RECURRENT NETWORK IMPLEMENTATIONS OF SEQUENTIAL GAUSSIAN PROCESS REGRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

In noisy environments, biological and artificial agents often approximate optimal Bayesian inference. However, whether and how neural circuits support such probabilistic computation remains unknown. Here, we investigate neural representations of uncertainty in recurrent neural networks trained to perform sequential Gaussian process (GP) regression, a rich setting where the target Bayesian computation is exactly specified and an expanding dataset must be compressed into a finite-capacity hidden state. We show that RNNs learn to implement reduced-rank recursive Bayesian regression to approximate GP inference. These networks learn an approximately linear belief representation over the leading eigenfunctions of the GP kernel, and this representation emerges through successive stage-like transitions during training. Causal perturbations to the hidden state confirm the identified representations are linearly steerable. Lastly, we develop a novel framework for examining how belief covariance is encoded in neural activity. Our results demonstrate how finite-state recurrent networks can implement sequential probabilistic inference, and offer a methodology for identifying probabilistic representations of uncertainty in artificial and biological dynamical systems.

1 INTRODUCTION

The noise and uncertainty inherent to daily life pose a challenge to perception and action. After all, the brain must contend not only with variability in the outside world but also with intrinsic noise in its sensory and neural apparatus (Tolhurst et al., 1983). To what degree, then, do biological agents calibrate their beliefs to account for this uncertainty? Theoretical treatments of this question invoke the language of Bayesian probability theory, arguing that explicit representations of uncertainty support better generalisation, flexibility under changing environments, and efficient use of sensory information (Koblinger et al., 2021).

Behavioral evidence has long seemed to suggest that humans make decisions approximately according to Bayesian inference (Ernst & Banks, 2002; Körding & Wolpert, 2004; Tavoni et al., 2022) and, thus, might internally represent uncertainty. However, behavior alone cannot adjudicate whether uncertainty is actually represented internally. Neural populations might maintain an explicit representation of uncertainty or instead arrive at similar predictions through history-dependent heuristics (Kalburge & Lengyel, 2025). Determining whether brains perform probabilistic inference thus rests upon probing neural representations for signatures of Bayesian computation.

Recurrent neural networks (RNNs) are artificial networks that propagate information required for future computation through finite-capacity internal representations. Such networks can serve as elucidative yet tractable testbeds for asking how a finite population of neurons can learn dynamical representations of uncertainty (Lambrechts et al., 2022; Hennig et al., 2023), representations that preserve the *sufficient statistics* relevant for future prediction while discarding nuisance details irrelevant to performing the task (Achille & Soatto, 2017; Kalburge & Lengyel, 2025).

We investigate emergent representations of uncertainty in RNNs trained to perform sequential Gaussian process (GP) regression (Rasmussen & Williams, 2008; Garnelo et al., 2018a; Müller et al., 2024), a probabilistic task where the target Bayesian computation is analytically specified, posterior uncertainty is explicit, and the network *must* compress an expanding dataset while making calibrated predictions at arbitrary query locations. We find that networks learn probabilistic repre-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

sentations corresponding to a reduced-rank approximation to GP inference (Rasmussen & Williams, 2008; Solin & Särkkä, 2020). We further demonstrate that this representation is learned sequentially according to a spectral bias (Saxe et al., 2014) and converges to a *linear* code that can be causally manipulated. We also provide a novel framework for examining how uncertainty—specifically covariance matrices—can be encoded in neural activity, refining ongoing debates about the format of neural codes of uncertainty. Taken together, this work demonstrates how recurrent networks can implement sequential inference and establishes a foundation for testing such claims rigorously in artificial and biological networks.

2 Related work

Growing interest in amortising probabilistic inference, particularly for Gaussian process regression on large datasets, has resulted in model families such as Neural Processes (Garnelo et al., 2018a;b) and prior-fitted networks (PFNs) (Müller et al., 2024; Hollmann et al., 2023). Recent work from Kang et al. (2026) made fruitful attempts to reverse-engineer how PFNs perform Bayesian (Gaussian process) regression across *layers* within-context. Similarly, growing interest in establishing a theory of in-context learning has converged on using Bayesian regression as a tractable task for characterising in-context transformer behavior (Ahn et al., 2023; Zhang et al., 2025). To our knowledge, our work is the first to reverse-engineer amortised in-context GP inference mechanisms in recurrent, rather than feedforward, networks. Hence, our analysis is concerned with how a *single* layer of neurons can dynamically represent uncertainty.

A distinct line of research in neuroscience uses artificial recurrent networks as *hypothesis generators* for neural dynamics (Sussillo & Barak, 2013; Sussillo et al., 2015; Maheswaranathan et al., 2019a;b; Kanitscheider & Fiete, 2017; Huang et al., 2025). Task-optimising artificial recurrent networks on inference tasks has been successful in reproducing dynamical motifs in biological data (Sussillo et al., 2015). However, the majority of prior work in this area has focused on tasks that can be solved with simple dynamical motifs (e.g., stable fixed points or line attractors) that mostly function autonomously, rather than being strongly input-driven at each time step (Arnold, 1998).

The GP inference task we consider is richer in several crucial ways. First, the target belief state is a posterior over functions, an infinite-dimensional object that the network must approximate within a finite-capacity hidden state. Second, the computation is strongly input-driven: Each observation arrives at an arbitrary location in the input domain, and the network must perform a location-dependent update to its belief state at every time step rather than relaxing toward a small set of attractor states. Third, in order to solve the task, the network must track its uncertainty along multiple interacting dimensions. How artificial networks do so remains poorly understood (Koblinger et al., 2021; Walker et al., 2023; Haefner et al., 2024), partly because prior studies (Orhan & Ma, 2017; Hennig et al., 2023) have typically explored neural representations over one-dimensional variables.

Our work is closest in spirit to (Mikulik et al. (2020), and relatedly Ortega et al. (2019)), which showed that meta-learning, a framework we employ on our networks, encourages Bayes-optimality.

3 Problem setup

We train recurrent (GRU) neural networks on the following meta-learning task. On each trial, context points $x_t \sim \mathcal{U}(-3, 3)$ have corresponding targets

$$y_t = f(x_t) + \varepsilon_t, \qquad f \sim \mathcal{GP}(\mathbf{0}, k), \;\; \varepsilon_t \sim \mathcal{N}(0, \sigma_n^2),$$

where $k(x, x')$ is the radial basis function (or squared exponential) kernel. Context is indexed by t to denote the RNN's recursion over time. Networks observe context and target points sequentially in a chained architecture: they first observe a context datum x_t and must predict $\hat{\mu}_t, \hat{\sigma}_t$, then observe target y_t (Figure 1A). They make predictions for $\hat{\mu}_t, \hat{\sigma}_t$ under a negative log-likelihood loss:

$$\ell_t = \tfrac{1}{2} \log \left(2\pi \hat{\sigma}_t^2\right) + \tfrac{(y_t - \hat{\mu}_t)^2}{2\hat{\sigma}_t^2}.$$

Given previous context $\mathbf{X} = (x_1, \ldots, x_{t-1})$, previous targets $\mathbf{y} = (y_1, \ldots, y_{t-1})$, and a new query point $x^\star$, the optimal prediction is

$$\begin{aligned} y^\star | x^\star, \mathbf{X}, \mathbf{y} &\sim \mathcal{N}\left(\hat{\mu}, \hat{\sigma}^2\right) \\ \hat{\mu} &= \mathbf{k}_\star^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y} \\ \hat{\sigma}^2 &= k(x^\star, x^\star) - \mathbf{k}_\star^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{k}_\star + \sigma_n^2, \end{aligned}$$

where $\mathbf{k}_\star = k(x^\star, \mathbf{X})$, and $\mathbf{K} = k(\mathbf{X}, \mathbf{X})$. While this prediction is exact, it requires that the recurrent state track a matrix ($\mathbf{K}$) that grows with context length. This is incompatible with a fixed-capacity recurrent network.

The Gaussian process regression problem can be formulated equivalently in weight space,

$$y_t = \mathbf{w}^\top \boldsymbol{\phi}(x_t) + \varepsilon_t, \qquad \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_M), \;\; \boldsymbol{\phi}(\cdot) = (\phi_1(\cdot), \phi_2(\cdot), \ldots)^\top$$

with basis functions $\phi_i(\cdot)$ chosen appropriately (see subsection 3.1 below and section 2.1 of Rasmussen & Williams (2008)). The squared exponential kernel requires an infinite number of basis functions. Since networks have only finite capacity, we hypothesised trained networks might perform weight-space regression using a truncation of the leading M basis functions $\phi_i(\cdot)$, which correspond to the eigenfunctions of the squared exponential kernel (subsection 3.1). Approximate inference in the regression problem then simplifies to recursively updating the M-dimensional mean $\boldsymbol{\mu}_t$ and $M \times M$ covariance matrix $\boldsymbol{\Sigma}_t$—the posterior sufficient statistics, sometimes referred to as the *belief state*—of $\mathbf{w}$ rather than memorizing the ever-expanding matrix $\mathbf{K}$.

In sections 4.2 and 4.3, we use ordinary least squares (OLS) linear regression to predict hidden layer activity in the RNN from task variables (e.g., different parameterisations of the posterior sufficient statistics). We call these regressions *encoding models*, in line with existing terminology (Kriegeskorte & Douglas (2019); Lange et al. (2023).

3.1 Approximate basis function expansion

An eigenfunction of the kernel $k(\mathbf{x}, \mathbf{x}')$ is a function $\phi(\cdot)$ satisfying $\int k(\mathbf{x}, \mathbf{x}')\psi_i(\mathbf{x}')d\mu(\mathbf{x}') = \lambda_i \psi_i(\mathbf{x})$ with respect to measure μ. These eigenfunctions are orthogonal to each other with respect to measure μ and can be chosen such that they are also orthonormal. By Mercer's theorem, we can write

$$k(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{\infty} \lambda_i \psi_i(\mathbf{x}) \psi_i(\mathbf{x}') = \sum_{i=1}^{\infty} [\sqrt{\lambda_i}\psi_i(\mathbf{x})][\sqrt{\lambda_i}\psi_i(\mathbf{x}')] = \sum_{i=1}^{\infty} \phi_i(\mathbf{x})\phi_i(\mathbf{x}').$$

For a zero-mean random process $f(x)$ with covariance $k(\mathbf{x}, \mathbf{x}')$, the KL expansion allows us to represent $f(\mathbf{x}) = \sum_i^\infty w_i \phi_i(\mathbf{x})$, where if $f \sim \mathcal{GP}(\mathbf{0}, k)$, weights $w_i \sim \mathcal{N}(0, 1)$ and the basis functions are $\phi_i(\mathbf{x}) = \sqrt{\lambda_i}\psi_i(\mathbf{x})$, derived from the eigenfunction expansion of the kernel above. For an infinite rank kernel like the squared exponential, the truncating the expansion at M basis functions is optimal in L^2 among all rank-M linear representations of f, and in practice can provide the correct solution up to floating point precision for an appropriate choice of M (Greengard & O'Neil, 2022). Thus, for RNNs trained with the RBF kernel, a reduced-rank approximation represents a viable solution.

To numerically approximate the eigenfunctions, we use the Nyström approximation (Rasmussen & Williams, 2008). If we let the measure μ be the probability measure, then we can use the approximation $\lambda_i \psi_i(\mathbf{x}) \simeq \frac{1}{S} \sum_{s=1}^{S} k(\mathbf{x}, \mathbf{x}_s)\psi_i(\mathbf{x}_s)$, so if we convert this to the matrix eigenproblem, $\mathbf{K}\mathbf{v}_i = \lambda_i^K \mathbf{v}_i$, where $\mathbf{K}$ is an $S \times S$ matrix comprised of the entries from a discretized grid over some domain of $\mathbf{x}$, then we can define our eigenfunctions to be scaled versions of our eigenvectors $\mathbf{v}_i$. Specifically, then

$$\psi_i(\mathbf{x}) \simeq \frac{\sqrt{S}}{\lambda_i^K} \mathbf{k}(\mathbf{x})^\top \mathbf{v}_i.$$

3.2 RNN architecture

We study RNNs comprised of three components: an encoder, which maps the first context to an initial hidden state for the recurrent module; a recurrent (GRU) module, which specifies the update

rule; and a readout, which predicts the mean and variance of the predictive posterior $p(y_t|x_t, \mathbf{X}, \mathbf{y})$ for each context point x_t.

Tokenization and hidden state evolution We sought an architecture that would be simple to train yet allow us to sweep across values of query x_{t+1} to produce predictions across the entire function. Therefore, we devised a tokenization scheme to alternate between two types of tokens, queries $\mathbf{h}_t^Q = (x_t, 1)$ and updates $\mathbf{u}_t^U = (y_t, 0)$ (Figure 1A). Note that the first query x_1 is absorbed into the encoder to produce $\mathbf{h}_0$.

Encoder & decoders The initial hidden state $\mathbf{h}_0$ must be learned, so we fit $\mathbf{W}_{\text{enc}}$ such that $\mathbf{h}_0 = \tanh\left(\mathbf{W}_{\text{enc}}\, x_1 + \mathbf{b}_{\text{enc}}\right)$. At every update step, we define two readout heads to decode $\hat{\mu}_t$ and $\hat{\sigma}_t$:

$$\hat{\mu}_t = \mathbf{w}_\mu \cdot \mathbf{h}_t + b_\mu,$$
$$\hat{\sigma}_t = \text{softplus}\left(\mathbf{w}_\sigma \cdot \mathbf{h}_t + b_\sigma\right) + \epsilon_{\text{floor}}, \qquad \epsilon_{\text{floor}} > 0$$

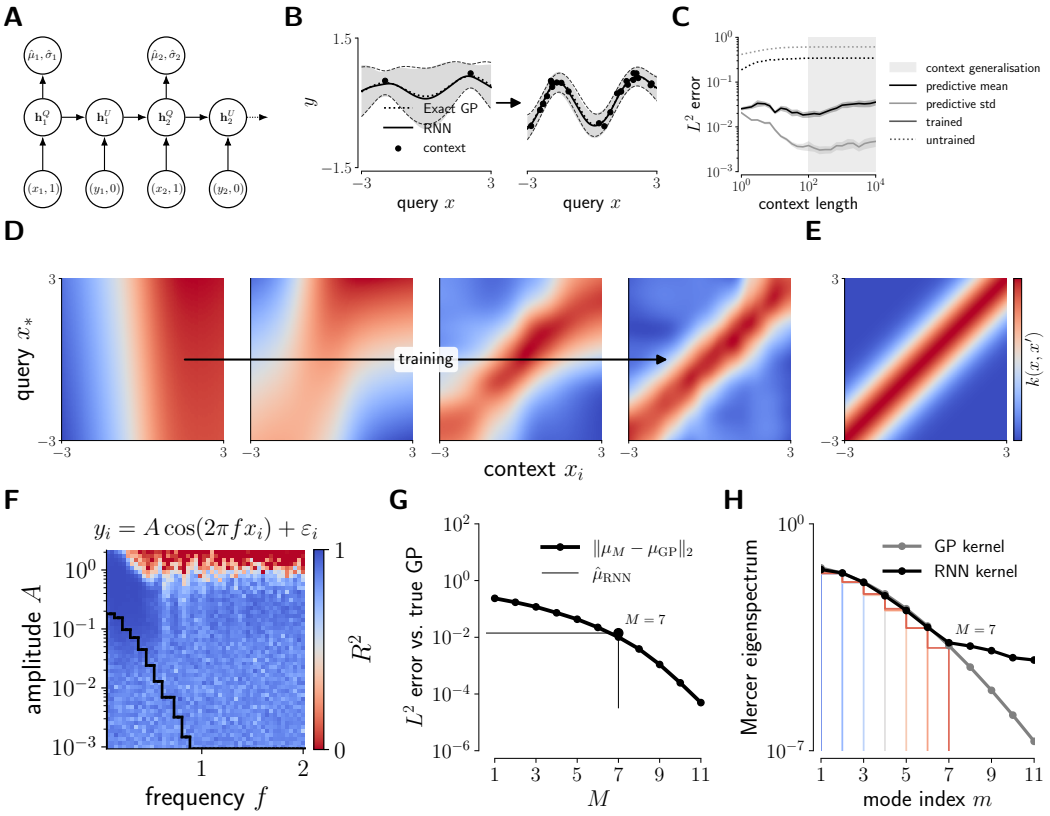


Figure 1: **RNN architecture and performance.** A) RNNs first see a context point x_t and have to predict $\hat{\mu}_t, \hat{\sigma}_t$, then passively observe y_t. B) Network (solid) closely matches optimal GP predictions (dotted) at small (left) and large (right) context lengths. C) RNNs generalise beyond train-time context length. D) From left to right, the RNN's estimated kernel over a fine query grid at successive stages of training. The fourth panel shows the fully trained RNN. E) The true GP kernel. F) Out-of-distribution amplitude generalisation at difference sinusoidal frequencies. Tiles below the solid black line are in-distribution. G) Estimating the rank of the approximation using predictive mean performance relative to GP. H) Estimating the rank using the eigendecomposition of the fully trained network's estimated kernel compared to that of the true GP kernel.

4 RESULTS

4.1 NETWORK BEHAVIOR

The trained recurrent network[1] proficiently approximated Gaussian process regression, tracking the true GP closely across context lengths and maintaining performance when generalising to context lengths much longer than those encountered during training (Figure 1B, C).

To visualize the network's learned kernel, it suffices to consider three time steps of computation with inputs (x_1, y_1, x^*). Holding $y_1 = 1$ fixed, the posterior mean $\mu^* = \mathbb{E}[y^* \mid x_1, y_1]$ is proportional to the kernel:

$$\mu^\star \;=\; k(x^\star, x_1)(k(x_1, x_1) + \sigma_n^2)^{-1} \;=\; \tfrac{k(x^\star, x_1)}{\sigma_l^2 + \sigma_n^2} \;\propto\; k(x^\star, x_1).$$

Collecting the network's predicted output $\hat{\mu}^*$ as a function of $x^\star$ and x_1 yielded the RNN's effective kernel, which we visualized as a 2D heatmap (Figure 1D, E). Plotting this over the course of training revealed a gradual progression from non-positive definite to the true GP kernel (Figure 1D, E). To characterise out-of-distribution generalisation, we probed the network with noisy sinusoids $y_t = A\cos(2\pi f x_t) + \varepsilon_t$, which approximately correspond to the eigenfunctions of the squared exponential kernel (Rasmussen & Williams, 2008). We then compared network predictions with optimal GP predictions over varying amplitudes A and frequencies f (Figure 1F). Since the network was only trained on GP function samples, eigenfunctions of frequency f have corresponding characteristic in-distribution amplitude $A = \sqrt{\lambda_i}$ (black line; Figure 1F). We therefore varied A to probe the network with increasingly low-probability amplitudes under the training distribution. We found that the network was indeed robust to a broad range of such low-probability amplitudes (Figure 1C).

Effective network rank We used two behavioral methods to ascertain the rank of the network's approximation. First, we found M such that the L^2 error of the RNN's predictive mean $\hat{\mu}_{\text{RNN}}$ relative to the GP matches that of the predictive mean of a rank-M approximation (Figure 1A)). We corroborated this estimate of M by eigendecomposing the effective kernel of the trained network (Figure 1D) and comparing its spectrum to the true GP's eigenspectrum (Figure 1B). Both methods indicated an effective rank of **7**.

4.2 LINEAR REPRESENTATION OF BELIEF STATE COORDINATES

Dimensionality of neural activity To determine the dimensionality of the network's representation, we probed it with function samples that were exact M-rank truncations of the KL expansion, $f_{\text{test}}(x) = \sum_{i=1}^{M} w_i \phi_i(x)$, rather than full GP samples. If the network's representation were to track posterior mean $\boldsymbol{\mu}_t$ of $\mathbf{w}$, it should converge to an encoding of $\mathbf{w}$ after sufficient context. For example, if $w_1 = 1$ and all remaining $w_i = 0$, the network would observe noisy realisations of the first eigenfunction, and its hidden representation should eventually converge to a representation of $\phi_1(x)$. Probing the hidden representation over a grid of w_i values would then trace out a potentially warped fixed point grid for the trained network.

Unlike the optimal belief filter, trained networks are only approximately permutation invariant: permutations of a context set converge to a neighborhood of fixed points (Figure 6D). Therefore, for context length C, we define the permutation-averaged fixed point, $\mathbf{h}^\star_C(\mathbf{w}) = \mathbb{E}_\pi[\mathbf{h}^\pi_C]$, where π denotes one particular ordering of the context data set.

We then tested whether weights w_i move along a direction $\mathbf{v}_i \in \mathbb{R}^H$ such that $\mathbf{h}^\star_C(\mathbf{w}) \approx \bar{\mathbf{h}} + \sum_{i=1}^{M} w_i \mathbf{v}_i$ in a local neighborhood. We estimated $\mathbf{v}_i$ for the top 15 eigenfunctions and indeed found that hidden state displacements organised into a linear coordinate system (an affine fit of $R^2 = 0.998$). We defined the dimensionality of the network's representation M_{eff} as the number of leading singular values of $\mathbf{V} = (\mathbf{v}_1, \cdots, \mathbf{v}_{15})$ needed to explain 99% of cumulative variance. Like the behavioral rank estimates (Figure 1), we estimated $M_{\text{eff}} = \mathbf{7}$ (Figure 2A); across 10 random seeds, $M_{\text{eff}} = 7.6 \pm 0.5$ (Table 1).

Sequential mode learning Training loss fell in several discrete steps (Figure 2A), reminiscent of *sequential mode learning* from deep linear network learning theory (Saxe et al., 2014). Separating

[1] For illustrative purposes, unless stated otherwise, results are for a single trained network.

the loss profile into distinct regions, we found that transitions roughly coincided with discrete expansions in M_{eff} (Figure 2A) and progressive linear decodability of individual weights w_i in order of decreasing kernel eigenvalue (i.e., a spectral bias; Figure 2B). While w_1 was robustly decodable for an untrained network, decoding R^2 plateaued for w_2 in region 3, exactly when M_{eff} increased from 1 to 2. A similar phenomenon occurred after the transition to region 4 (Figure 2A, B).

To visualise this sequential expansion, we restricted $M = 3$ so that function samples were mixtures of only the top 3 eigenfunctions. We plotted $\mathbf{h}^\star_C(\mathbf{w})$ for seven evenly spaced settings of $w_1, w_2, , w_3 \in [0, 1]$ projected onto the top 3 principal components (Figure 2C-F). At initialisation (region 1; Figure 2C), the network maps all states onto a small region in state space. By region 2, (Figure 2D), the representation expands along the first axis, and by region 3 (Figure 2E), the representation expands along a second dimension. By the end of training, the states $\mathbf{h}^\star_C(\mathbf{w})$ form a nearly Euclidean three-dimensional grid.

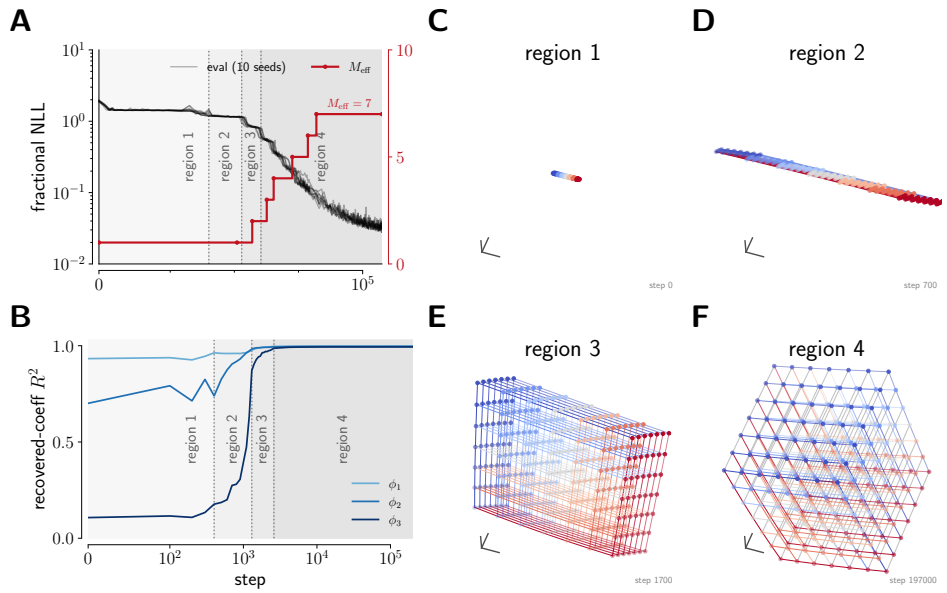


Figure 2: **Stage-like learning and representational expansion.** A) Training and evaluation loss curves (as a fraction of the GP's NLL) exhibit distinct stages for ten randomly initialised networks. Overlaid in red is the estimated effective dimension M_{eff} described in the main text. B) Decodability of the weights $w_{1,2,3}$ associated with the top 3 eigenfunctions $\phi_{1,2,3}$ over the course of training. C)-F) The network's permutation-averaged representations of weights for a fixed corpus at different stages of learning (marked by the associated regions in the loss curves in panels A and B).

Causal manipulation This linear posterior mean representation of $\mathbf{w}$ allows for simple *linear* causal steering via a context-invariant encoding map $\mathbf{W}^\mu_e : \forall t,\ \mathbf{h}_t = \mathbf{W}^\mu_e \boldsymbol{\mu}_t$. Then, to perturb $\mathbf{h}_t \to \mathbf{h}_t^{\text{target}}$,

$$\mathbf{h}_t^{\text{target}} = \mathbf{h}_t + \boldsymbol{\delta}, \qquad \boldsymbol{\delta} = \mathbf{W}^\mu_e(\boldsymbol{\mu}_t^{\text{target}} - \boldsymbol{\mu}_t).$$

We tested two such manipulations: perturbing $\boldsymbol{\mu}_t \to \mathbf{0}$ (Figure 3B) and $\boldsymbol{\mu}_t \to -\boldsymbol{\mu}_t$ (Figure 3C). Both manipulations appropriately changed output mean estimates without changing output standard deviation estimates (Figure 3D).

4.3 Emergent representation of uncertainty

In order to solve the task, the network must represent the covariance of the posterior distribution in some format. However, there are multiple formats to choose from, most notably the raw posterior covariance itself, $\boldsymbol{\Sigma}_t$, and its inverse $\boldsymbol{\Lambda}_t$. Prior work from Ma et al. (2006) proposed that neural activity linearly encodes the natural/canonical parameterisation $\frac{1}{2}\boldsymbol{\Lambda}_t$ (probabilistic population codes; PPCs), while Vértes & Sahani (2018) instead proposed linear representations of the expectation/moment parameterisation $\boldsymbol{\Sigma}_t + \boldsymbol{\mu}_t\boldsymbol{\mu}_t^\top$ (distributed distributional codes; DDCs). While $\boldsymbol{\Lambda}_t$ has a

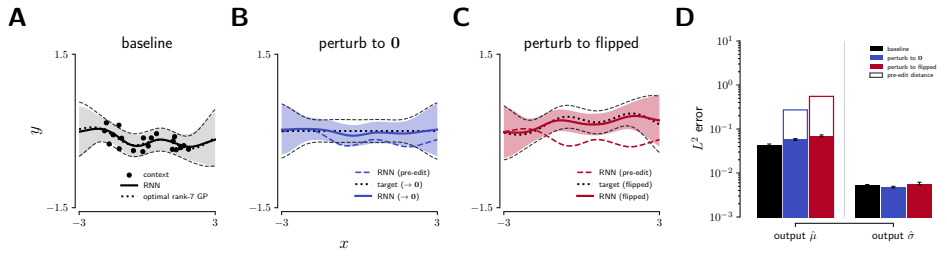


Figure 3: **Causal perturbations of the mean encoding.** A) An example context rollout. B) Perturbing $\boldsymbol{\mu}_t \to \mathbf{0}$ and C) perturbing $\boldsymbol{\mu}_t \to -\boldsymbol{\mu}_t$ are successful in changing output predictive mean without meaningfully changing predictive variance. D) Average L^2 error for each manipulation across many repeated trials. Baseline RNN error (grey) relative to the rank-7 GP approximation provides a useful soft lower bound for perturbation error. Total edit distance (base RNN to perturbed RNN; unfilled outline) provides a soft upper bound. Results are separated by output mean and output standard deviation. Perturbations successfully change output mean behavior (with error relatively close to baseline) while leaving output standard deviation approximately unchanged.

simpler rank-1 recursive update, its state can grow without bound, making it poorly suited to track in bounded neurons;[2] likewise, while $\boldsymbol{\Sigma}_t$ has bounded updates, its recursion is nonlinear and coupled with $\boldsymbol{\mu}_t$ (Equation 5). Beyond raw coordinates $\boldsymbol{\Sigma}_t$ or $\boldsymbol{\Lambda}_t$, the network may instead represent $\boldsymbol{\Sigma}_t^{1/2}$, which is useful since predictive variance $\sigma_t^2 = ||\boldsymbol{\Sigma}_t^{1/2}\boldsymbol{\phi}_t||^2 + \sigma_n^2$, or the whitening matrix $\boldsymbol{\Lambda}_t^{1/2}$. It may also represent $\log \boldsymbol{\Sigma}_t$,[3] ensuring Euclidean geometry does not violate positive-definiteness. In fact, since $\log \boldsymbol{\Sigma}_t = -\log \boldsymbol{\Lambda}_t$, these five candidates lie on a genuine spectrum,

$$\underbrace{\boldsymbol{\Sigma}_t \qquad \boldsymbol{\Sigma}_t^{1/2} \qquad \log \boldsymbol{\Sigma}_t}_{\text{expectation parameter / DDC}} \underbrace{\qquad \boldsymbol{\Lambda}_t^{1/2} \qquad \boldsymbol{\Lambda}_t}_{\text{natural parameter / PPC}},$$

corresponding, up to scalar multiplication and additive constants, to discrete points $\alpha \in \{1, 1/2, 0, -1/2, -1\}$ in the *power-Euclidean* family,

$$\mathcal{G}(\boldsymbol{\Sigma}; \alpha) = \frac{(\boldsymbol{\Sigma}^{-\alpha} - \mathbf{I})}{\alpha}$$

(Dryden et al., 2009; 2010). $\log \boldsymbol{\Sigma}_t$ is the limit as $\alpha \to 0$, the matrix equivalent of the Box-Cox transformation (Dryden et al. (2010); subsection A.3).

We fitted context-independent linear encoding models $\mathbf{W}_e : \forall t, \mathbf{h}_t = \mathbf{W}_e \mathbf{b}_t$, with belief state $\mathbf{b}_t = (\boldsymbol{\mu}_t, \operatorname{vech} \mathcal{G}(\boldsymbol{\Sigma}_t; \alpha)$, as a function of power parameter α. This provided a continuous knob between expectation ($\boldsymbol{\Sigma}_t$; DDCs) and natural ($\boldsymbol{\Lambda}_t$; PPCs) parameterisations. While this revealed an objective maximum $\alpha^\star = 1/4$, this analysis was most useful in ruling out a natural parameter representation: Encoding R^2 was flat and close to $\alpha^\star$ in the expectation parameter range $[0, 1]$, whereas encoding quality decayed steeply and monotonically in the natural parameter range $\alpha \in [-1, 0]$, reaching $R^2 = 0.71$ for $\boldsymbol{\Lambda}_t$ (Figure 4A, Table 2). At $\alpha^\star = 1/4$ encoding $R^2 = 0.83$ (compared to $R^2 = 0.35$ for the $\boldsymbol{\mu}_t$-only encoding; Table 1). We also found that the network rapidly shifted from the natural parameter range to the expectation parameter range as task performance improved over learning (Figure 4A, inset). Subsequent analysis set $\alpha = 1/4$, and we denote the transformed $\boldsymbol{\Sigma}_t$ as $\mathcal{G}(\boldsymbol{\Sigma}_t; 1/4)$.

Geometry of uncertainty As with the belief mean $\boldsymbol{\mu}_t$ (Figure 2C-F), we sought to visualise the geometry of uncertainty. In GP inference, posterior covariance is independent of targets y_t (subsection A.1). Thus, setting all $y_t = 0$ clamps $\boldsymbol{\mu}_t = \mathbf{0}$ without affecting posterior covariance. Under these conditions, variation in RNN hidden states $\mathbf{h}_t$ should only reflect changes in uncertainty (up to the network's permutation sensitivity). Moreover, repeated observations at the same query locations $x^\star$ shrink covariance along that location's feature direction roughly as $1/n$. Thus, we chose

[2] This can be remedied by scaling the update equations in Equation 2 by $1/t$ and having the neural population also track the number of context points t for readout.

[3] This refers to the matrix logarithm, not the element-wise logarithm.

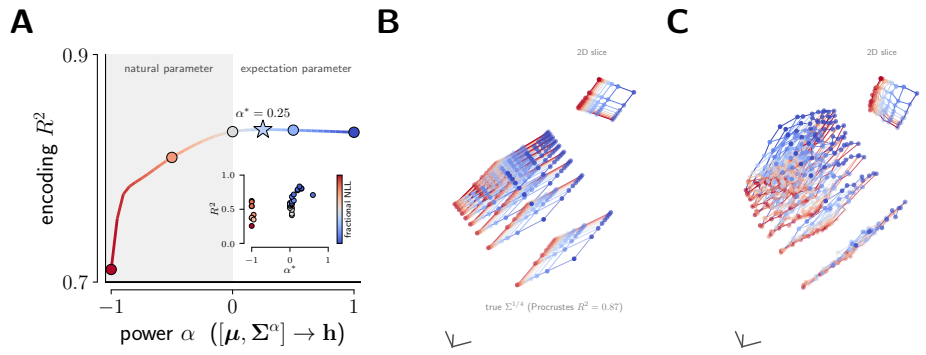


Figure 4: **Power-Euclidean covariance encoding and geometry.** A) Sweeping over power parameter α in the power-Euclidean family interpolates between expectation and natural parameter linear encoding models, revealing a strong preference for expectation parameter codes and a shallow maximum at $\alpha^\star = 1/4$; inset shows $\alpha^\star$ vs. encoding R^2 over the course of training. B) Isolating the geometry of uncertainty for an optimal encoding of $\mathcal{G}(\boldsymbol{\Sigma}_t; 1/4)$ reveals a flat but unevenly spaced geometry. The inset shows a 2D slice along the first two PCs when x_t is sampled uniformly from $\{-2.7, 2.7\}$ (the most orthogonal pair of query locations). The disjoint planes emerge when adding a third, only approximate query location at $x_t = 0$. Color denotes the number of observations n (blue $\to$ red). C) Applying the same visualisation approach to the RNN yields a similar geometry.

three query locations that were maximally orthogonal $x_t \in \{-2.7, 0, 2.7\}$ and clamped $y_t = 0$, then swept over a discrete number of observations $n \in \{1, 3, \ldots, 17\}$ and averaged the resulting hidden state over several random sequences of x_t (Figure 4C). We then applied this procedure to the true $\mathcal{G}(\boldsymbol{\Sigma}_t; 1/4)$ (Figure 4B). Like the optimal filter, the RNN learned a three-dimensional, nearly orthogonal, unevenly spaced grid, with a Procrustes similarity to the optimal filter of $R^2 = 0.87$.

Causal manipulation As with the belief mean, we attempted simple causal perturbations of uncertainty via the encoding $\mathbf{W}_e^\Sigma : \forall t, \mathbf{h}_t = \mathbf{W}_e^\Sigma \operatorname{vech}(\mathcal{G}(\boldsymbol{\Sigma}_t; 1/4))$:[4]

$$\mathbf{h}^{\text{target}} = \mathbf{h} + \boldsymbol{\delta}, \qquad \boldsymbol{\delta} = \mathbf{W}_e^\Sigma(\mathcal{G}(\boldsymbol{\Sigma}_t^{\text{target}}; 1/4) - \mathcal{G}(\boldsymbol{\Sigma}_t; 1/4)).$$

Whereas manipulations of $\boldsymbol{\mu}_t$ (Figure 3) did not affect the predictive variance behavior, perturbations to $\mathcal{G}(\boldsymbol{\Sigma}_t; 1/4)$ appropriately changed output $\hat{\sigma}_t$ but *also* changed output mean $\hat{\mu}_t$ (Figure 5B and 10A, B). However, clamping $\boldsymbol{\mu}_t$ to the unedited target (Figure 5C and 10C, D) successfully aligns output $\hat{\mu}_t$ without changing output $\hat{\sigma}_t$ (Figure 5D). We test three kinds of causal manipulations—reversion to the prior, reversion to $t - 5$, and rollout to $t + 5$. In all cases (Figure 5D), clamping the mean while editing $\mathcal{G}(\boldsymbol{\Sigma}_t; 1/4)$ yielded accurate causal manipulations of behavior.

5 Discussion

In this work, we construct a tractable setting—Gaussian process regression—to investigate neural coding of uncertainty. We find that recurrent neural networks robustly implement a finite-rank recursive approximation to Gaussian process inference and generalise to new inputs and context lengths (Figure 1). We demonstrate how this rank can be estimated independently from both behavior and the dimensionality of neural activity, yielding similar estimates (Figure 1, 2). Moreover, we demonstrate that stage-like learning dynamics correspond with progressive expansion in a linear encoding of the belief state mean $\boldsymbol{\mu}_t$ (Figure 2). To isolate the representation of uncertainty, we fit linear encoding models that span the power-Euclidean α-continuum from $\boldsymbol{\Sigma}_t \to \boldsymbol{\Lambda}_t$, showing that encoding quality is relatively flat in the expectation parameter range of α but drops steeply in the natural parameter range. Most importantly, we validate that these representations of mean and uncertainty are *causally*, rather than correlationally (Baker et al., 2025; Walker et al., 2023), related to network behavior; we also validate that these manipulations survive changes in context length (Figure 9).

[4] vech denotes half-vectorisation, i.e., vectorising the $M(M+1)/2$ unique elements in a symmetric matrix.

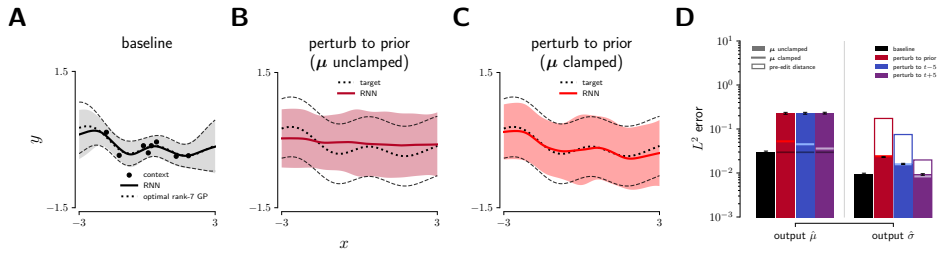


Figure 5: **Causal perturbations of the uncertainty encoding.** A) An example context rollout, similar to Figure 3A. B) Perturbation of $\mathcal{G}(\mathbf{\Sigma}_t; \alpha = 1/4)$ such that $\mathbf{\Sigma}_t \rightarrow$ prior changes output $\hat{\sigma}$ behavior appropriately but also perturbs the output $\hat{\mu}$. C) Clamping μ_t to its pre-edit state (using the encoder in Figure 3) fixes the output $\hat{\mu}$ while maintaining the perturbation to output $\hat{\sigma}$. D) Average L^2 error for each manipulation across many random trials (using a similar convention to Figure 3D). Baseline RNN error serves as a soft lower bound, and the total edit distance, as a soft upper bound. For the left half (output mean), we include perturbation performance when $\boldsymbol{\mu}$ is unclamped (filled bar) and clamped (colored horizontal line). Clamped edits maintain output mean accuracy; however, output standard deviation is almost identical for clamped versus unclamped edits.

Unlike previous work in probabilistic coding (Walker et al., 2020; Lambrechts et al., 2022; Hennig et al., 2023), our framework does not rely on decoding analyses, where fitted weights do not generally identify the representation generating neural activity (Haufe et al., 2014). Instead, we use encoding models (Haufe et al., 2014; Marks & Tegmark, 2024) to explain variability in neural activity. Importantly, our approach also includes causal manipulations, validating that the identified representations are behaviorally relevant. While causal manipulation has become a common practice in mechanistic interpretability (Li et al., 2024), it is absent in most prior work in neuroscience that studies probabilistic coding, both theoretically (Orhan & Ma, 2017) and experimentally (Walker et al., 2020). With the increasing popularity of causal perturbations via optogenetics (Deisseroth, 2011), we believe our methodology can be applied to closed-loop investigations of neural representations of uncertainty *in vivo*.

Our framework also refines existing theories in parametric neural coding (c.f. Haefner et al., 2024). In Bayesian inference on exponential family distributions, theoretical results often cluster parametric representations into natural (PPCs) and expectation (DDC) parameterisations (Haefner et al., 2024). Our approach instead embeds these as two extremes on a continuous spectrum parameterised by the power-Euclidean family of covariance matrices. Our approach provides a nuanced framework for evaluating different hypotheses (e.g., PPCs versus DDCs) in neural activity and suggests that networks may admit several approximately equivalent parameterisations of covariance.

These results are not inherently biologically plausible; we use GRU networks that are not constrained by Dale's law nor trained via local, biologically plausible learning rules. Our results instead serve to illustrate broader representational and computational hypotheses about how sequential filtering might be implemented in neural circuits and how to identify neural representations of uncertainty in practice. Accordingly, our approach focuses predominantly on characterising the *geometry* of the network's internal representation rather than its *dynamics*; we propose that future work could use tools such as dynamical similarity analysis (Ostrow et al., 2023), Koopman operator theory (Brunton et al., 2021), and nonautonomous systems theory (Kloeden & Rasmussen, 2011) to investigate dynamical motifs in these representations. Additionally, our analysis relies primarily on linear methods. This was motivated primarily by the desire to present a simple and accessible methodology, but it is nonetheless likely that analysing the nonlinear geometry of RNN activations may reveal a more holistic perspective on these belief representations Chung & Abbott (2021). Finally, our proposed inference task, Gaussian process regression, may appear divorced from canonical neuroscience tasks like changepoint detection (Tavoni et al., 2022) or reaching movements (Kaufman et al., 2014; Sussillo et al., 2015) and might not seem obviously ecologically relevant. However, given the exact equivalence between a class of GPs and state space models (Särkkä & Solin, 2019), studying inference tasks like GP regression can illustrate broader principles underlying how humans and other animals perform filtering and tracking tasks (Orban De Xivry et al., 2013; Kessler et al., 2024).

AI USE STATEMENT

In this work, we used generative AI tools for implementing methods used for analysis and providing feedback on research methodology. We have not used generative AI tools for interpreting results or supporting qualitative and thematic data analysis, and uses such as generating synthetic data sets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, assisting with translation, or cleaning and reformatting datasets are not applicable to this work. Additionally, we used generative AI tools for creating scientific figures and identifying relevant literature. We have reviewed all AI-assisted work. Core functions were written in equal parts by the authors and by LLMs, while figure-specific methodology used LLM-generated code with heavy supervision from the authors. Code has been verified for correctness by one author thus far. LLMs were not used to automatically generate citations or text in this manuscript. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Alessandro Achille and Stefano Soatto. Information Dropout: Learning Optimal Representations Through Noisy Computation, 2017.

Kwangjun Ahn, Xiang Cheng, Hadi Daneshmand, and Suvrit Sra. Transformers learn to implement preconditioned gradient descent for in-context learning. *Advances in Neural Information Processing Systems*, 2023.

Ludwig Arnold. *Random Dynamical Systems*. Springer Monographs in Mathematics. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

Ben Baker, Richard D Lange, Andrew Richmond, Nikolaus Kriegeskorte, Rosa Cao, and Odelia Schwartz. Use and usability: Concepts of representation in philosophy, neuroscience, cognitive science, and computer science. 2025.

Steven L. Brunton, Marko Budišić, Eurika Kaiser, and J. Nathan Kutz. Modern Koopman Theory for Dynamical Systems, 2021.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, 2014.

SueYeon Chung and L.F. Abbott. Neural population geometry: An approach for understanding biological and artificial neural networks. *Current Opinion in Neurobiology*, 70:137–144, 2021.

Karl Deisseroth. Optogenetics. *Nature Methods*, 8(1):26–29, 2011.

Ian L. Dryden, Alexey Koloydenko, and Diwei Zhou. Non-Euclidean statistics for covariance matrices, with applications to diffusion tensor imaging. *The Annals of Applied Statistics*, 3(3), 2009.

Ian L. Dryden, Xavier Pennec, and Jean-Marc Peyrat. Power Euclidean metrics for covariance matrices with application to diffusion tensor imaging, 2010.

Marc O. Ernst and Martin S. Banks. Humans integrate visual and haptic information in a statistically optimal fashion. *Nature*, 415(6870):429–433, 2002.

Marta Garnelo, Dan Rosenbaum, Chris J. Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo J. Rezende, and S. M. Ali Eslami. Conditional Neural Processes, 2018a.

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J. Rezende, S. M. Ali Eslami, and Yee Whye Teh. Neural Processes, 2018b.

Philip Greengard and Michael O'Neil. Efficient reduced-rank methods for Gaussian processes with eigenfunction expansions, 2022.

Ralf M. Haefner, Jeff Beck, Cristina Savin, Mehrdad Salmasi, and Xaq Pitkow. How does the brain compute with probabilities?, 2024.

Stefan Haufe, Frank Meinecke, Kai Görgen, Sven Dähne, John-Dylan Haynes, Benjamin Blankertz, and Felix Bießmann. On the interpretation of weight vectors of linear models in multivariate neuroimaging. *NeuroImage*, 87:96–110, 2014.

Jay A. Hennig, Sandra A. Romero Pinto, Takahiro Yamaguchi, Scott W. Linderman, Naoshige Uchida, and Samuel J. Gershman. Emergence of belief-like representations through reinforcement learning. *PLOS Computational Biology*, 19(9):e1011067, 2023.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second, 2023.

Ann Huang, Satpreet H. Singh, Flavio Martinelli, and Kanaka Rajan. Measuring and Controlling Solution Degeneracy across Task-Trained Recurrent Neural Networks, 2025.

Ishan Kalburge and Máté Lengyel. When sufficiency is insufficient: The functional information bottleneck for identifying probabilistic neural representations, 2025.

Gyeonghun Kang, Changwoo J. Lee, and Xiang Cheng. Transformers Can Learn Posterior Predictive Distributions In-Context, 2026.

Ingmar Kanitscheider and Ila Fiete. Training recurrent networks to generate hypotheses about how the brain solves hard navigation problems. *Advances in Neural Information Processing Systems*, 2017.

Matthew T Kaufman, Mark M Churchland, Stephen I Ryu, and Krishna V Shenoy. Cortical activity in the null space: Permitting preparation without movement. *Nature Neuroscience*, 17(3):440–448, 2014.

Fabian Kessler, Julia Frankenstein, and Constantin A. Rothkopf. Human navigation strategies and their errors result from dynamic interactions of spatial uncertainties. *Nature Communications*, 15 (1):5677, 2024.

Peter E. Kloeden and Martin Rasmussen. *Nonautonomous Dynamical Systems*. Number v. 176 in Mathematical Surveys and Monographs. American Mathematical Society, Providence, R.I, online-ausg edition, 2011.

Ádám Koblinger, József Fiser, and Máté Lengyel. Representations of uncertainty: Where art thou? *Current Opinion in Behavioral Sciences*, 38:150–162, 2021.

Konrad P. Körding and Daniel M. Wolpert. Bayesian integration in sensorimotor learning. *Nature*, 427(6971):244–247, 2004.

Nikolaus Kriegeskorte and Pamela K Douglas. Interpreting encoding and decoding models. *Current Opinion in Neurobiology*, 55:167–179, 2019.

Gaspard Lambrechts, Adrien Bolland, and Damien Ernst. Recurrent networks, hidden states and beliefs in partially observable environments. *Transactions on Machine Learning Research*, 2022.

Richard D. Lange, Sabyasachi Shivkumar, Ankani Chattoraj, and Ralf M. Haefner. Bayesian encoding and decoding as distinct perspectives on neural coding. *Nature Neuroscience*, 26(12): 2063–2072, 2023.

Kenneth Li, Aspen K. Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent World Representations: Exploring a Sequence Model Trained on a Synthetic Task, 2024.

Wei Ji Ma, Jeffrey M Beck, Peter E Latham, and Alexandre Pouget. Bayesian inference with probabilistic population codes. *Nature Neuroscience*, 9(11):1432–1438, 2006.

Niru Maheswaranathan, Alex Williams, Matthew D. Golub, Surya Ganguli, and David Sussillo. Reverse engineering recurrent networks for sentiment classification reveals line attractor dynamics, 2019a.

Niru Maheswaranathan, Alex H. Williams, Matthew D. Golub, Surya Ganguli, and David Sussillo. Universality and individuality in neural dynamics across large populations of recurrent networks, 2019b.

Samuel Marks and Max Tegmark. The Geometry of Truth: Emergent Linear Structure in Large Language Model Representations of True/False Datasets, 2024.

Vladimir Mikulik, Grégoire Delétang, Tom McGrath, Tim Genewein, Miljan Martic, Shane Legg, and Pedro A. Ortega. Meta-trained agents implement Bayes-optimal agents, 2020.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers Can Do Bayesian Inference. *Advances in Neural Information Processing Systems*, 2024.

Jean-Jacques Orban De Xivry, Sébastien Coppe, Gunnar Blohm, and Philippe Lefèvre. Kalman Filtering Naturally Accounts for Visually Guided and Predictive Smooth Pursuit Dynamics. *The Journal of Neuroscience*, 33(44):17301–17313, 2013.

A. Emin Orhan and Wei Ji Ma. Efficient probabilistic inference in generic neural networks trained with non-probabilistic feedback. *Nature Communications*, 8(1):138, 2017.

Pedro A. Ortega, Jane X. Wang, Mark Rowland, Tim Genewein, Zeb Kurth-Nelson, Razvan Pascanu, Nicolas Heess, Joel Veness, Alex Pritzel, Pablo Sprechmann, Siddhant M. Jayakumar, Tom McGrath, Kevin Miller, Mohammad Azar, Ian Osband, Neil Rabinowitz, András György, Silvia Chiappa, Simon Osindero, Yee Whye Teh, Hado van Hasselt, Nando de Freitas, Matthew Botvinick, and Shane Legg. Meta-learning of Sequential Strategies, 2019.

Mitchell Ostrow, Adam Eisen, Leo Kozachkov, and Ila Fiete. Beyond Geometry: Comparing the Temporal Structure of Computation in Neural Circuits with Dynamical Similarity Analysis, 2023.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, Mass., 3. print edition, 2008.

Simo Särkkä and Arno Solin. *Applied Stochastic Differential Equations*. Cambridge University Press, 1 edition, 2019.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks, 2014.

Arno Solin and Simo Särkkä. Hilbert Space Methods for Reduced-Rank Gaussian Process Regression. *Statistics and Computing*, 30(2):419–446, 2020.

David Sussillo and Omri Barak. Opening the Black Box: Low-Dimensional Dynamics in High-Dimensional Recurrent Neural Networks. *Neural Computation*, 25(3):626–649, 2013.

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

Gaia Tavoni, Takahiro Doi, Chris Pizzica, Vijay Balasubramanian, and Joshua I. Gold. Human inference reflects a normative balance of complexity and accuracy. *Nature Human Behavior*, 6: 1153–1168, 2022.

D. J. Tolhurst, J. A. Movshon, and A. F. Dean. The statistical reliability of signals in single neurons in cat and monkey visual cortex. *Vision Research*, 21(8):475–485, 1983.

Eszter Vértes and Maneesh Sahani. Flexible and accurate inference and learning for deep generative models. *Advances in Neural Information Processing Systems*, 2018.

Edgar Y. Walker, R. James Cotton, Wei Ji Ma, and Andreas S. Tolias. A neural basis of probabilistic computation in visual cortex. *Nature Neuroscience*, 23(1):122–129, 2020.

Edgar Y. Walker, Stephan Pohl, Rachel N. Denison, David L. Barack, Jennifer Lee, Ned Block, Wei Ji Ma, and Florent Meyniel. Studying the neural representations of uncertainty. *Nature Neuroscience*, 26(11):1857–1867, 2023.

Yedi Zhang, Aaditya K Singh, Peter E Latham, and Andrew Saxe. Training Dynamics of In-Context Learning in Linear Attention. *International Conference on Machine Learning*, 2025.

APPENDIX

A ADDITIONAL TECHNICAL DETAILS

A.1 BELIEF FILTERING UPDATE DYNAMICS

Suppose, instead of operating in function space, we consider the weight space Bayesian regression problem

$$\mathbf{y} = \mathbf{\Phi}\mathbf{w} + \boldsymbol{\epsilon}, \qquad \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_w), \;\; \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma_n^2 \mathbf{I}),$$

where $\mathbf{\Phi}$ is a design matrix that is lifted into a kernel space with basis functions $\phi_i(\cdot)$. It is easy to see how this is exactly a Gaussian process, namely that

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}; \mathbf{0}, \underbrace{\mathbf{\Phi}\mathbf{\Sigma}_w\mathbf{\Phi}^\top}_{\mathbf{K}} + \sigma_n^2 \mathbf{I}_N),$$

where $\mathbf{K}$ is the kernel matrix. Thus, any Bayesian regression problem has a kernel equal to $\mathbf{\Phi}\mathbf{\Sigma}_w\mathbf{\Phi}^\top$, and the GP regression problem simplifies to simply tracking the posterior over weights $p(\mathbf{w}|\mathbf{X}, \mathbf{y})$:

$$\begin{aligned}
p(\mathbf{w}|\mathbf{X}, \mathbf{y}) &\propto p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w}) \\
&= \mathcal{N}(\mathbf{\Phi}\mathbf{w}, \sigma_n^2 \mathbf{I}_N)\mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_w) \\
&\propto \exp\left(-\frac{1}{2}\left(-\frac{2}{\sigma_n^2}\mathbf{w}^\top\mathbf{\Phi}^\top\mathbf{y} + \mathbf{w}^\top\left(\frac{1}{\sigma_n^2}\mathbf{\Phi}^\top\mathbf{\Phi} + \mathbf{\Sigma}_w^{-1}\right)\mathbf{w}\right)\right) \\
&= \mathcal{N}(\boldsymbol{\mu} = \sigma_n^{-2}\mathbf{\Lambda}^{-1}\mathbf{\Phi}^\top\mathbf{y}, \mathbf{\Sigma} = \mathbf{\Lambda}^{-1}) \\
\mathbf{\Lambda} &= \sigma_n^{-2}\mathbf{\Phi}^\top\mathbf{\Phi} + \mathbf{\Sigma}_w^{-1}.
\end{aligned}$$

Because $\boldsymbol{\eta} = \mathbf{\Lambda}\boldsymbol{\mu} = \frac{1}{\sigma_n^2}\mathbf{\Phi}^\top\mathbf{y} = \frac{1}{\sigma_n^2}\sum_i \boldsymbol{\phi}(\mathbf{x}_i)y_i$, and $\mathbf{\Phi}^\top\mathbf{\Phi} = \sum_{i=1}^N \phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^\top$, the information-precision (natural) parameter updates can be written simply as two decoupled update rules:

$$\boldsymbol{\eta}_t = \boldsymbol{\eta}_{t-1} + \tfrac{1}{\sigma_n^2}\boldsymbol{\phi}_t y_t \tag{1}$$

$$\mathbf{\Lambda}_t = \mathbf{\Lambda}_{t-1} + \tfrac{1}{\sigma_n^2}\boldsymbol{\phi}_t\boldsymbol{\phi}_t^\top, \quad \boldsymbol{\phi}_t = \phi(\mathbf{x}_t), \tag{2}$$

where ϕ now refers to the vector $\{\boldsymbol{\phi}_t\}_j = \phi_j(\mathbf{x}_t)$. Note that we have switched from indexing data using i to t to suggest that this recursion can be done over time, i.e., using an RNN.

The the mean-covariance (closely related to the expectation) update equations are more complicated and, crucially, coupled:

$$\mathbf{k}_t = \frac{\mathbf{\Sigma}_{t-1}\boldsymbol{\phi}_t}{\boldsymbol{\phi}_t^\top\mathbf{\Sigma}_{t-1}\boldsymbol{\phi}_t + \sigma_n^2} \qquad \text{(Kalman gain)} \tag{3}$$

$$\boldsymbol{\mu}_t = \boldsymbol{\mu}_{t-1} + \mathbf{k}_t(y_t - \boldsymbol{\phi}_t^\top\boldsymbol{\mu}_{t-1}) \tag{4}$$

$$\mathbf{\Sigma}_t = \mathbf{\Sigma}_{t-1} - \mathbf{k}_t(\phi_t^\top\mathbf{\Sigma}_{t-1}) \tag{5}$$

A.2 REWRITING UPDATE EQUATIONS AS LDSS

Mean-covariance versus information-precision parameter update rules prescribe surprisingly different dynamics. For the optimal Bayesian filter, the natural parameter updates form a system of uncoupled dynamical systems:

$$\begin{aligned}
\boldsymbol{\eta}_t &= \mathbf{A}\boldsymbol{\eta}_{t-1} + \mathbf{b}_t y_t, \quad \mathbf{A} = \mathbf{I}, \;\; \mathbf{b}_t = \tfrac{1}{\sigma_n^2}\boldsymbol{\phi}_t \\
\mathbf{\Lambda}_t &= \mathbf{C}\mathbf{\Lambda}_{t-1} + \mathbf{D}_t, \quad \mathbf{C} = \mathbf{I}, \;\; \mathbf{D}_t = \tfrac{1}{\sigma_n^2}\boldsymbol{\phi}_t\boldsymbol{\phi}_t^\top
\end{aligned}$$

In the mean-covariance case, the update rule is coupled, but still a LDS:

$$\begin{aligned}
\boldsymbol{\mu}_t &= \mathbf{H}_t\boldsymbol{\mu}_{t-1} + \mathbf{k}_t y_t, \quad \mathbf{H}_t = \mathbf{I} - \mathbf{k}_t\boldsymbol{\phi}_t^\top \\
\mathbf{\Sigma}_t &= \mathbf{H}_t\mathbf{\Sigma}_{t-1}
\end{aligned}$$

A.3 POWER-EUCLIDEAN FAMILY

The power-Euclidean family of positive definite matrices contains all transformations

$$\mathcal{G}(\mathbf{\Sigma}; \alpha) = \frac{\mathbf{\Sigma}^\alpha - \mathbf{I}}{\alpha},$$

which is well-defined for positive definite matrices like covariances. It is the matrix equivalent of the Box-Cox transformation (Dryden et al., 2010). In the limiting case $\lim_{\alpha \to 0} \mathcal{G}(\mathbf{\Sigma}, \alpha) = \log \mathbf{\Sigma}$.

Take the scalar (original Box-Cox) case:

$$\begin{aligned}
\lim_{\alpha \to 0} g(y; \alpha) &= \lim_{\alpha \to 0} \frac{y^\alpha - 1}{\alpha} \\
&= \lim_{\alpha \to 0} \frac{y^\alpha \log y}{1} \qquad \text{(L'Hopital's rule)} \\
&= \log y.
\end{aligned}$$

An identical construction can be made for the matrix case; eigendecomposition $\mathbf{\Sigma} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$ produces the diagonal $\mathbf{\Lambda}$, upon which each eigenvalue can be Box-Cox transformed:

$$\begin{aligned}
\lim_{\alpha \to 0} \mathcal{G}(\mathbf{\Sigma}; \alpha) &= \lim_{\alpha \to 0} \mathbf{V} \mathrm{diag}\left(\frac{\lambda_i^\alpha - 1}{\alpha}\right) \mathbf{V}^\top \\
&= \mathbf{V} \mathrm{diag}(\log \lambda_i) \mathbf{V}^\top \\
&= \log \mathbf{\Sigma}
\end{aligned}$$

A.4 GRU ARCHITECTURE

The results and figures presented in this manuscript analyse a task-trained gated recurrent unit (GRU) RNN (Cho et al., 2014). Unlike a vanilla RNN, which has a hidden state $\mathbf{h}_t$ and inputs $\mathbf{x}_t$, a gated recurrent unit (GRU) RNN has two update gates that augment the hidden state:

$$\begin{aligned}
\mathbf{r}_t &= \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_t + \mathbf{b}_r), && \text{(reset gate)} \\
\mathbf{z}_t &= \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_t + \mathbf{b}_r), && \text{(update gate)} \\
\bar{\mathbf{h}}_t &= \tanh(\mathbf{W}_h \mathbf{x}_t + \mathbf{U}_h (\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_h) && \text{(candidate state)} \\
\mathbf{h}_t &= (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \bar{\mathbf{h}}_t \\
&= \mathbf{h}_{t-1} + \mathbf{z}_t \odot (\bar{\mathbf{h}}_t - \mathbf{h}_{t-1}).
\end{aligned}$$

A.5 TRAINING

Networks were trained in `jax` using Adam optimizer, a learning rate of 0.001, a batch size of 16, and a sufficient number of epochs (200,000) to ensure convergence in negative log-likelihood loss. Networks examined in the main text had 128 hidden neurons (denoted H), and training data for those networks were generated with fixed context size $C = 100$ (i.e., the RNN saw 100 (x_i, y_i) pairs per trial) and GP lengthscale $\ell = 1$. however, we further investigate. Context size in the training data was fixed to 100 (i.e., 100 (x_i, y_i) pairs).

B ADDITIONAL RESULTS

B.1 APPROXIMATE PERMUTATION INVARIANCE

Unlike in previous approaches to amortising GP inference (Garnelo et al., 2018a;b; Müller et al., 2024), we use an architecture—a standard gated recurrent network—that is not naturally permutation invariant. Thus, we validated the network's approximate learned permutation invariance empirically. After training, networks are approximately behaviorally permutation invariant (Figure 6A), and this invariance—measured as the output squared error between permutations normalised by the error between to distinct context sets—generalises to out-of-distribution context lengths (Figure 6B). Importantly, this permutation invariance is progressively learned over training (Figure 6C). We also validate that this permutation invariance is present not just in behavior but also in hidden state space. Indeed, different orderings of the same context set are consistently closer in normalised Euclidean distance than altogether different context sets (Figure 6D).

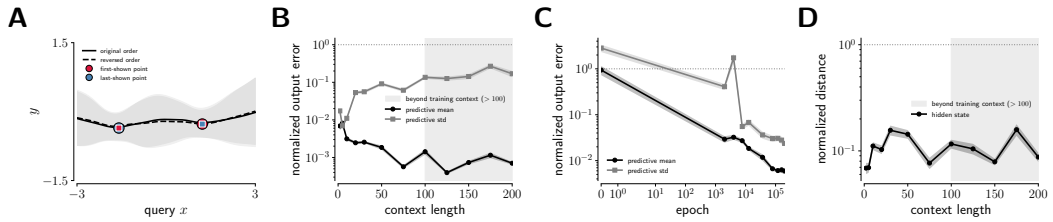


Figure 6: **Approximate, learned permutation invariance**. A) An example of RNN output on a 2-context trial where the ordering of context points is flipped. Behavioral output is approximately the same. B) Normalised (see complementary text) output error is relatively stable across context lengths. C) Permutation sensitivity decreases monotonically over training. D) Permutation invariance is reflect not just in output error but also in hidden state distance: permutations of the same context set are closer than different context sets.

B.2 Reproduction across random seeds

We selected a few key analyses, namely behavioral (Figure 1) and representational (Figure 2) rank estimate protocols, and reported their results across multiple random seeds (Table 1). Representational and behavioral rank estimates agree on average but differ somewhat from seed to seed.

	representational	behavioural	
seed	M_{eff} (Fig. 2A)	M, mean error (Fig. 1G)	M, kernel spectrum (Fig. 1H)
0	8	7	7
1	7	7	7
2	8	7	7
3	7	7	7
4	7	7	6
5	7	7	7
6	8	7	7
7	8	7	7
8	8	7	7
9	8	7	7
mean $\pm$ sd	7.6 ± 0.5	7.0 ± 0.0	6.9 ± 0.3
Figs. 1–2 run	7	7	7

Table 1: Rank estimation procedures from Figure 1 and Figure 2 reproduced across multiple random seeds.

B.3 Alternative network models

We similarly explore several model variants to see if our results generalise to other networks. Specifically, we test different recurrent cores—LSTMs (Hochreiter & Schmidhuber, 1997) and vanilla RNNs—as well as a different loss function (MSE). We also test several architectural ablations to the GRU, including 1) removing the chained (sequential $x_t \to y_t \to x_{t+1} \to ...$) tokenisation and instead feeding in x_t, y_{t-1} together ("concat input"), 2) retaining the chained architecture but removing the appended tokens $0, 1$ for update versus query steps ("no type flag"), 3) randomly sampling context length C on each batch from $\mathcal{U}(50, 100)$ and $\mathcal{U}(5, 150)$ instead of fixing $C = 100$. The GRU trained with MSE loss approximately reproduces several of the baseline GRU and LSTM behavior. However, the vanilla RNN seems to learn a much lower-rank approximation. Most notably, across all architectures besides the vanilla RNN, $\alpha^\star$ seems to favor expectation codes (i.e., $\alpha^\star$ is never very negative).

		representational		behavioural	
model	seeds	M_{eff} (Fig. 2A)	α^* (Fig. 3A)	M, mean error (Fig. 1G)	M, kernel spectrum (Fig. 1H)
GRU (baseline)	10	7.6 ± 0.2	0.24 ± 0.07	7.0 ± 0.0	6.9 ± 0.1
LSTM	5	7.4 ± 0.2	0.07 ± 0.04	7.0 ± 0.0	7.0 ± 0.0
vanilla RNN	5	5.6 ± 0.2	-1.00 ± 0.00	4.0 ± 0.0	2.4 ± 0.2
GRU, MSE loss	5	7.8 ± 0.2	0.12 ± 0.06	7.0 ± 0.0	5.6 ± 0.6
GRU, concat input†	5	7.6 ± 0.2	-0.14 ± 0.02	6.8 ± 0.2	2.6 ± 0.2
GRU, no type flag	5	7.6 ± 0.2	0.17 ± 0.04	7.0 ± 0.0	6.6 ± 0.4
GRU, var. ctx 50–100	5	7.0 ± 0.0	0.28 ± 0.07	7.0 ± 0.0	7.0 ± 0.0
GRU, var. ctx 50–150	5	7.0 ± 0.0	0.24 ± 0.06	7.0 ± 0.0	7.0 ± 0.3

mean $\pm$ SEM over seeds. †for M_{eff}, h^* excludes the last of the 200 context observations; for α^*, Σ_t is paired with the state that has absorbed y_t (see STab1.py).

Figure 7: Selected results reproduced on other model variants.

B.4 Exploration of other hyperparameters

Finally, we present loss curves for different settings of the hyperparameters of our experiments: hidden layer size H, GP lengthscale ℓ, and context set length C. As expected, decreasing all three hyperparameters monotonically reduced network performance (measured in fractional NLL).

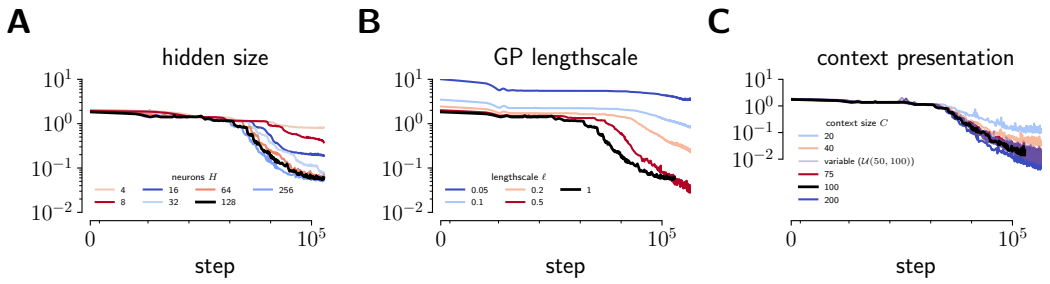


Figure 8: **Hyperparameter exploration**. A) Decreasing the number of hidden layer neurons monotonically decreases performance and is suggestive of progressively smaller-rank representations. B) Similarly, decreasing lengthscale (which gives nonzero weight to more eigenfunctions also decreases network performance monotonically. C) Presenting shorter context rollouts also decreases performance.

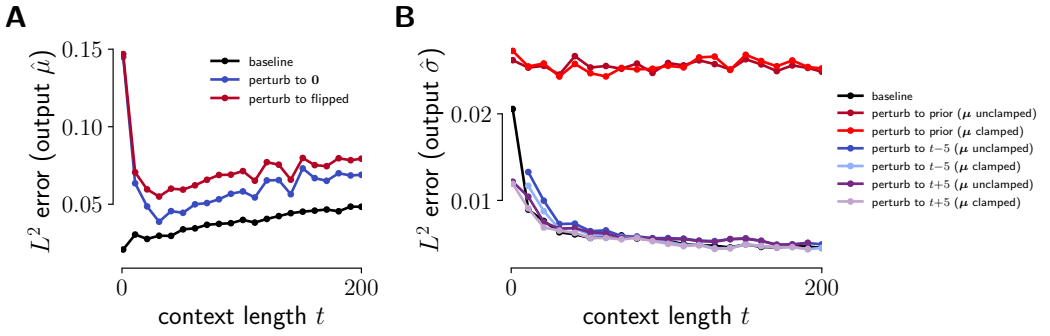


Figure 9: **Causal manipulations over varying context length**. Causal manipulations to belief state mean (A) and covariance (B) survive changes in context length. Mean manipulations drift slightly outside the context range used to train the encoder ($C = 100$).

$\boldsymbol{\Sigma}$ parameterisation	encode R^2 ($[\boldsymbol{\mu}, \cdot] \to \mathbf{h}$)
$\boldsymbol{\Sigma}$	0.8317
$\boldsymbol{\Sigma}^{1/2}$	0.8335
$\boldsymbol{\Sigma}^{1/4}$	**0.8341**
$\log \boldsymbol{\Sigma}$	0.8322
$\boldsymbol{\Sigma}^{-1/2}$	0.8095
$\boldsymbol{\Sigma}^{-1}$	0.7111
none ($\boldsymbol{\mu}$ only)	0.3479

Fully trained network; held-out, context length 50.

Table 2: Held-out encoding R^2 values for different α-parameterisations of $\boldsymbol{\Sigma}_t$ reveals a shallow optimum at $\alpha^\star = 1/4$. Encoding quality drops when $\boldsymbol{\Sigma}$ is dropped altogether.

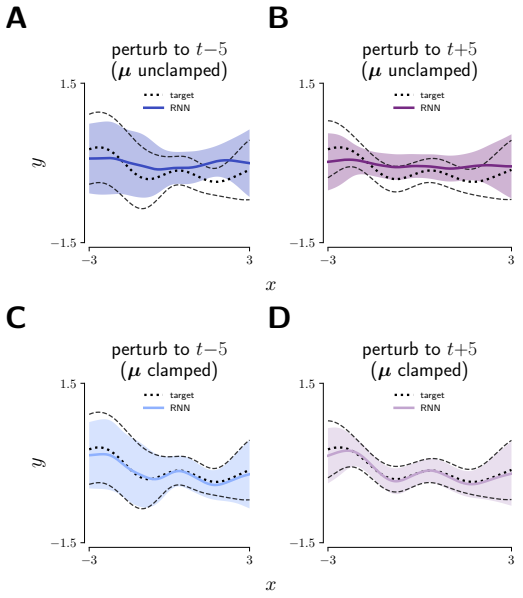


Figure 10: **Additional perturbation visualisations**. We test reversion to $t-5$ (A, C) and rollout to $t+5$ (B, D) perturbations of covariance $\mathcal{G}(\boldsymbol{\Sigma}_t; 1/4)$. These perturbations also show that unclamped (A, B) perturbations affect output mean, which can be fixed by clamping $\boldsymbol{\mu}_t$ (C, D).