

Decision Making in Low-Rank Recurrent Neural Networks

ARJUN PURI

Bernstein Center for Computational Neuroscience

July 2026

1 INTRODUCTION

1.1 Overview

In this report, we investigate how low-rank RNNs, as an interpretable computational framework, learn to solve decision-making and working-memory tasks. Our work builds on [Dubreuil et al. \(2022\)](#), who used the low-rank RNN framework to examine the role of functional subpopulations in solving common cognitive neuroscience tasks. For this report, we scope our work to the learned latent dynamics of a single population and the statistics of its connectivity parameters.

In essence, the low-rank RNN framework relies on constraining the recurrent connectivity of the network to a low-dimensional subspace, allowing us to examine the latent trajectories of the full system both analytically and computationally. We find that the statistics of the recurrent connectivity vectors were sufficient to form an equivalent low-dimensional circuit to solve the decision-making task, but showed degraded performance on the parametric working memory task.

We first lay out the essential ideas of the low-rank RNN framework and how they are used. We then examine rank-one networks on a perceptual decision-making task. Finally, we turn to a parametric working-memory task with rank-two networks and contrast the two results.

2 METHODS

In this section, we first discuss the analytical structure of the low-rank RNN framework and then outline the computational method we employed to evaluate the results suggested by the analytical reduction.

In essence, the analytical reduction takes two steps: we start by explaining how an N -dimensional RNN with low-rank connectivity can be viewed through the lens of a trajectory in a low-dimensional latent space. Then, we reason about how the second-order statistics of the network’s loading space (connectivity parameters) give rise to a simple, low-dimensional system that performs the essential computation of the original network. This is where the interpretability of the low-rank RNN framework lies.

2.1 Model

For our model, we use $N = 512$ leaky neurons driven by recurrent activity and a weighted scalar input. $\mathbf{x}(t) \in \mathbb{R}^N$ is the population state, τ the neural time constant, $\mathbf{I}$ the input weights, $u(t)$ the scalar input, and $z(t)$ a linear readout with weights $\mathbf{w}$:

$$\begin{aligned} \tau \frac{d\mathbf{x}}{dt} &= -\mathbf{x} + \mathbf{J}\phi(\mathbf{x}) + \mathbf{I}u(t), \\ \phi(\mathbf{x}) &= \tanh(\mathbf{x}), \quad z(t) = \frac{1}{N} \mathbf{w}^\top \phi(\mathbf{x}). \end{aligned} \quad (1)$$

2.2 Latent Dynamics

The core of the low-rank RNN framework rests on constraining $\mathbf{J}$ to rank R via a factorization into left and right connectivity vectors $\mathbf{m}^{(r)}, \mathbf{n}^{(r)} \in \mathbb{R}^N$ for $r = 1, \dots, R$:

$$\mathbf{J} = \frac{1}{N} \sum_{r=1}^R \mathbf{m}^{(r)} (\mathbf{n}^{(r)})^\top = \frac{1}{N} \mathbf{M} \mathbf{N}^\top \quad (2)$$

where $\mathbf{M}, \mathbf{N} \in \mathbb{R}^{N \times R}$. For $R = 1$, this reduces to $\mathbf{J} = \frac{1}{N} \mathbf{m} \mathbf{n}^\top$.

In essence, this rank constraint ensures that the population activity $\mathbf{x}(t)$ stays in the low-dimensional subspace spanned by the columns of $\mathbf{M}$ and the raw input vector $\mathbf{I}$. We define the rate projection $\hat{\mathbf{x}}(t) = \frac{1}{N} \mathbf{N}^\top \phi(\mathbf{x}(t))$ as the dynamical target for the latent coordinates $\boldsymbol{\kappa}(t) \in \mathbb{R}^R$. The population state then is

$$\mathbf{x}(t) = \mathbf{M} \boldsymbol{\kappa}(t) + \mathbf{I}v(t). \quad (3)$$

Substituting this decomposition into the RNN equation and matching coefficients along $\mathbf{M}$ and $\mathbf{I}$ yields the latent dynamics below (see the Methods of [Dubreuil et al. \(2022\)](#) for details). We are then able to fully describe the low-rank network in terms of its latent dynamics, input dynamics, and readout.

$$\begin{aligned} \tau \dot{\boldsymbol{\kappa}} &= -\boldsymbol{\kappa} + \hat{\boldsymbol{\kappa}}, \\ \hat{\boldsymbol{\kappa}} &= \frac{1}{N} \mathbf{N}^\top \phi(\mathbf{M} \boldsymbol{\kappa} + \mathbf{I}v), \end{aligned} \quad (4)$$

$$\tau \dot{v} = -v + u(t), \quad (5)$$

$$z = \frac{1}{N} \mathbf{w}^\top \phi(\mathbf{M} \boldsymbol{\kappa} + \mathbf{I}v). \quad (6)$$

2.3 Loading Space

While the previous section allowed us to inspect what the low-rank RNNs are doing in latent space, this section shows how the population as a whole arrives at this computation. The crux of this analytical step is to show that the network of neurons can be described solely by its connectivity parameters in what [Dubreuil et al. \(2022\)](#) term the “loading space.”

Looking at the rate projection, $\hat{\boldsymbol{\kappa}}_r$, we notice that it is built solely from a sum over a function of the network’s connectivity parameters. This allows us to treat the rate projection as an empirical average over a distribution of the connectivity parameters, where each neuron becomes a point $\mathbf{a}_i = (n_i^{(1)}, m_i^{(1)}, \dots, n_i^{(R)}, m_i^{(R)}, w_i, I_i)$ in a $(2R + 2)$ -dimensional loading space (R dimensions for each n and m , with two added dimensions for w and I).

$$\hat{\kappa}_r(t) = \frac{1}{N} \sum_{j=1}^N n_j^{(r)} \phi \left(\sum_{r'=1}^R \kappa_{r'}(t) m_j^{(r')} + I_j v(t) \right) \quad (7)$$

In the linear regime of $\tanh(\phi(x) \approx x)$, this average expands into overlaps of the loading coordinates:

$$\begin{aligned}\hat{\kappa}_r &\approx \sum_{r'=1}^R \sigma_{n(r)m(r')} \kappa_{r'} + \sigma_{n(r)I} v, \\ \sigma_{ab} &\equiv \frac{1}{N} \sum_{j=1}^N a_j b_j,\end{aligned}\quad (8)$$

and likewise $z \approx \sum_r \sigma_{wm(r)} \kappa_r + \sigma_{wI} v$. The latent and readout dynamics are therefore controlled by the covariances of the loading-space cloud. The full nonlinear case replaces these by the same covariances times a population gain (see Methods of [Dubreuil et al. \(2022\)](#) for the full explanation). Below, we document the key results without derivation (single population, for clarity). We assume that the parameters come from a zero-mean Gaussian distribution.

$$\begin{aligned}\tau \dot{\kappa}_r &= -\kappa_r + \sum_{r'=1}^R \tilde{\sigma}_{n(r)m(r')} \kappa_{r'} + \tilde{\sigma}_{n(r)I} v, \\ z &= \sum_{r=1}^R \tilde{\sigma}_{wm(r)} \kappa_r + \tilde{\sigma}_{wI} v,\end{aligned}\quad (9)$$

with effective couplings $\tilde{\sigma}_{ab} = \sigma_{ab} \langle \Phi' \rangle(\Delta)$, average gain

$$\langle \Phi' \rangle(\Delta) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} dz e^{-z^2/2} \phi'(\Delta z), \quad (10)$$

and standard deviation of the recurrent currents

$$\Delta = \sqrt{\sum_{r'=1}^R \sigma_{m(r')}^2 \kappa_{r'}^2 + \sigma_I^2 v^2}. \quad (11)$$

For P subpopulations, [Dubreuil et al. \(2022\)](#) write more generally $\tilde{\sigma}_{ab} = \sum_{p=1}^P \alpha_p \sigma_{ab}^{(p)} \langle \Phi' \rangle_p$, with $\langle \Phi' \rangle_p = \langle \Phi' \rangle(\Delta^{(p)})$.

We, however, focus on a single population, which results in a shared gain applied across all second-order moments.

Intuitively, this analytical reduction shows that the high-dimensional RNN can be viewed as a low-dimensional dynamical system governed by the second-order moments of the loading space.

For rank-one and rank-two networks, [Dubreuil et al. \(2022\)](#) used the following reduced models. For the perceptual decision-making task (rank one),

$$\begin{aligned}\frac{d\kappa}{dt} &= -\kappa + \tilde{\sigma}_{nm} \kappa + \tilde{\sigma}_{nI} v(t), \\ z &= \tilde{\sigma}_{wm} \kappa + \tilde{\sigma}_{wI} v(t), \\ \tilde{\sigma}_{ab} &= \sigma_{ab} \langle \Phi' \rangle(\Delta), \\ \Delta &= \sqrt{\sigma_m^2 \kappa^2 + \sigma_I^2 v^2}.\end{aligned}\quad (12)$$

For the parametric working-memory task (rank two), with only the diagonal recurrent covariances kept nonzero,

$$\begin{aligned}\frac{d\kappa_1}{dt} &= -\kappa_1 + \tilde{\sigma}_{n(1)m(1)} \kappa_1 + \tilde{\sigma}_{n(1)I} v(t), \\ \frac{d\kappa_2}{dt} &= -\kappa_2 + \tilde{\sigma}_{n(2)m(2)} \kappa_2 + \tilde{\sigma}_{n(2)I} v(t),\end{aligned}\quad (13)$$

$$\Delta = \sqrt{\sigma_{m(1)}^2 \kappa_1^2 + \sigma_{m(2)}^2 \kappa_2^2 + \sigma_I^2 v(t)^2}. \quad (14)$$

2.4 Tasks

Our work focused on two different tasks: 1) perceptual decision-making and 2) parametric working-memory.

For the perceptual decision-making task, we created input trials by sampling the stimulus strength ($\bar{u}$) from a uniform distribution and then adding Gaussian noise ($u(t) = \bar{u} + \xi(t)$). We trained an RNN with a linear readout to predict the dominant evidence (the sign of $\bar{u}$) in a noisy signal as a binary decision (+1 or -1). The training objective was the mean-squared error between the linear readout and the stimulus-strength sign, $\text{sign}(\bar{u})$.

$$u(t) = \begin{cases} \bar{u} + \xi(t), & \text{if } 5 \leq t \leq 45, \\ \xi(t) & \text{otherwise.} \end{cases} \quad (15)$$

For the parametric working-memory task, we uniformly sampled two values, f_1 and f_2 , in each trial and presented them one after the other with a delay. We trained the network to output the difference between the values. The network needed to learn to encode the earlier value, f_1 , in its two-dimensional latent space and retrieve it during the readout to perform the subtraction. We found that training the network with a fixed delay did not reproduce the dynamics shown by [Dubreuil et al. \(2022\)](#). Instead, the network learned oscillatory dynamics in the latents that overfit to the fixed delay.

Concretely, f_1, f_2 are drawn uniformly from $\{10, 14, 18, 22, 26, 30, 34\}$ with $f_{\min} = 10, f_{\max} = 34$, mapped to centered inputs

$$u_i = \frac{1}{f_{\max} - f_{\min}} \left(f_i - \frac{f_{\max} + f_{\min}}{2} \right), \quad i = 1, 2, \quad (16)$$

and presented as two pulses

$$u(t) = \begin{cases} u_1 & 5 \leq t \leq 10, \\ u_2 & 60 \leq t \leq 70, \\ 0 & \text{otherwise,} \end{cases} \quad y = \frac{f_1 - f_2}{f_{\max} - f_{\min}}. \quad (17)$$

2.5 Training

All networks used $N = 512$ neurons. The rank-one perceptual decision network was trained for 300 epochs. The rank-two working-memory network was trained for 600 epochs. We used Adam with learning rate 5×10^{-3} and batches of 32. To isolate and interpret the recurrent dynamics, we trained only the left and right connectivity vectors, $\mathbf{M}$ and $\mathbf{N}$, while keeping $\mathbf{I}$ and $\mathbf{w}$ fixed. All parameters were initialized from a standard normal distribution except for $\mathbf{w}$, which was sampled with standard deviation 4 to give the readout sufficient room to reach the target signs of +1 and -1. Training minimized a mean-squared error on the readout, masked to the decision period at the end of each trial:

$$\mathcal{L} = \sum_{k,t} M_t (z_{k,t} - \hat{z}_{k,t})^2, \quad (18)$$

where k indexes trials, t indexes timesteps, $z_{k,t}$ is the network readout, $\hat{z}_{k,t}$ is the target, and $M_t \in \{0, 1\}$ is nonzero only on decision steps.

3 RESULTS

In this section, we discuss the computational results obtained by training rank-one and rank-two RNNs on the perceptual decision-making and parametric working-memory tasks, and relate them to the analytical reduction above.

3.1 A rank-one RNN accumulates evidence toward fixed-point attractors through the statistics of its loading space

Training rank-one RNNs on the perceptual decision-making task. We first look at a rank-one RNN trained to report the sign of

the mean stimulus strength from noisy evidence. We generated 200 training trials and 256 test trials, in which the stimulus strength $\bar{\mu}$ was applied during a fixed time interval. The network was trained to predict the sign of the stimulus strength during the final 15 decision steps of its output. For this task, the recurrent connectivity is constrained to rank one:

$$\begin{aligned} \tau \frac{d\mathbf{x}}{dt} &= -\mathbf{x} + \frac{1}{N} \mathbf{m} \mathbf{n}^\top \phi(\mathbf{x}) + \mathbf{I} u(t), \\ \phi(\mathbf{x}) &= \tanh(\mathbf{x}), \quad z(t) = \frac{1}{N} \mathbf{w}^\top \phi(\mathbf{x}). \end{aligned} \quad (19)$$

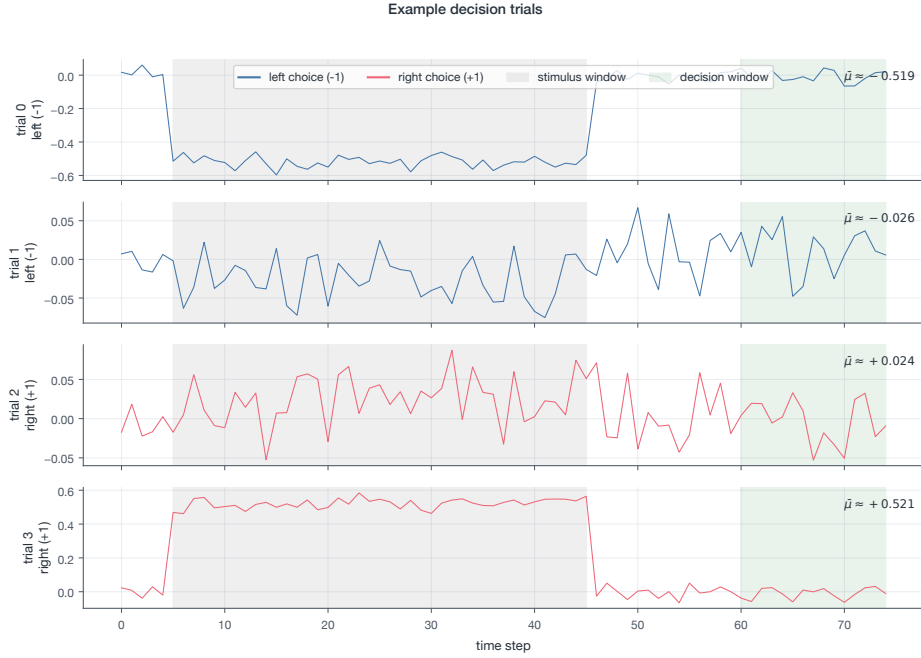


Figure 1. Four representative trials with strong and weak stimulus strengths. The stimulus was applied during one part of the trial, while the network had to predict its sign during a later decision window.

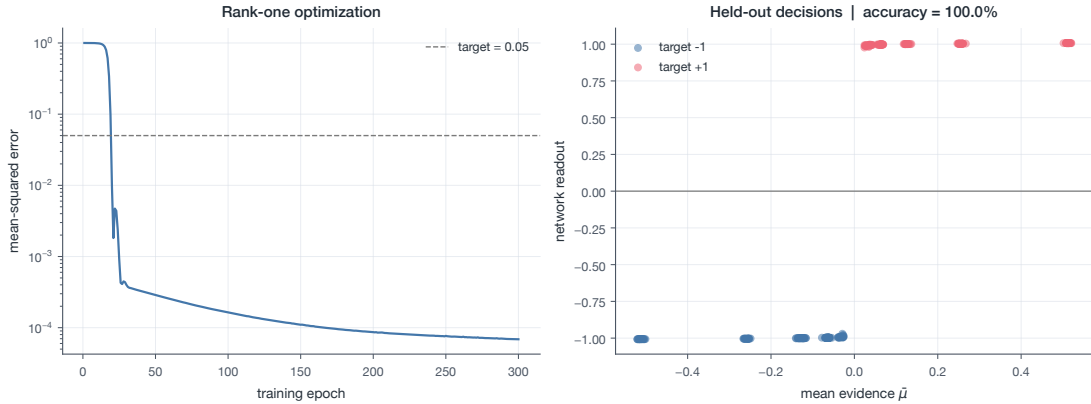


Figure 2. The mean-squared error converged in under 50 epochs, producing a network that could solve the task with 100% accuracy.

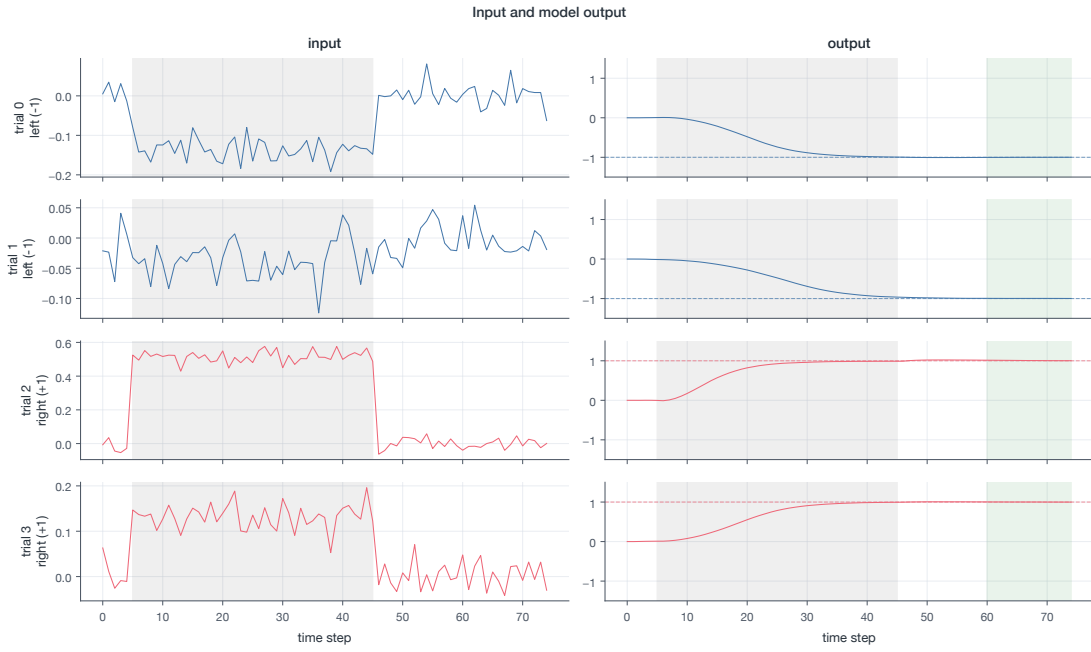


Figure 3. Model outputs (right) for corresponding inputs (left). The model chooses a direction and sticks with it.

Interpreting the latent trajectories. With a rank-one RNN that successfully performs this decision-making task, we now ask how it does so. Earlier, we showed that the population’s trajectory lies entirely in the subspace spanned by the left connectivity vector, m , and the input mixing vector, I . By projecting the population activity onto an orthogonalized basis that spans this subspace, we observe that the latent trajectories accumulate evidence and then quickly collapse to a decision along either the positive or negative

direction of m , depending on the stimulus sign.

Since the population trajectories lie in a two-dimensional subspace, we used PCA to find the directions of maximum variance and confirmed empirically that most of the variance in the data was explained by the first two components. Projecting the trajectories onto the subspace spanned by these two directions produced a latent-trajectory plot similar to the one above.

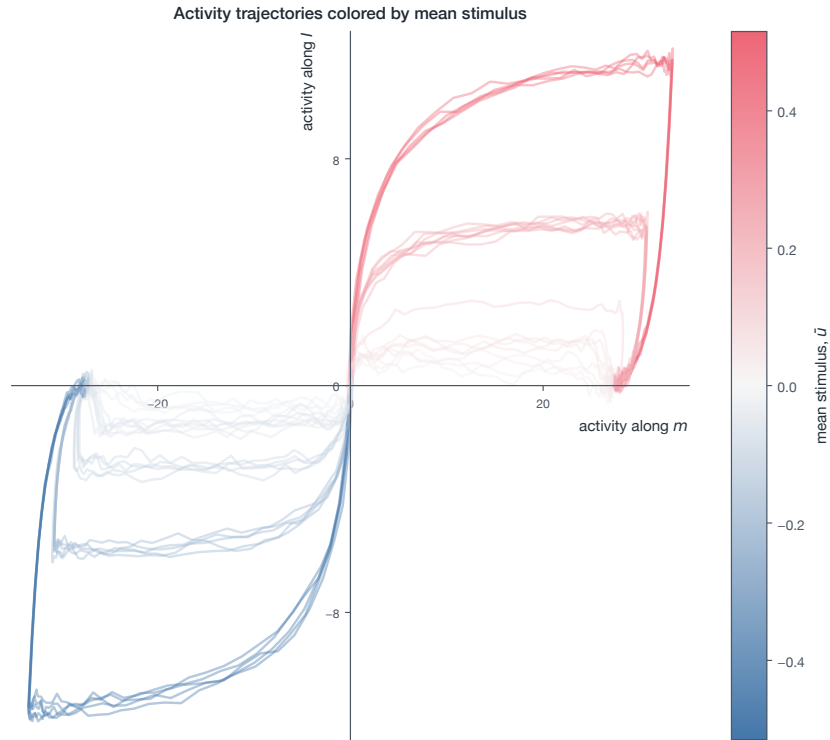


Figure 4. Test-trial population activity projected onto an orthonormal basis spanning $\text{span}(\mathbf{M}, \mathbf{I})$. Trajectories are colored by signed stimulus strength $\bar{u}$ and move toward the positive or negative recurrent axis according to the target sign.

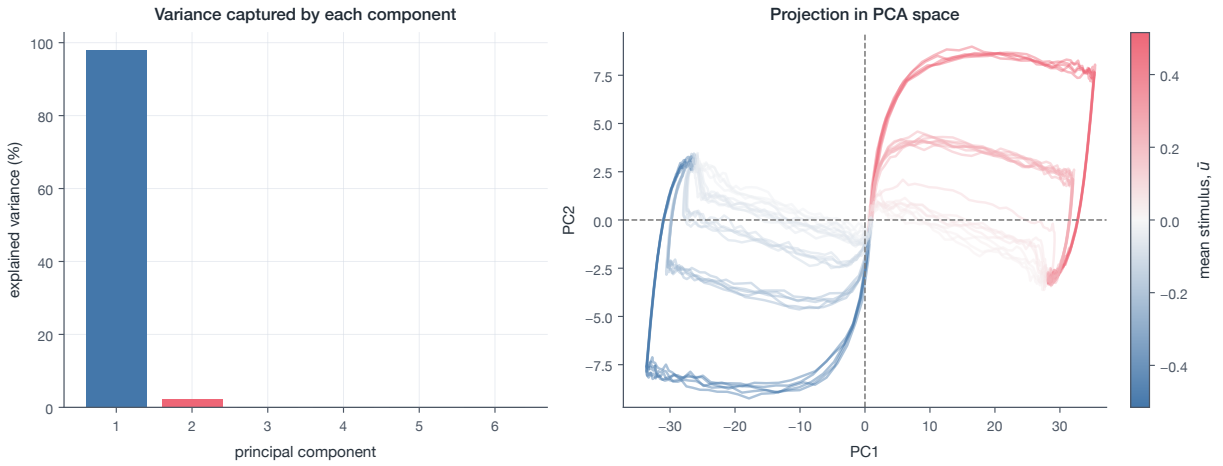


Figure 5. The first two principal components of the population activity across all trials and time points (right). Projecting activity onto these components reproduces the trajectory found in the basis of m and l , though rotated.

Covariance between connectivity parameters in loading space. From the previous result, we found an interpretable trajectory in latent space for the perceptual decision-making task. The network accumulates evidence in latent space and then collapses to one of two directions along m . Now, we ask how the population as a whole achieves this trajectory. To do this, we rely on the insight from the mean-field analysis described in the methods section. We

saw that the second-order moments of the loading space (under the assumption that the parameters are jointly Gaussian) carry the dynamics of the latent. First, we look at the second-order moments of the loading space. The values give us a clear interpretation: σ_{nl} brings the sensory evidence into the latent dynamics, σ_{nm} reinforces the evidence through positive feedback, and σ_{mw} aligns the latent variable with the output.

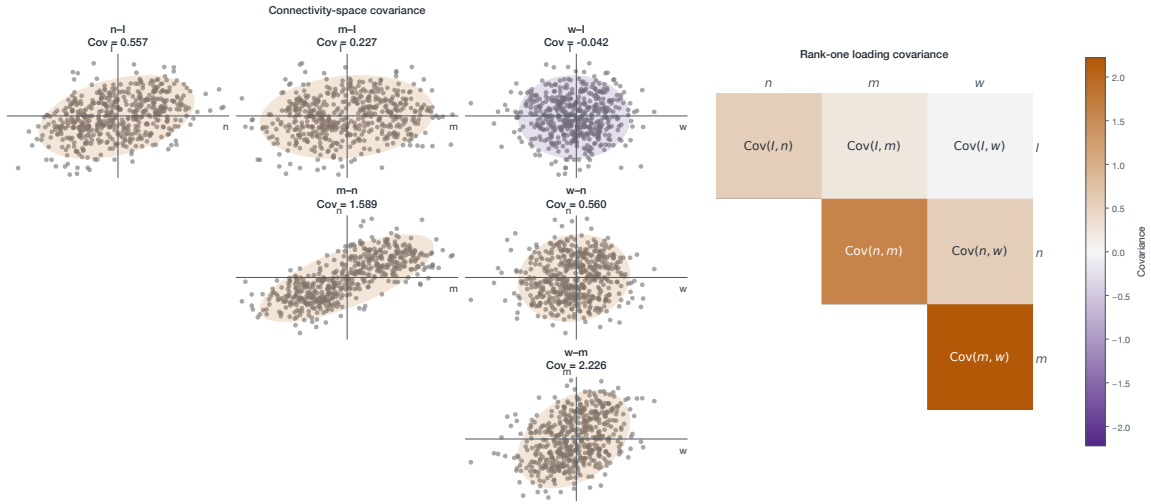


Figure 6. Loading space point clouds and the covariance heatmap for the current $N = 512$ rank-one run. The three task-relevant population covariances are approximately $(\sigma_{nl}, \sigma_{mn}, \sigma_{mw}) = (0.557, 1.589, 2.226)$.

Resampling from the distribution. Next, we show computationally that the distribution of the loading space is sufficient to solve the perceptual decision-making task. We fit a four-dimensional Gaussian distribution to the point cloud of the trained network’s connectivity vectors. From this, we reconstruct ten new

networks with the same number of neurons as the original network. We test the performance of the sampled networks on the test set and find that they achieve comparable accuracy in predicting the sign of the stimulus strength.

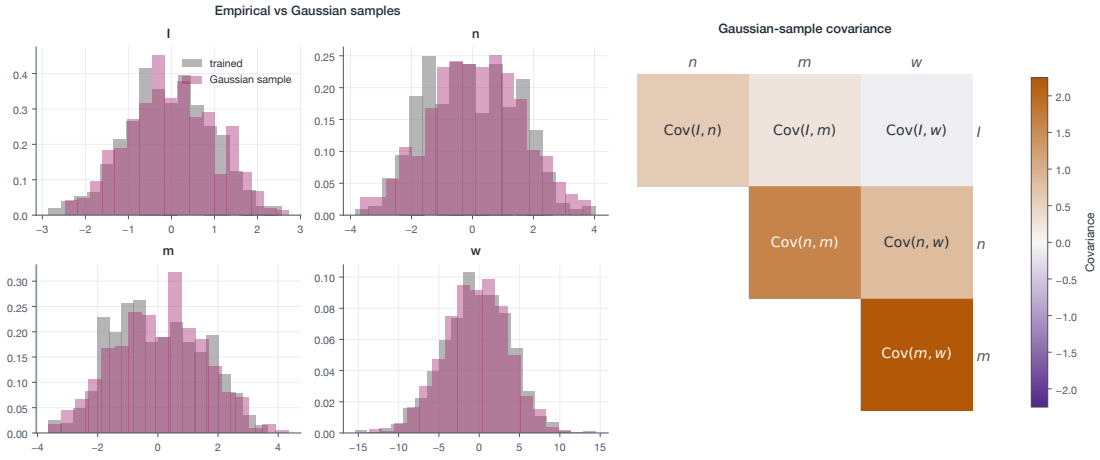


Figure 7. On the left, we show the distribution of connectivity vectors for a single network. On the right is the covariance heatmap.

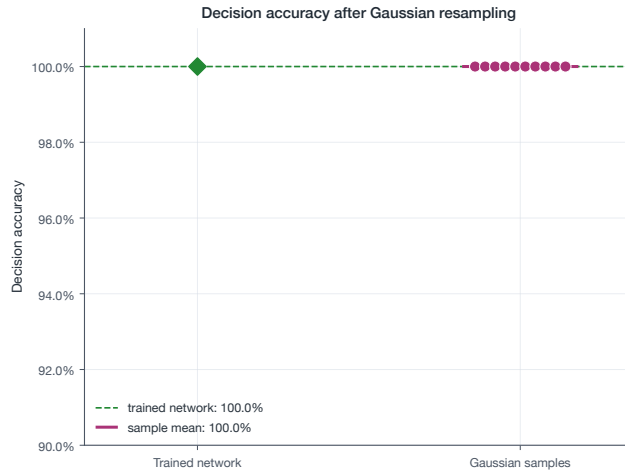


Figure 8. The Gaussian-sampled networks achieve 100% accuracy on the perceptual decision-making task, showing that the loading-space distribution has the necessary parametrization to perform the task.

Equivalent circuit. As discussed above in the Methods section, a mean-field analysis of the population dynamics allows us to describe the dynamics of the latent variable κ in terms of the covariances of the loading parameters and the average gain of the population.

$$\begin{aligned}\tau \dot{\kappa} &= -\kappa + \tilde{\sigma}_{nm} \kappa + \tilde{\sigma}_{nl} v \\ &= -(1 - \tilde{\sigma}_{nm}) \kappa + \tilde{\sigma}_{nl} v, \\ z &= \tilde{\sigma}_{wm} \kappa + \tilde{\sigma}_{wl} v \approx \tilde{\sigma}_{wm} \kappa,\end{aligned}\quad (20)$$

$\sigma_{n,m}$ sets an effective timescale $\tau_{\text{eff}} = \tau / (1 - \tilde{\sigma}_{nm})$: as $\tilde{\sigma}_{nm} \rightarrow 1$, the recurrent and leak terms are killed and κ perfectly retains the input. The vectors $\mathbf{w}$ and $\mathbf{l}$ are sampled independently from zero-mean distributions, so their overlap is zero in expectation, $\mathbb{E}[\sigma_{wl}] = 0$.

$$\tilde{\sigma}_{ab} = \sigma_{ab} \langle \Phi' \rangle(\Delta), \quad \Delta = \sqrt{\sigma_m^2 \kappa^2 + \sigma_l^2 v^2}, \quad (21)$$

$$\langle \Phi' \rangle(\Delta) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} dz e^{-z^2/2} \phi'(\Delta z). \quad (22)$$

The alignment of the latent and recurrent connectivity vectors, $\tilde{\sigma}_{nm}$, changes the timescale of the dynamics, while the alignment of the input and latent vectors, $\tilde{\sigma}_{nl}$, influences the direction of the driving force on the latent variable κ . $\tilde{\sigma}_{wm}$ determines the network's readout direction. The gain is what makes this a nonlinear system.

We simulate this one-dimensional dynamical system along with the filtered inputs, $\tau \dot{v} = -v + u$, using the covariance parameters of our trained network. We find that this equivalent circuit performs as well as the original high-dimensional RNN and produces very similar latent trajectories.

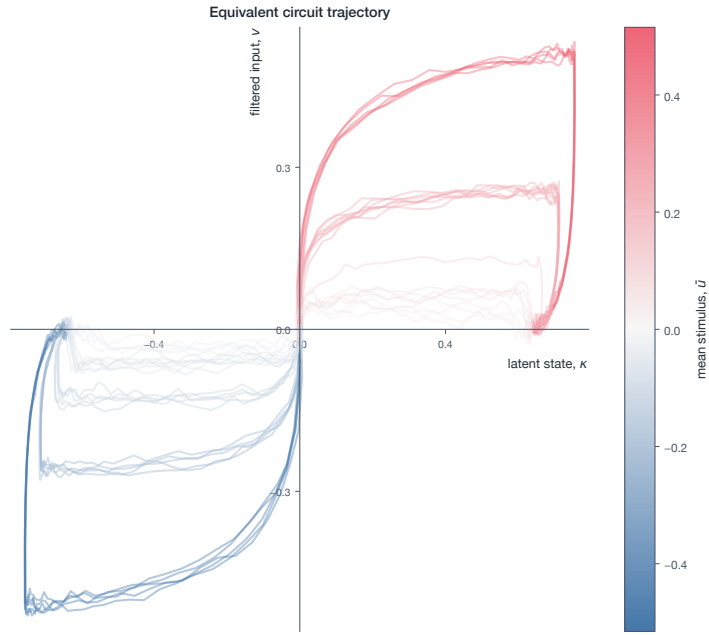


Figure 9. Latent trajectories of the one-dimensional equivalent circuit split by mean stimulus strength.

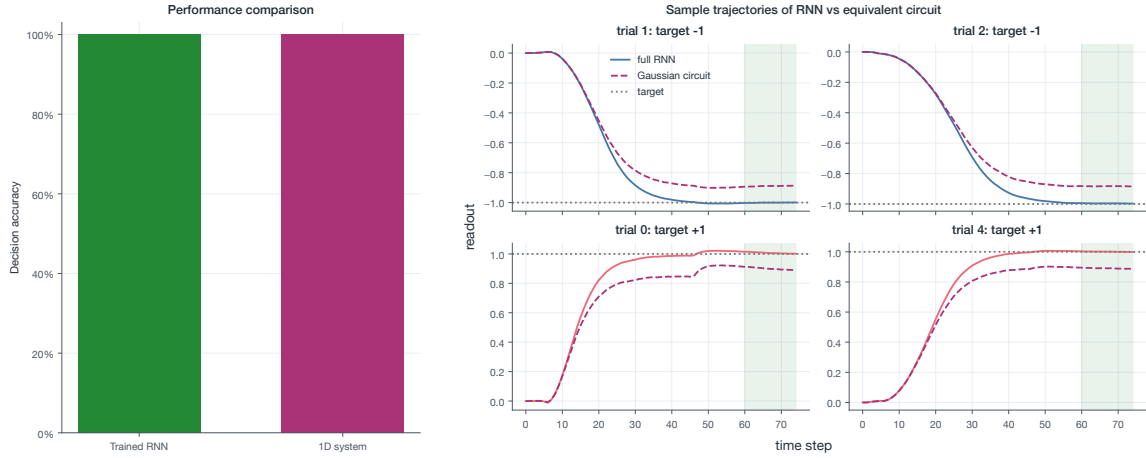


Figure 10. Accuracy between trained RNN and 1D system (left), and representative trajectories (right).

Fixed-point analysis. The final step in our analysis of rank-one networks on the perceptual decision-making task is to find the fixed points of the reduced dynamics and reason about their stability. Writing the system as $\dot{\kappa} = F(\kappa)$, we follow [Sussillo and Barak \(2013\)](#) and minimize the energy

$$q(\kappa) = \frac{1}{2} |F(\kappa)|^2. \quad (23)$$

Zeros of q are fixed points ($F(\kappa^*) = 0$). Linearizing around each fixed point, $\delta \dot{\kappa} = F'(\kappa^*) \delta \kappa$, allows us to determine stability by

looking at the sign of the eigenvalues of the Jacobian (in the 1D case, $F'(\kappa^*)$).

We find two stable fixed points on either side of the origin, with the origin acting as an unstable fixed point. This clearly showcases the behavior seen in the latent trajectories: early evidence pushes the system toward the attractor points. For the system to integrate inputs instead of just reporting the sign, we would expect a line attractor to emerge, as the network would need to settle on a continuum of points. This is what we should expect from the parametric working-memory task.

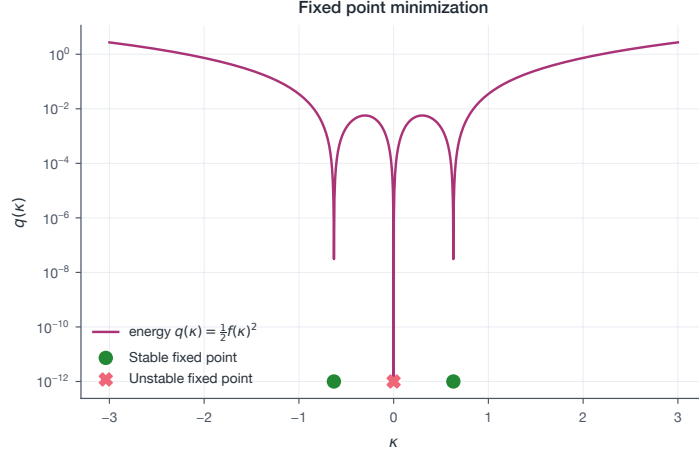


Figure 11. Energy function with the corresponding minima. Stability is determined using the sign of $F'(\kappa)$.

3.2 A rank-two RNN on a parametric working-memory task illustrates oscillatory dynamics in latent space

Now, we turn our attention to rank-two RNNs on a task that requires the network to predict a continuum of values. In the working-memory task, two stimuli, f_1 and f_2 , are presented one after the other with a delay. The network is trained to report the normalized difference between the two. To solve this task, it must be able to retain the value of f_1 . We trained a rank-two RNN with a fixed delay of 50 time steps between the two stimuli. Contrary to what was reported in [Dubreuil et al. \(2022\)](#), this network developed oscillatory dynamics instead of maintaining a persistent trace as in [Dubreuil et al. \(2022\)](#). We hypothesize that a variable-delay approach, as used in [Dubreuil et al. \(2022\)](#), trains the model to maintain a memory of the stimulus in the latents rather than exploit the timing to achieve a correct readout, as in our oscillating network.

Training a rank-two network on the parametric working-memory task. For this task, we initialized a rank-two network and trained it using the mean-squared error to output the normalized difference between the two input stimuli, $y = \frac{f_1 - f_2}{f_{\max} - f_{\min}}$, where $f_{\max}$ and $f_{\min}$ were the extrema of the distribution from which we

sampled the stimuli.

$$\begin{aligned} \tau \frac{d\mathbf{x}}{dt} &= -\mathbf{x} + \frac{1}{N} \left(\mathbf{m}^{(1)} (\mathbf{n}^{(1)})^\top + \mathbf{m}^{(2)} (\mathbf{n}^{(2)})^\top \right) \phi(\mathbf{x}) \\ &\quad + \mathbf{I}u(t), \\ \phi(\mathbf{x}) &= \tanh(\mathbf{x}), \quad z(t) = \frac{1}{N} \mathbf{w}^\top \phi(\mathbf{x}). \end{aligned} \quad (24)$$

The network had two recurrent degrees of freedom to use to solve the task. Population activity lives in the subspace spanned by $\mathbf{m}^{(1)}$, $\mathbf{m}^{(2)}$, and the raw input vector $\mathbf{I}$, so we can write

$$\mathbf{x}(t) = \mathbf{m}^{(1)} \kappa_1(t) + \mathbf{m}^{(2)} \kappa_2(t) + \mathbf{I}u(t), \quad (25)$$

with latent coordinates κ_1, κ_2 and input coordinate v obeying

$$\begin{aligned} \tau \dot{\kappa}_1 &= -\kappa_1 + \hat{\kappa}_1, \\ \tau \dot{\kappa}_2 &= -\kappa_2 + \hat{\kappa}_2, \\ \tau \dot{v} &= -v + u(t), \end{aligned} \quad (26)$$

$$\begin{aligned} \hat{\kappa}_r &= \frac{1}{N} (\mathbf{n}^{(r)})^\top \phi(\mathbf{m}^{(1)} \kappa_1 + \mathbf{m}^{(2)} \kappa_2 + \mathbf{I}u), \\ r &= 1, 2. \end{aligned} \quad (27)$$

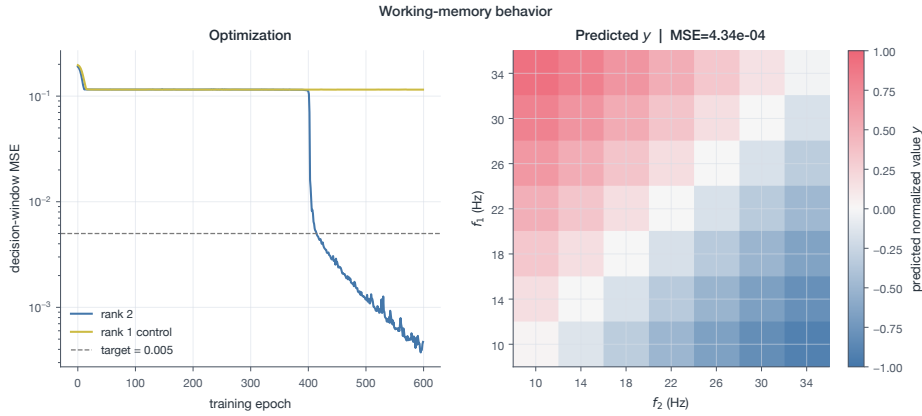


Figure 12. A rank-two network learned to perform the task, but a rank-one network could not (left). With only one degree of freedom, the rank-one network has no room to store a memory of the first stimulus for use at a later time point. On the right, we show the target values the model needed to learn for the corresponding inputs.

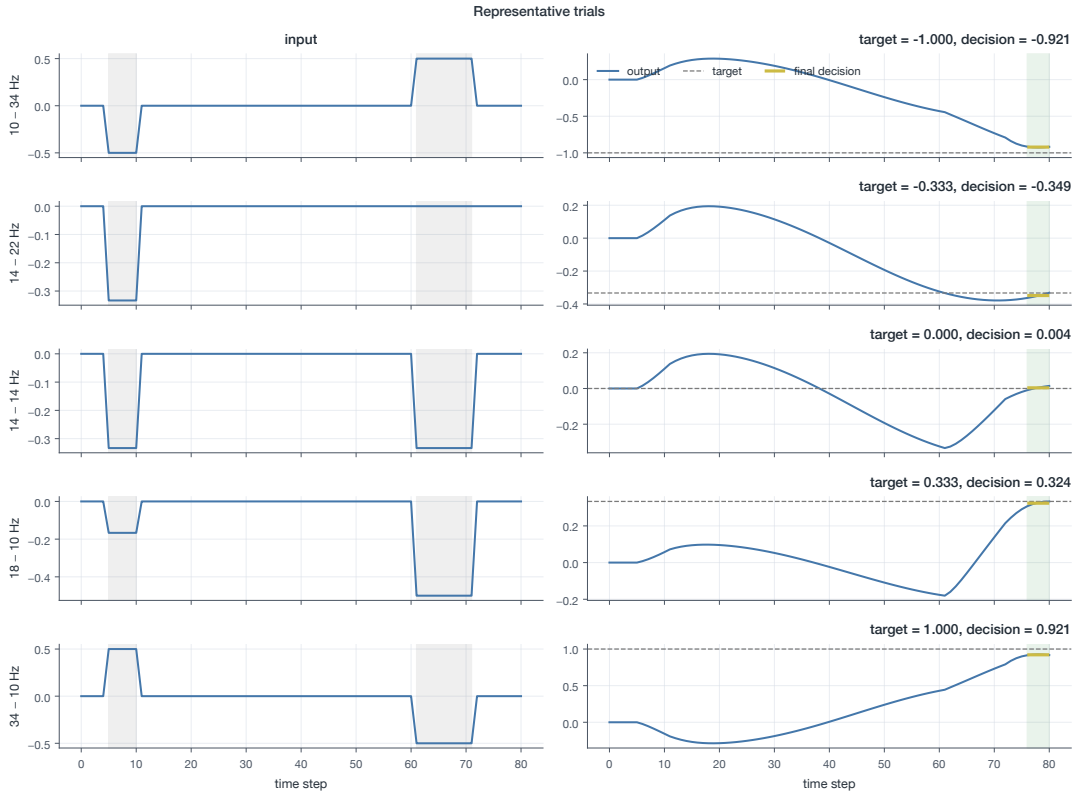


Figure 13. Representative inputs and model outputs across a range of values.

Interpreting the latent trajectories. Now, we unpack the underlying computation that the rank-two RNN performs by studying what the latent coordinates are doing in the low-rank subspace. We isolate the effect of each stimulus on the latents by zeroing out the other. We expected the network to maintain a persistent memory of one stimulus in a latent, such that a linear readout could

determine the difference between the two stimuli. This is what [Dubreuil et al. \(2022\)](#) found. However, we found that the latents settle into coupled, oscillating dynamics. Later, we show that the network overfit to the fixed delay between the stimuli and relied on the timing of the oscillations to solve the task.

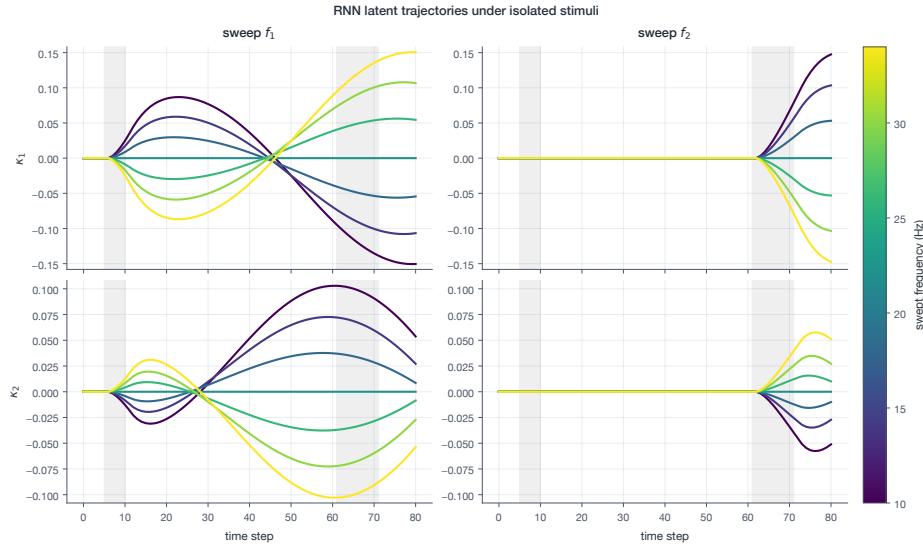


Figure 14. The isolated effects of each stimulus on the latents, found by holding $f_i = 22$ Hz (resulting in a mean-adjusted zero stimulus). Once a stimulus is presented, both latents begin a coupled rotating trajectory. On the left, we set $f_2 = 22$ Hz; on the right, we set $f_1 = 22$ Hz.

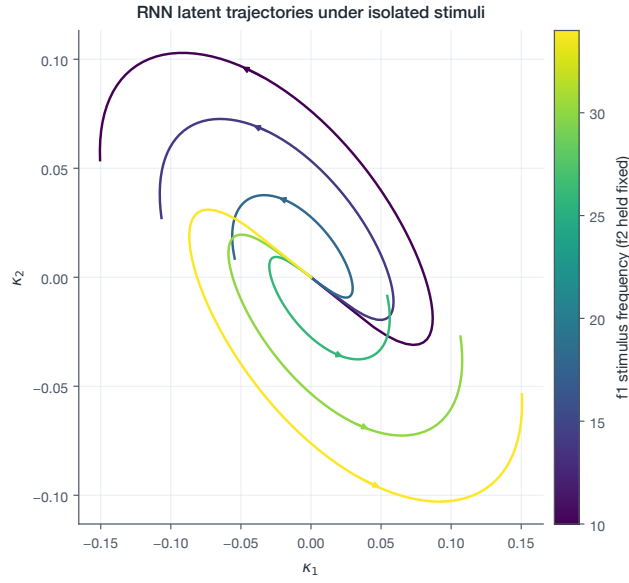


Figure 15. Oscillating trajectory of the combined latents, holding $f_2 = 22$ Hz while sweeping f_1 .

Covariance between connectivity parameters in loading space. Now, we turn to the second-order structure of the loading-space distribution to see how the population as a whole supports these dynamics. The trained network develops strong positive recurrent covariances within each latent mode (σ_{n_1, m_1} , σ_{n_2, m_2}), to-

gether with cross-latent covariances (σ_{n_1, m_2} , σ_{n_2, m_1}) that couple the two latents to support the rotational dynamics. The strong covariance between m_1 and w aligns this latent activity with the readout.

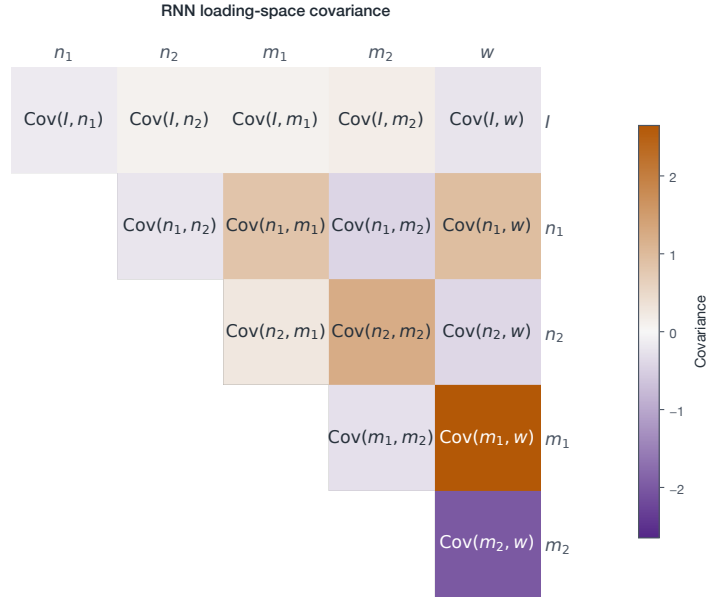


Figure 16. Upper-triangular covariance matrix of the six-dimensional loading space (l, n_1, n_2, m_1, m_2, w). Positive within-mode covariances slow the latent dynamics, while opposite-signed cross-mode covariances generate the observed oscillatory dynamics.

Resampling from the distribution. Now, we check whether the loading space distribution itself is sufficient to perform the necessary computation of the parametric working memory task. We fit a six-dimensional Gaussian to the loading vectors and sample ten

rank-two networks each with $N=512$ neurons. Unlike the rank-one network, these sampled networks do not retain the performance of the original network. Their median MSE is 0.0845, with no networks meeting the target of $\text{MSE} < 0.005$.

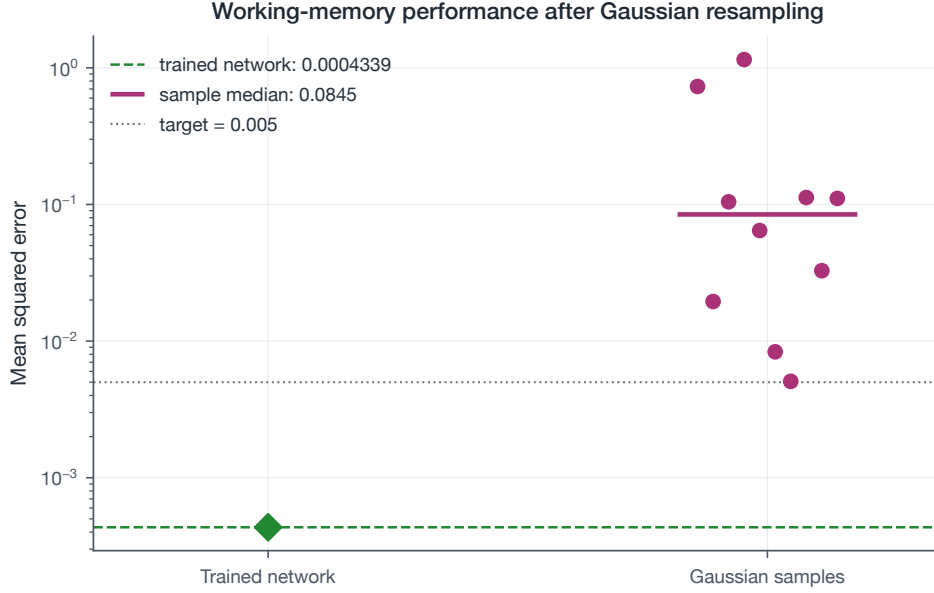


Figure 17. Performance of the trained RNN and ten Gaussian-sampled RNNs. The source network obtains MSE 0.000434, whereas the finite samples have median MSE 0.0845 and vary widely in performance.

Overfit performance. From the observation that our trained network relies on the oscillatory timescale to solve the task, we hypothesized that the network would only perform when the delay between the stimuli was tuned to a fixed period. We thus varied the

delay between the stimuli and tested the network’s performance (MSE) across a large range of delays. We found that the network’s performance oscillates with the delay - a clear signal that the network is overfit to the delay between the stimuli.

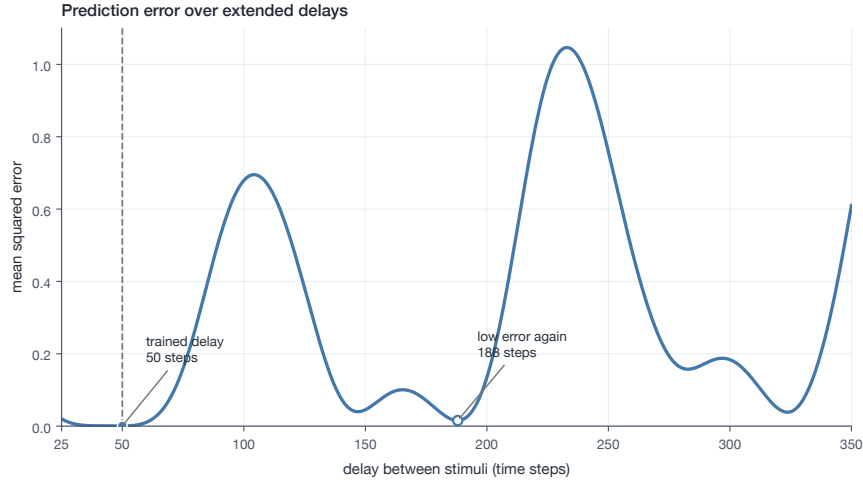


Figure 18. Mean-squared error of the network with stimuli separated by a varying delay. Task performance oscillates with the delay, showing a clear phase preference

Equivalent circuit. As our training procedure didn’t produce a network with the dynamics called out in [Dubreuil et al. \(2022\)](#), we next asked how a network with no cross-mode covariances solves the task. We used the covariances listed in [Dubreuil et al. \(2022\)](#) and constructed the equivalent circuit with only the self-feedback covariance terms.

$$\begin{aligned}\tau \dot{\kappa}_1 &= -\kappa_1 + \tilde{\sigma}_{n^{(1)}m^{(1)}} \kappa_1 + \tilde{\sigma}_{n^{(1)}I} v, \\ \tau \dot{\kappa}_2 &= -\kappa_2 + \tilde{\sigma}_{n^{(2)}m^{(2)}} \kappa_2 + \tilde{\sigma}_{n^{(2)}I} v.\end{aligned}\quad (28)$$

For our implementation of the circuit in [Dubreuil et al. \(2022\)](#), we used

$$\begin{aligned}\sigma_{n^{(1)}m^{(1)}} &= 1, & \sigma_{n^{(2)}m^{(2)}} &= 0.5, \\ \sigma_{n^{(1)}I} &= 0.5, & \sigma_{n^{(2)}I} &= 1.9.\end{aligned}\quad (29)$$

Here, $\sigma_{m^{(r)}}^2$ and σ_I^2 are the variances of the corresponding loading-space coordinates. Thus, Δ varies with the current latent state (κ_1, κ_2) and input coordinate v . The intrinsic timescale of each latent is influenced by the covariance in the following way:

$$\tau_{\text{eff},r}(\Delta) = \frac{\tau}{1 - \tilde{\sigma}_{n^{(r)}m^{(r)}}} = \frac{\tau}{1 - \sigma_{n^{(r)}m^{(r)}} \langle \Phi' \rangle(\Delta)}, \quad r = 1, 2. \quad (30)$$

Ignoring gain, the first recurrent covariance ($\sigma_{n^{(1)}m^{(1)}} = 1$) perfectly retains a memory of the input in its dynamics, while the second still retains the leak at a timescale of $\tau_{\text{eff},2} = 2\tau$. So, when the first stimulus is presented, the first latent preserves that value, and then when the second stimulus is presented, the second latent has its value in the transient deflection. This enables a linear readout to compute the difference.

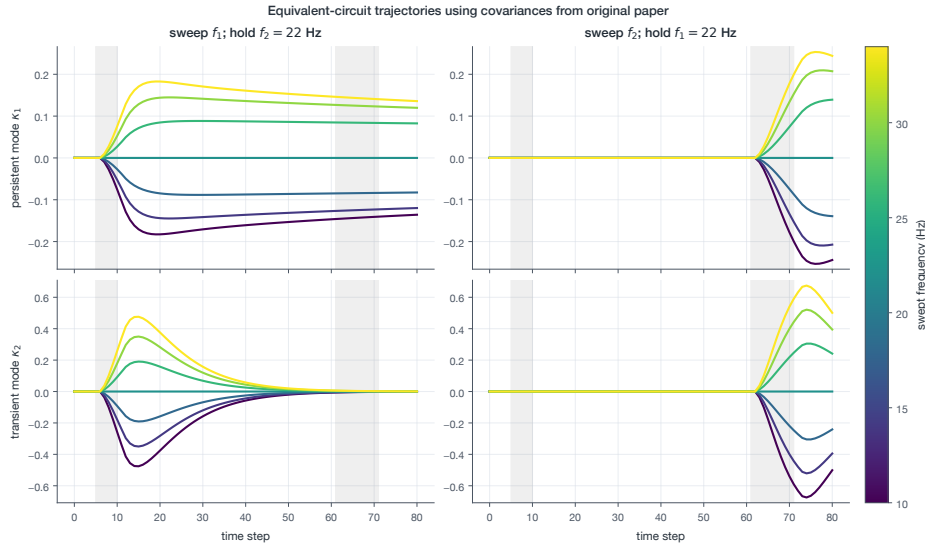


Figure 19. Trajectories of each latent under isolated stimuli (i.e zeroing out the effect of one via $f_i = 22\text{Hz}$). One latent reflects a persistent memory, while the other is a transient deflection. Together, a linear readout is able to solve the task.

Comparing performance. Such a network with a persistent and transient deflection is robust to a varying delay. Below, we conduct a similar analysis to what we did with the oscillating network,

but show that under arbitrary delay the persistent-transient network maintains consistent performance. Though, the equivalent circuit doesn't perform as well as the original trained RNN.

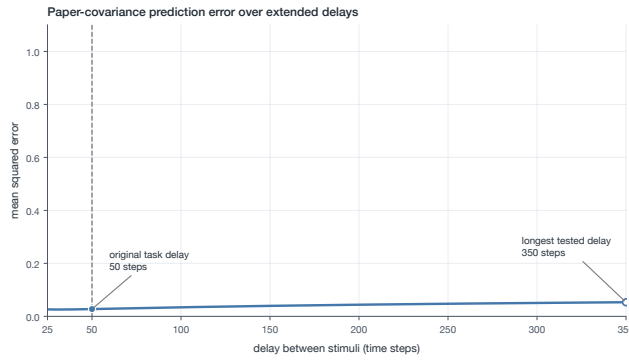


Figure 20. Persistent-transient equivalent circuit shows consistent performance across delays

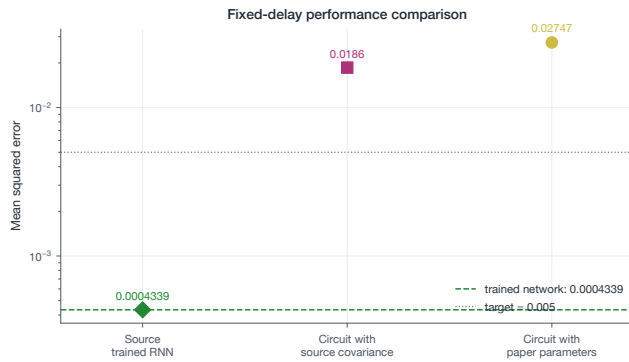


Figure 21. Equivalent circuit constructed from our original RNN's covariances and the covariances reported by Dubreuil et al. (2022) do not achieve target performance ($\text{MSE} < 0.005$)

4 DISCUSSION

This work used low-rank recurrent neural networks to interpret the computations performed by populations of recurrently connected neurons. We studied the low-dimensional latent dynamics and connected them to the statistics of the network's connectivity parameters. We then tested whether these population-level statistics were sufficient to reproduce the learned computations by

constructing reduced circuits and sampling new networks from fitted loading-space distributions. This approach succeeded for the perceptual decision-making network but was less reliable for the more dynamically sensitive working-memory solution.

In the perceptual decision-making task, the rank-one network developed a one-dimensional decision variable with two stable fixed points. A mean-field analysis conducted by Dubreuil et al.

(2022) helped us see how the covariances of the loading space could work to perform this computation: σ_{nl} brings the sensory evidence into the latent dynamics, σ_{nm} reinforces the evidence through positive feedback, and σ_{mw} aligns the latent variable with the output. The success of the Gaussian resampling approach showed us that the population-level statistics were sufficient to capture the core computational needs of the task.

In the working-memory task, our rank-two network learned oscillatory dynamics in the latent space. The fixed delay between stimuli in the task setup allowed the network to find a solution that overfit to this delay period, but that did not generalize to other delays like what was found in Dubreuil et al. (2022). A future analysis could look into eigenvalues of the linearized Jacobian to quantify the oscillatory dynamics, their stability and timescales. Using the loading space covariances from Dubreuil et al. (2022), we found that a more robust solution would learn a line attractor in one of the latents where the latent could maintain a persistent memory of the first stimulus, while leveraging the second latent as a transient, deflection in activity to perform the difference computation. We hypothesize that a more robust training procedure (ex. by varying delays) could encourage the network to learn this more robust solution.

Taken together, our results show that the low-rank framework

is a useful tool for interpreting how computations emerge from the collectivity activity and connectivity of neural populations. By looking at a single population, we were able to relate low-dimensional dynamics directly to the statistics of the loading space. Dubreuil et al. (2022) extend this framework to multiple functional subpopulations, offering a path toward understanding how more complex population structure supports computation.

5 CODE AVAILABILITY

The code for this project is available at arjunpur/low-rank-rnn.

6 AI USAGE

No AI was used in writing the report.

AI coding tools were used to help with plotting, analysis, and refactoring code, not with the core model and training.

7 REFERENCES

- Dubreuil, A., Valente, A., Beiran, M., Mastrogiuseppe, F., & Ostojic, S. (2022). The role of population structure in computations through neural dynamics. *Nature Neuroscience*, 25, 783–794. <https://doi.org/10.1038/s41593-022-01088-4>.
- Sussillo, D., & Barak, O. (2013). Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Computation*, 25(3), 626–649. https://doi.org/10.1162/NECO_a_00409.