUC-0001-01). M.B. was supported by the Ecole de Neurosciences de Paris. N.M. was supported by a Whitaker Health Sciences Fund Fellowship. H.S. was supported by a BBRF Young Investigator grant. S.O. was supported by the program "Ecoles Universitaires de Recherche" ANR-17-EURE-0017. M.J. 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.[44] The authors would also like to acknowledge and thank Alexandra Ferguson for her involvement in the design of the behavioral task in monkeys and early feedback on the analyses of trained recurrent neural networks.

AUTHOR CONTRIBUTIONS

M.B., M.J., and S.O. conceived the study of recurrent neural networks. M.B. trained, simulated, and analyzed the networks. N.M., H.S., and M.J. conceived the *in vivo* experiments. H.S. and N.M. collected the behavioral and electrophysiological data. N.M. analyzed the physiology data. M.J. and S.O. supervised the project. All authors were involved in interpreting the results and contributed to the writing of the manuscript.

DECLARATION OF INTERESTS

The authors declare no competing interests.

INCLUSION AND DIVERSITY

We support inclusive, diverse, and equitable conduct of research. While citing references scientifically relevant for this work, we also actively worked to promote gender balance in our reference list.

Received: November 13, 2021
Revised: September 23, 2022
Accepted: December 8, 2022
Published: January 13, 2023

REFERENCES

1. Markman, E.M. (1989). Categorization and Naming in Children (MIT Press).
2. Körding, K.P., and Wolpert, D.M. (2004). The loss function of sensorimotor learning. Proc. Natl. Acad. Sci. USA *101*, 9839–9842.
3. Courville, A.C., Daw, N.D., and Touretzky, D.S. (2006). Bayesian theories of conditioning in a changing world. Trends Cognit. Sci. *10*, 294–300.
4. Lake, B.M., Salakhutdinov, R., and Tenenbaum, J.B. (2015). Human-level concept learning through probabilistic program induction. Science *350*, 1332–1338.
5. Monosov, I.E. (2020). How outcome uncertainty mediates attention, learning, and decision-making. Trends Neurosci. *43*, 795–809.
6. Lake, B.M., Ullman, T.D., Tenenbaum, J.B., and Gershman, S.J. (2017). Building machines that learn and think like people. Behav. Brain Sci. *40*, e253.
7. Sinz, F.H., Pitkow, X., Reimer, J., Bethge, M., and Tolias, A.S. (2019). Engineering a less artificial intelligence. Neuron *103*, 967–979.
8. Saxe, A., Nelli, S., and Summerfield, C. (2021). If deep learning is the answer, what is the question? Nat. Rev. Neurosci. *22*, 55–67.
9. 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. Curr. Opin. Neurobiol. *32*, 148–155.
10. Gallego, J.A., Perich, M.G., Miller, L.E., and Solla, S.A. (2017). Neural Manifolds for the Control of Movement. Neuron *94*, 978–984.
11. Saxena, S., and Cunningham, J.P. (2019). Towards the neural population doctrine. Curr. Opin. Neurobiol. *55*, 103–111.
12. Jazayeri, M., and Ostojic, S. (2021). Interpreting neural computations by examining intrinsic and embedding dimensionality of neural activity. Curr. Opin. Neurobiol. *70*, 113–120.
13. DiCarlo, J.J., and Cox, D.D. (2007). Untangling invariant object recognition. Trends Cognit. Sci. *11*, 333–341.
14. DiCarlo, J.J., Zoccolan, D., and Rust, N.C. (2012). How does the brain solve visual object recognition? Neuron *73*, 415–434.
15. 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*, 585–590.
16. Fusi, S., Miller, E.K., and Rigotti, M. (2016). Why neurons mix: High dimensionality for higher cognition. Curr. Opin. Neurobiol. *37*, 66–74.
17. Chung, S., Lee, D.D., and Sompolinsky, H. (2018). Classification and Geometry of General Perceptual Manifolds. Phys. Rev. X *8*, 031003.
18. Cayco-Gajic, N.A., and Silver, R.A. (2019). Re-evaluating Circuit Mechanisms Underlying Pattern Separation. Neuron *101*, 584–602.
19. 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*, 954–967.e21.
20. Nogueira, R., Rodgers, C.C., Bruno, R.M., and Fusi, S. (2021). The geometry of cortical representations of touch in rodents. Preprint at bioRxiv. https://doi.org/10.1101/2021.02.11.430704.
21. Rigotti, M., Rubin, D.B.D., Morrison, S.E., Salzman, C.D., and Fusi, S. (2010). Attractor concretion as a mechanism for the formation of context representations. Neuroimage *52*, 833–847.
22. Mante, V., Sussillo, D., Shenoy, K.V., and Newsome, W.T. (2013). Context-dependent computation by recurrent dynamics in prefrontal cortex. Nature *503*, 78–84.
23. Saez, A., Rigotti, M., Ostojic, S., Fusi, S., and Salzman, C.D. (2015). Abstract Context Representations in Primate Amygdala and Prefrontal Cortex. Neuron *87*, 869–881.
24. Remington, E.D., Egger, S.W., Narain, D., Wang, J., and Jazayeri, M. (2018). A Dynamical Systems Perspective on Flexible Motor Timing. Trends Cognit. Sci. *22*, 938–952.
25. 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. Proc. Natl. Acad. Sci. USA *117*, 23021–23032.
26. Badre, D., Bhandari, A., Keglovits, H., and Kikumoto, A. (2021). The dimensionality of neural representations for control. Curr. Opin. Behav. Sci. *38*, 20–28.
27. Flesch, T., Juechems, K., Dumbalska, T., Saxe, A., and Summerfield, C. (2022). Orthogonal representations for robust context-dependent task performance in brains and neural networks. Neuron *110*, 1258–1270.e11.
28. Naumann, L.B., Keijser, J., and Sprekeler, H. (2022). Invariant neural subspaces maintained by feedback modulation. Elife *11*, e76096.
29. Buonomano, D.V., and Maass, W. (2009). State-dependent computations: spatiotemporal processing in cortical networks. Nat. Rev. Neurosci. *10*, 113–125.
30. Mello, G.B.M., Soares, S., and Paton, J.J. (2015). A scalable population code for time in the striatum. Curr. Biol. *25*, 1113–1122.
31. Gouvêa, T.S., Monteiro, T., Motiwala, A., Soares, S., Machens, C., and Paton, J.J. (2015). Striatal dynamics explain duration judgments. Elife *4*, e11386.
32. Merchant, H., and Averbeck, B.B. (2017). The computational and neural basis of rhythmic timing in medial premotor cortex. J. Neurosci. *37*, 4552–4564.
33. Wang, J., Narain, D., Hosseini, E.A., and Jazayeri, M. (2018). Flexible timing by temporal scaling of cortical responses. Nat. Neurosci. *21*, 102–110.
34. Remington, E.D., Narain, D., Hosseini, E.A., and Jazayeri, M. (2018). Flexible Sensorimotor Computations through Rapid Reconfiguration of Cortical Dynamics. Neuron *98*, 1005–1019.e5.
35. 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 Biol. *17*, e3000054.
36. Sohn, H., Narain, D., Meirhaeghe, N., and Jazayeri, M. (2019). Bayesian Computation through Cortical Latent Dynamics. Neuron *103*, 934–947.e5.
37. Egger, S.W., Remington, E.D., Chang, C.J., and Jazayeri, M. (2019). Internal models of sensorimotor integration regulate cortical dynamics. Nat. Neurosci. *22*, 1871–1882 1871–1882.
38. Bi, Z., and Zhou, C. (2020). Understanding the computation of time using neural network models. Proc. Natl. Acad. Sci. USA *117*, 10530–10540.
39. Monteiro, T., Rodrigues, F.S., Pexirra, M., Cruz, B.F., Gonçalves, A.I., Rueda-Orozco, P.E., and Paton, J.J. (2021). Using temperature to analyse the neural basis of a latent temporal decision. Preprint at bioRxiv. https://doi.org/10.1101/2020.08.24.251827.
40. Meirhaeghe, N., Sohn, H., and Jazayeri, M. (2021). A precise and adaptive neural mechanism for predictive temporal processing in the frontal cortex. Neuron *109*, 2995–3011.e5.
41. Mastrogiuseppe, F., and Ostojic, S. (2018). Linking Connectivity, Dynamics, and Computations in Low-Rank Recurrent Neural Networks. Neuron *99*, 609–623.e29.

42. Schuessler, F., Dubreuil, A., Mastrogiuseppe, F., Ostojic, S., and Barak, O. (2020). Dynamics of random recurrent networks with correlated low-rank structure. Phys. Rev. Research *2*, 013111.

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

44. Dubreuil, A., Valente, A., Beiran, M., Mastrogiuseppe, F., and Ostojic, S. (2022). The role of population structure in computations through neural dynamics. Nat. Neurosci. *1–12*.

45. Jazayeri, M., and Shadlen, M.N. (2010). Temporal context calibrates interval timing. Nat. Neurosci. *13*, 1020–1026.

46. Jazayeri, M., and Shadlen, M.N. (2015). A Neural Mechanism for Sensing and Reproducing a Time Interval. Curr. Biol. *25*, 2599–2609.

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

48. Vyas, S., Golub, M.D., Sussillo, D., and Shenoy, K.V. (2020). Computation Through Neural Population Dynamics. Annu. Rev. Neurosci. *43*, 249–275.

49. Pollock, E., and Jazayeri, M. (2020). Engineering recurrent neural networks from task-relevant manifolds and dynamics. PLoS Comput. Biol. *16*, e1008128.

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

51. Hardy, N.F., Goudar, V., Romero-Sosa, J.L., and Buonomano, D.V. (2018). A model of temporal scaling correctly predicts that motor timing improves with speed. Nat. Commun. *9*, 4732.

52. Rajalingham, R., Piccato, A., and Jazayeri, M. (2022). Recurrent neural networks with explicit representation of dynamic latent variables can mimic behavioral patterns in a physical inference task. Nat. Commun. *13*, 5865.

53. Izquierdo, A., Brigman, J.L., Radke, A.K., Rudebeck, P.H., and Holmes, A. (2017). The neural basis of reversal learning: an updated perspective. Neuroscience *345*, 12–26.

54. Sarafyazd, M., and Jazayeri, M. (2019). Hierarchical reasoning by neural circuits in the frontal cortex. Science *364*, eaav8911.

55. 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. Nat. Neurosci. *18*, 1025–1033.

56. Neyshabur, B., Tomioka, R., and Srebro, N. (2015). In Search Of The Real Inductive Bias: On The Role Of Implicit Regularization In Deep Learning (ICLR (Workshop)).

57. Bordelon, B., and Pehlevan, C. (2021). Population codes enable learning from few examples by shaping inductive bias. Preprint at bioRxiv. https://doi.org/10.1101/2021.03.30.437743.

58. Canatar, A., Bordelon, B., and Pehlevan, C. (2021). Spectral bias and task-model alignment explain generalization in kernel regression and infinitely wide neural networks. Nat. Commun. *12*, 2914.

59. Rikhye, R.V., Gilra, A., and Halassa, M.M. (2018). Thalamic regulation of switching between cortical representations enables cognitive flexibility. Nat. Neurosci. *21*, 1753–1763.

60. Logiaco, L., Abbott, L.F., and Escola, S. (2021). Thalamic control of cortical dynamics in a model of flexible motor sequencing. Cell Rep. *35*, 109090 109090.

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

62. Bouchacourt, F., Palminteri, S., Koechlin, E., and Ostojic, S. (2020). Temporal chunking as a mechanism for unsupervised learning of task-sets. Elife *9*, e50469.

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

64. Kingma, D.P., and Ba, J.L. (2015). Adam: A method for stochastic optimization. Preprint at arXiv. https://doi.org/10.48550/arXiv.1412.6980.

65. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., et al. (2017). Automatic differentiation in PyTorch. Adv. Neural Inf. Process. Syst.

66. Schuessler, F., Mastrogiuseppe, F., Dubreuil, A., Ostojic, S., and Barak, O. (2020). The interplay between randomness and structure during learning in RNNs. Adv. Neural Inf. Process. Syst. *33*.

67. Rajan, K., Abbott, L., and Sompolinsky, H. (2010). Inferring stimulus selectivity from the spatial structure of neural network dynamics. Adv. Neural Inf. Process. Syst. *23*.

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

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

70. Rabinovich, M.I., Huerta, R., Varona, P., and Afraimovich, V.S. (2008). Transient cognitive dynamics, metastability, and decision making. PLoS Comput. Biol. *4*, e1000072.

71. Rabinovich, M., Huerta, R., and Laurent, G. (2008). Transient dynamics for neural processing. Science *321*, 48–50.

72. Darshan, R., and Rivkind, A. (2022). Learning to represent continuous variables in heterogeneous neural networks. Cell Rep. *39*, 110612.

STAR★METHODS

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Experimental models: Organisms/strains		
Rhesus macaque (*Macaca mulatta*)	Alpha genesis	N/A
Deposited data		
Electrophysiology data	This paper	Zenodo: https://doi.org/10.5281/zenodo.7359212
Software and algorithms		
MATLAB	MathWorks	https://www.mathworks.com/products/matlab.html
KiloSort	Pachitariu et al., 2016	https://github.com/cortex-lab/KiloSort
Code used to analyze the electrophysiology data	This paper	https://github.com/jazlab/MB_NM_HS_MJ_SO_ParamControlManifold; https://doi.org/10.5281/zenodo.7359212
Code used to run and analyze the neural networks	This paper	https://github.com/emebeiran/parametric_manifolds; https://doi.org/10.5281/zenodo.7375026
Other		
CerePlex Direct	Blackrock Microsystems	https://blackrockmicro.com/neuroscience-research-products/neural-data-acquisition-systems/cereplex-direct-daq/
Plexon V-Probes	Plexon	https://plexon.com/products/plexon-v-probe/
Eyelink 1,000 eye tracker	SR Research	https://www.sr-research.com/products/eyelink-1000-plus/

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources should be directed to and will be fulfilled by the lead contact, Srdjan Ostojic (srdjan.ostojic@ens.psl.eu).

Materials availability

The study did not generate new unique reagents.

Data and code availability

- Electrophysiology data have been deposited at Zenodo: https://doi.org/10.5281/zenodo.7359212 and are publicly available as of the date of publication. DOIs are listed in the key resources table.
- The original code related to the recurrent neural networks has been deposited at and is publicly available at https://github.com/emebeiran/parametric_manifolds as of the date of publication. DOIs are listed in the key resources table.
- The original code related to the analysis of electrophysiology data has been deposited at and is publicly available at https://github.com/jazlab/MB_NM_HS_MJ_SO_ParamControlManifold as of the date of publication. DOIs are listed in the key resources table.
- Any additional information required to reanalyze the data reported in this paper is available from the lead contact upon request.

EXPERIMENTAL MODEL AND SUBJECT DETAILS

All experimental procedures conformed to the guidelines of the National Institutes of Health and were approved by the Committee of Animal Care at the Massachusetts Institute of Technology. Experiments involved two awake behaving monkeys (macaca mulatta); two males (monkey G and H; weight: 6.8 and 6.6 kg; age: 4 years old) for the RSG task and the adaptation task. All animals were pair-housed and had not participated in previous studies. During the experiments, animals were head-restrained and seated comfortably in a dark and quiet room and viewed stimuli on a 23-inch monitor (refresh rate: 60 Hz). Eye movements were registered by an infrared camera and sampled at 1kHz (Eyelink 1000, SR Research Ltd, Ontario, Canada). The MWorks software package (https://mworks.github.io/) was used to present stimuli and to register eye position. Neurophysiology recordings were made by 1 to 3 24-channel

laminar probes (V-probe, Plexon Inc., TX) through a bio-compatible cranial implant whose position was determined based on stereotaxic coordinates and structural MRI scan of the animals. Analyses of both behavioral and electrophysiological data were performed using custom MATLAB code (Mathworks, MA).

METHOD DETAILS

Recurrent neural network dynamics

We trained recurrent neural networks (RNNs) consisting of $N = 1000$ units with dynamics given by Equation (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

$$z(t) = \sum_{i=1}^{N} w_i \varphi(x_i(t)), \quad \text{(Equation 5)}$$

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

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

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[63] 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

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

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,[44] we used the ADAM optimizer[64] in pytorch[65] 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.[44]

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

$$\sigma_{m_r n_s} = \sigma_0 \delta_{rs}, \quad \text{(Equation 8)}$$

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, which was useful to propagate errors back in time during learning.[66] 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 Figure 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.[44]

Flexible timing tasks for RNNs

We considered three flexible timing tasks, Cue-Set-Go (CSG,[33]), Measure-Wait-Go (MWG) and Measure-Wait-Go with context (MWG + Ctxt). 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,[33] that started at value -0.5 when the 'Set' signal is received, and grew until the threshold value $+0.5$ (Figures 1A, 1B, and 2A). 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 (Equation 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 1,550 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 1,500 ms after the second input, the 'Set' input indicated the beginning of the production epoch. Four different input intervals, ranging from 800 ms to 1,550 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 1,550 ms. In trials with those target intervals, the amplitude of the contextual input was $u_{ctxt} = 0$. The other four ranged between 1,600 ms and 3,100 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., Figure S1B), 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 (Figure 2B) using two complementary approaches: the variance explained by the first principal components, and the participation ratio. For the variance explained, 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. For the participation ratio (Figure 2B inset), we focused on the whole trial duration, to better show the differences in dimensionality between the different tasks and connectivity constraints.

We applied principal component analysis to the firing rates $\varphi(x_i(t))$ 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 Figure 2B.

Additionally, we quantified the dimensionality by means of the *participation ratio*,[67–69] defined as:

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

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

$$C_{ij} = [\varphi(x_i)\varphi(x_j)]. \qquad \text{(Equation 10)}$$

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 (Equation 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 (Equation 6) where the left (resp. right) connectivity vectors $\{\boldsymbol{m}^{(r)}\}_{r=1\ldots R}$ (resp. $\{\boldsymbol{n}^{(r)}\}_{r=1\ldots R}$) are orthogonal to each other.

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

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

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

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

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 (that represents the total input received by each unit) $\boldsymbol{x}(t)$ was then expressed in the basis given by the left connectivity vectors $\{\boldsymbol{m}^{(r)}\}_{r=1\ldots R}$ and the orthogonal input vectors components $\boldsymbol{I}_{\perp}^{(s)}$:[43,44]

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

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

$$\kappa_r(t) = \frac{\boldsymbol{m}^{(r)^T}\boldsymbol{x}(t)}{\boldsymbol{m}^{(r)^T}\boldsymbol{m}^{(r)}}, \quad \text{(Equation 14)}$$

and similarly for the input variables.

Inserting Equation (13) in Equation (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:

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

This analysis effectively reduces a high-dimensional dynamical system (Equation 11, N variables) to a lower dimensional dynamical system (Equation 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 (Equation 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:

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

To study the dynamical landscape of low-dimensional activity, we define the speed q[47] at a given neural state as a scalar function

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

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 Equation (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 κ in the recurrent subspace, and as a function of time (Figure S4). 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, Figure S4A). The manifolds obtained through this method are closely related to the concept of heteroclinic orbits.[70,71]

In rank-three networks trained on the MWG task, randomly initialized trajectories converged to a sphere-like manifold (Figure S4A). 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 Equation (17) (gray surface, Figure S4A; 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, Figure S4A) 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 (Figure S4C). 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 (Figure S4B).

Networks of rank R do not necessarily generate manifolds. As a counter-example, Figure S4 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[41–43]

$$m_i^{(1)},\ldots,m_i^{(R)},n_i^{(1)},\ldots,n_i^{(R)},I_i^{(1)},\ldots,I_i^{(S)} \sim \mathcal{N}(\mathbf{0},\mathbf{\Sigma}), \qquad \text{(Equation 18)}$$

where $\mathbf{\Sigma}$ is the $(2R+N_{in})\times(2R+N_{in})$ covariance matrix. We introduce the notation $P(\underline{m},\underline{n},\underline{I}) = P(m^{(1)},\ldots,m^{(R)},n^{(1)},\ldots,n^{(R)},I^{(1)},\ldots,I^{(S)})$ 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:

$$\widehat{\sigma}_{xy} = \frac{1}{N}\sum_{i=1}^{N}(x_i-\overline{x})(y_i-\overline{y}) \qquad \text{(Equation 19)}$$

Here $\overline{x}$ 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[43]). For that reason, we defined the overlap matrix $\boldsymbol{\sigma_{mn}}$ as the matrix with elements $\sigma_{m^{(r)}n^{(r')}}$, for $r,r' = 1,\ldots,R$, 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 , in the limit of large networks $N\to\infty$, the sum over N units in Equation (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:

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

The Gaussian integral in Equation (20) can be further expressed in terms of the covariances of the probability distribution as[42–44]

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

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

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

The dynamics plotted in Figure S6 were generated directly from Equation (21).

For constant inputs (as in Figure 5) 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 $\sum_{r=1}^{R} |\sigma_{In^{(r)}}| > 0$.

In Figure 5, we chose as parameters in panel A, $\sigma_{m_1 n_1} = 2.6, \sigma_{m_2,n_2} = 2.4$, $\sigma_{I^2} = 1.4 \cdot \text{cue}$, and in panel B: $\boldsymbol{\sigma_{nI}} = (0, 0.1) \cdot \text{cue}$, $\sigma_{I^2} = 0.02 \cdot \text{cue}^2$.

Analysis of dynamics in mean-field low-rank networks

We first focus on fixed points of the dynamics in Equation 21 in absence of inputs. The fixed points correspond to values of κ_r for which the r.h.s of Equation 21 is zero, which leads to:

$$\kappa_r = \langle \varphi' \rangle \sum_{r'=1}^{R} \sigma_{m^{(r')}n^{(r)}} \kappa_{r'}. \quad \text{(Equation 23)}$$

Since the gain $\langle \varphi' \rangle$ is implicitly a function of κ_r (Equation (22)), Equation (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, $\lambda_r > 1$[42,43]. Only the fixed points corresponding to the largest eigenvalue are stable, while all the other non-trivial fixed points are saddle points.[42,43]

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 (Figure S6B).[41,43,72] 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 Equation (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 (Figure S6C). 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 (Figure S6D).

We then studied the effects of the dynamics of mean-field low-rank networks receiving a constant input $v\boldsymbol{I}$ (Figure 5). The fixed point equation with an external input reads:

$$\kappa_r = \langle \varphi' \rangle \sum_{r'=1}^{R} \sigma_{m^{(r')}n^{(r)}} \kappa_{r'} + v\sigma_{In_r}. \quad \text{(Equation 24)}$$

Tonic inputs affect the dynamics of low-rank networks in two different ways. One effect is given by the second term in Equation (24), which is additive and directed along the vector $\boldsymbol{\sigma_{In}}$. The second effect corresponds to changes in the gain $\langle \varphi' \rangle$ which decreases monotonically toward zero when v is increased (Equation 22). Such changes correspond to the multiplicative factor in Equation (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 $r = 1,\ldots,R$ (Figures 5A and 5B), while subspace-specific inputs combine both effects (Figures 5D and 5E).

Behavioral task

Details about the behavioral task in monkeys can be found in[36] and.[40] Briefly, two macaque monkeys (monkey G and H) performed a time interval reproduction task known as the 'Ready-Set-Go' (RSG) task[36,40,46] 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° 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_{in}, between 'Ready' and 'Set'). The sample interval, t_s, was sampled from one of two discrete uniform distributions, with 5 values each between 480 and 800 ms for the fast context, and 800–1200 ms for the slow context. 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 Fast context, blue for Slow context).

The trial was rewarded if the relative error, $|t_{out} - t_{in}|/t_{in}$, 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_{in} 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_{in} values), we set the inter-trial intervals of the Short and Long contexts to 1,220 ms and 500 ms, respectively.

Neural data

For Figure 5, details about the neural recordings related to the RSG task (compared with MWG + Ctxt task) can be found in[36] and.[40] 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_{in}) was presented on a given trial, we averaged trials irrespective of the t_{in} 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 (Figure 6C, bottom left). We normalized the projection such that the mean is zero, and the two average contexts correspond to -0.5 and 0.5. 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 normalized the speed such that the maximum average speed is 1.

We performed principal component analysis (PCA) to visualize neural trajectories (Figures 5A and 5B, 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. Figure 5A was obtained by projecting the trajectories onto the top three principal components. Figure 5B was obtained by projecting the trajectories onto the first three principal components and then onto the subspace orthogonal to the estimated contextual axis.

Geometrical analyses of neural trajectories

To quantify the geometrical analyses of neural trajectories in Figure 5, we applied the 'kinematic analysis of neural trajectories' (KiNeT) framework developed in.[24] In particular, we used three different geometrical assessments to quantify the three properties observed in the data visualization: (i) low-dimensional modulation of trajectories based on context, (ii) invariance of trajectories along a subspace of neural state-space and (iii) speed modulation of trajectories based on the context signal.

For the low-dimensional modulation of trajectories, we computed the angle Θ between two states in the average trajectory of the slow context and the average trajectory of the fast context. We take as reference trajectory the slow context, and we parameterize the set of successive states by the reference time t_{ref}, the time it takes to reach such state after the first pulse. Secondly, at each state separated by a time lag τ from the reference state t_{ref}, we determine which is the closest point in the fast context trajectory, and store this context vector $\vec{\Delta}(t_{ref}+\tau)$ between those two different points. Finally, we compute the angle Θ, in absolute value, between the pairs of context vectors $\vec{\Delta}(t_{ref})$ and $\vec{\Delta}(t_{ref}+\tau)$. If context perfectly separates neural trajectories along a one-dimensional axis during the measurement epoch, the angle Θ should be zero for all values of time lags τ and reference states t_{ref}. Figure S8A shows the average angle for all possible reference times as a function of time lag, while Figure 5A shows the mean of the average angle for all different time lags. As a control, we compute surrogate context trajectories where we randomly shuffle the context labels of neural trajectories, and compute the angles Θ.

To test the invariance of trajectories along dimensions orthogonal to the context modulation axis (Figure 5B), we similarly calculate the distance between the average neural trajectories in each context after removing the projection of neural trajectories along the average context dimension, as calculated in Figure 5A. We use a different set of trials for the reference trajectory and the target neural trajectories. As a control, we calculate the distance between trajectories after removing the projection of neural trajectories along a random direction (average over 40 different random directions in state space). If trajectories largely overlap with each other, we expect to find a very short distance between trajectories after removing the context axis. To avoid artifacts due to high-dimensional noise in the recordings, and better compare with the three-dimensional PC visualization, we kept the three first principal components

of neural trajectories for this analysis. Figure S8B shows the computed distance as a function of the reference time during measurement.

Finally, we can compute the speed of neural trajectories (Figure 5C right) by comparing the time elapsed t_{comp} to reach the closest point to a state in the reference trajectory, compared to how long it took to reach that state in the reference trajectory t_{ref}. If trajectories evolve at the same speed in the reference and the comparison trajectory, t_{comp} and t_{ref} should be equal. If trajectories evolve twice faster in the comparison trajectory, t_{comp} should be half the value of t_{ref}.

Adaptation experiment in monkeys

For the adaptation experiment (Figure 6), we collected new data from DMFC in the same animals (n = 110 neurons in H, n = 129 in monkey G) while they underwent an uncued transition from two alternating contexts (similar to RSG) to a novel intermediate context. The interval distribution used in the two contexts differed from the ones used in the RSG task. For monkey H, the pre-switch distribution was [720;820;920] ms in the slow context, [560;640;720] ms in the fast context, and the post-switch distribution associated with the slow context was reduced to a single interval equal to 720 ms. For monkey G, the pre-switch distribution was [720;820;920;1,020] ms in the slow context, [480;560;640;720] ms in the fast context, and the post-switch distribution associated with the slow context was reduced to a single interval equal to 720 ms. The uncued transition occurred randomly during the session, on trial 714 for monkey H, and trial 603 for monkey G. We used 100 trials immediately following the uncued switch to estimate neural data post-switch. Pre-switch data included all trials before the switch, i.e., nearly 100 trials per interval of each context. All other experimental details were identical to the RSG experiment and can be found in Sohn et al.[36] and Meirhaeghe et al.[40]

Inference of context signal

In order to infer the context signal from the data, we add an unsupervised predictive module (Figure 7). This module can be split into two mechanisms. First, it estimates the average input interval to predict the input interval in the next trial, by performing a leaky estimation of the input interval over trials, t_{pred}. For that, the network requires one parameter, the update timescale τ. The predicted interval is updated at every trial n following:

$$t_{pred}^{[n]} = t_{pred}^{[n-1]} + \frac{1}{\tau}\left(t_{in}^{[n-1]} - t_{pred}^{[n-1]}\right). \quad \text{(Equation 25)}$$

The second term corresponds to the prediction error or mismatch between the input interval and the predicted interval.

Secondly, the module maps the predicted interval to an input amplitude by applying an affine transformation. This mapping requires in general two parameters, the bias term s_0 and the slope β, which we assumed have been learned over the course of learning. The contextual input amplitude thus reads

$$s^{[n]} = \beta t_{pred}^{[n]} + s_0. \quad \text{(Equation 26)}$$

The β and s_0 parameters are set so that it corresponds to the context input values used during training (0 and 0.1 in this case) for the mean interval in the slow and fast contexts (1,150 ms and 2,350 ms, respectively).

Linear decoders

For Figures 6C left and 7C, 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:

$$\boldsymbol{w}_{\boldsymbol{Ctxt}} = [\boldsymbol{x}(t, \text{slow ctxt})] - [\boldsymbol{x}(t, \text{fast ctxt})] \quad \text{(Equation 27)}$$

where the square brackets indicate the average across time points and trials of a given context. After the estimation of the decoder, we projected the neural activity of an independent set of trials, different from the training set.