Mechanistic Interpretability of RNNs emulating Hidden Markov Models

Elia Torre **Michele Viscione** **Lucas Pompe** **Benjamin F. Grewe** **Valerio Mante**

Institute of Neuroinformatics, University of Zurich & ETH Zurich

Abstract

Recurrent neural networks (RNNs) provide a powerful approach in neuroscience to infer latent dynamics in neural populations and to generate hypotheses about the neural computations underlying behavior. However, past work has focused on relatively simple, input-driven, and largely deterministic behaviors - little is known about the mechanisms that would allow RNNs to generate the richer, spontaneous, and potentially stochastic behaviors observed in natural settings. Modeling with Hidden Markov Models (HMMs) has revealed a segmentation of natural behaviors into discrete latent states with stochastic transitions between them, a type of dynamics that may appear at odds with the continuous state spaces implemented by RNNs. Here we first show that RNNs can replicate HMM emission statistics and then reverse-engineer the trained networks to uncover the mechanisms they implement. In the absence of inputs, the activity of trained RNNs collapses towards a single fixed point. When driven by stochastic input, trajectories instead exhibit noise-sustained dynamics along closed orbits. Rotation along these orbits modulates the emission probabilities and is governed by transitions between regions of slow, noise-driven dynamics connected by fast, deterministic transitions. The trained RNNs develop highly structured connectivity, with a small set of "kick neurons" initiating transitions between these regions. This mechanism emerges during training as the network shifts into a regime of stochastic resonance, enabling it to perform probabilistic computations. Analyses across multiple HMM architectures — fully connected, cyclic, and linear-chain — reveal that this solution generalizes through the modular reuse of the same dynamical motif, suggesting a compositional principle by which RNNs can emulate complex discrete latent dynamics. Code available at `https://github.com/EliaTorre/hmmrnn`.

1 Introduction

Modern large-scale electrophysiology and imaging techniques provide access to the simultaneous activity of thousands of neurons in freely behaving animals, revealing the population-level dynamics underlying perception, cognition, and movement [39, 18, 4, 3, 19, 40]. In parallel to advances in neural recordings, it has become possible to obtain quantitative descriptions of unconstrained, natural behaviors across many timescales [45]. The combination of these techniques holds great promise in unlocking the neural computations underlying behavior [7].

Machine learning is critical to interpret and link the high-dimensional neural and behavioral data produced by modern neuroscience experiments. Hidden Markov Models (HMMs) can capture unconstrained, natural behaviors and decompose them into sequences of elemental motifs [45]. However, their discrete state representation may oversimplify the potentially continuous nature of neural processes. On the other hand, Recurrent Neural Networks (RNNs) provide a powerful approach to model the dynamics of large neural populations and to generate hypotheses about the underlying computations [10, 47] . Despite this potential, RNNs are primarily used to model highly constrained

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

tasks that are input-driven and largely deterministic. Hence, evidence for RNNs applicability in modelling stochastic discrete state transitions in a manner similar to HMMs remains limited.

RNNs and HMMs may appear incompatible, as RNNs represent latent processes as trajectories through continuous state spaces, whereas HMMs rely on discrete latent states with stochastic transitions between them. Understanding whether RNNs can use continuous dynamics to generate stochastic transitions between discrete states would help bridge this conceptual gap and reduce the need for strong assumptions about the structure of the latent space. To address this question, we develop a training approach to directly fit RNNs to HMMs, and then reverse-engineer the trained RNNs to uncover the computations they implement. This strategy may ultimately lead to testable hypotheses about how biological neural circuits may implement discrete behavioral modes through continuous internal dynamics.

This work makes three main novel contributions:

1. A training paradigm for stochastic RNN behaviors (Sections 3, Appendix B, Figure 1): unlike previous work focused on deterministic tasks, we introduce a training method combining noise-driven RNNs with Sinkhorn divergence optimization, enabling learning of the stochastic state transitions typical of HMMs.
2. A demonstration that RNNs can emulate HMM statistics (Sections 3, Appendix C, Figures 10, 11): we show that vanilla RNNs accurately replicate the emission statistics of various HMM architectures, matching both transition dynamics and stationary distributions.
3. A multi-level mechanistic account of how RNNs generate discrete stochastic outputs.
   - 3.1 Global dynamics (Section 4.1, Figures 3, 4): We reveal how training drives the RNNs into a regime where stochastic inputs sustain dynamics along closed orbits ("orbital dynamics"), while in their absence, the activity converges to a single fixed point.
   - 3.2 Local dynamics (Section 4.2, Figure 5): We identify three functional zones: clusters, kick-zones, and transitions, each with distinct dynamical signatures.
   - 3.3 Connectivity and single neurons (Section 5, Figures 6, 7): We uncover structured connectivity between "noise-integrating populations" and "kick neurons" and validate their role in triggering state transitions through causal interventions.
   - 3.4 Computational principle (Section 6, Figure 8): We show the inferred mechanism relies on self-induced stochastic resonance.

Our findings form a coherent picture: local dynamical features, such as clusters and kick-zones, explain how global orbital dynamics give rise to discrete outputs, while the network's connectivity reveals how individual neurons instantiate these dynamics. Together, these findings show that RNNs discover a general computational motif that combines slow noise integration with fast transitions.

The mechanism we uncover offers a novel interpretation of how RNNs can generate stochastic transitions between discrete states. This mechanism (Contributions 3.1 - 3.4) acts as a "dynamical primitive", reusable in a modular fashion to emulate more complex discrete latent structures. Multiple instances of this motif combine to govern transitions between specific pairs of states.

Our results lay the foundation for future work, both in scaling to more complex HMM structures and in investigating whether similar dynamical motifs are employed by biological neural circuits during naturalistic behavior.

2 Related Work

RNNs have been trained on simple perceptual, cognitive, and movement tasks, both with supervised learning [30, 42, 44] and reinforcement learning [43, 46]. These RNNs are often trained with few or no constraints and then reverse-engineered to reveal potentially novel hypotheses about the computations performed by biological neural circuits. Fully understanding the dynamics of RNNs and characterizing the mechanisms they implement (i.e., mechanistic interpretability) remains a challenging and open problem [37, 36].

A common reverse-engineering approach focuses on characterizing the topology of fixed points and on obtaining linear approximations of the dynamics around them [41]. For example, RNNs trained on motion discrimination [30] or on sentiment-classification [28] integrate inputs along line-attractors.

Networks trained to reproduce reach movements implement rotational dynamics similar to that in primate motor cortex [42] and RNNs performing beat continuation generate low-dimensional oscillatory dynamics [47]. Across many tasks, the topology of fixed points is invariant to the details of the RNN implementation (e.g. vanilla RNN, GRU, LSTM), suggesting a "universality" in the types of solutions implemented by RNNs [29].

Recent work has pushed mechanistic interpretability beyond population dynamics, to the underlying RNN connectivity. Constraining RNNs weights to being low-rank can reveal the relation between connectivity, computations, and dynamics [31]. Such relations may extend to RNNs trained without constraints, as their function may rely on a low-rank component of the full connectivity [24, 20]. Highly structured weights may emerge in particular when connectivity and latent dynamics are trained concurrently while imposing biological constraints [25] or when training on many different tasks at once [22, 9, 35].

Building on this line of work, here we investigate whether the above insights can be extended to RNNs implementing a different type of computation, namely internally-driven, probabilistic transitions between discrete latent states.

3 Approximating HMMs with RNNs

To understand how RNNs encode discrete, probabilistic structure in their state spaces, we trained them to replicate the outputs of HMMs, which implement a process that is discrete and probabilistic by construction (Fig. 1).

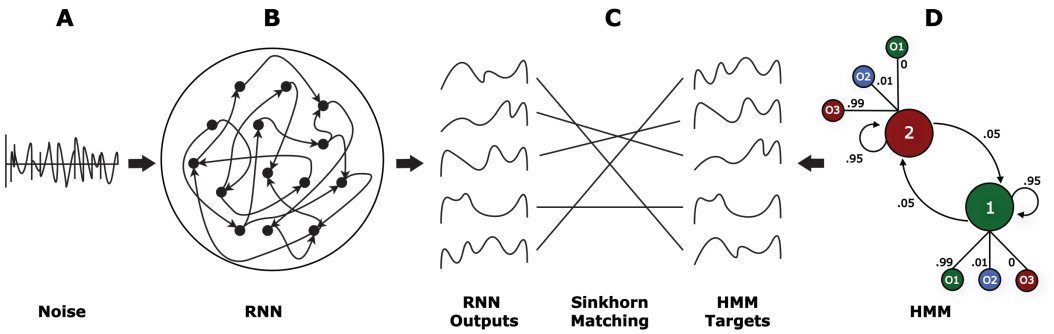

Figure 1: **Training pipeline. A**: At each time-step the network receives an i.i.d. Gaussian input vector $x_t \sim \mathcal{N}(0, I_d)$, where the input dimensionality is varied across experiments ($d \in \{1, 10, 100, 200\}$). **B**: The signal is processed by a vanilla recurrent neural network, $h_t = \text{ReLU}\big(h_{t-1}W_{hh}^{\mathsf{T}} + x_t W_{ih}^{\mathsf{T}}\big)$; the hidden-state dimension is also varied ($|h| \in \{50, 150, 200\}$). Finally, the hidden state is mapped linearly to a three-dimensional output via $y_t = h_t A^{\mathsf{T}}$. To mimic HMM emissions, logits are converted to categorical samples via *Gumbel–Softmax*. **C**: Parameters $\Theta = \{W_{hh}, W_{ih}, A\}$ are optimised by minimising the *Sinkhorn divergence* between batches of predicted outputs $Y = \{y_t\}_{t=1}^{T}$ and target sequences $Y^\star = \{y_s^\star\}_{s=1}^{T}$. **D**: Target sequences $Y^\star$ are generated by *linear-chain*, *fully-connected*, and *cyclic* HMMs (described in Section 3.1 and Appendix A); the example shows a 2-state *linear-chain* model.

3.1 Families of HMM architectures

To investigate how RNNs encode discrete, probabilistic structure across different latent state topologies, we train them to replicate the outputs of HMMs with three distinct types of architectures: *linear-chain*, *fully-connected* and *cyclic* structures.

Linear–chain HMMs. We first consider a family of *linear-chain* HMMs that span a spectrum from maximally discrete to quasi-continuous representations. These models have $M \in \{2, 3, 4, 5\}$ latent states $\mathcal{S}^{(M)} = \{1, \ldots, M\}$ and a common alphabet of observations $\mathcal{O} = \{1, 2, 3\}$. Each model $\mathcal{H}_M$ is fully specified by an emission matrix $\mathbf{E}^{(M)} = [e_i^{(M)}]_{i=1}^{M}$ and a band-diagonal transition matrix $\mathbf{T}^{(M)} = [t_{ij}^{(M)}]_{i,j=1}^{M}$.

For each latent state i, we obtain emission probabilities as linear interpolation between a state skewed toward observation 1 ($i = 1$) and one skewed toward observation 3 ($i = M$), while keeping $P(o_t = 1) = \varepsilon$ constant. At each timestep, the chain either remains in the current state or transitions to a neighboring state. Given a change-rate hyper-parameter $\rho = 0.05$, we set $q_M = \rho^{1/(M-1)} \in (0, \frac{1}{2})$, $\alpha_i = (i-1)/(M-1)$, fix $\varepsilon = 0.01$ and define:

$$e_i^{(M)} = \begin{bmatrix} (1-\varepsilon)(1-\alpha_i) \\ \varepsilon \\ (1-\varepsilon)\,\alpha_i \end{bmatrix}, \quad t_{ij}^{(M)} = \begin{cases} 1-q_M, & i \in \{1, M\},\ j = i, \\ 1-2q_M, & 1 < i < M,\ j = i, \\ q_M, & j = i \pm 1, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

This parametrization maintains a constant probability ρ of reaching the most distant state in $M-1$ steps. When $M = 2$ the system is maximally discrete; as M increases, the same $0 \to 2$ continuum is partitioned into progressively finer bins, yielding a quasi–continuous latent representation. This design systematically varies discreteness to reveal how RNN dynamics adapt as the latent space becomes more continuous.

Fully-connected HMMs. To assess generalization beyond simple structures, we use a 3-state fully connected HMM where each state can transition to any other. Each state favors one of the three outputs but retains small probabilities for the others, testing RNNs on fully interconnected dynamics.

Cyclic HMMs. We also consider a 4-state cyclic HMM with bidirectional closed-loop transitions. For each state, emissions are biased toward a characteristic pair of outputs, one dominant and one weaker component, with adjacent states sharing a common output.

The complete specification of all HMM architectures, including transition and emission matrices, is provided in Appendix A (Fig. 9).

3.2 Training Networks with Sinkhorn Loss Function and Performance Metrics

Network Architecture. We employ standard, *vanilla* RNNs [12] of hidden-state size $|h| \in \{50, 150, 200\}$. At each time-step, the network receives Gaussian input $x_t \sim \mathcal{N}(0, I_d)$, with $d \in \{1, 10, 100, 200\}$. The hidden state is updated and projects onto the three logits via:

$$h_t = \text{ReLU}(h_{t-1} W_{hh}^{\mathsf{T}} + x_t W_{ih}^{\mathsf{T}}), \quad y_t = h_t A^{\mathsf{T}}. \tag{2}$$

To mimic the discrete emissions of an HMM, these logits are converted to categorical samples using the *Gumbel-Softmax reparametrization trick*: we add i.i.d. Gumbel noise, divide by a temperature τ (set to 1 in all experiments), and apply a soft-max. This continuous relaxation acts as a differentiable proxy for the non-differentiable *argmax* in the Gumbel-Max method, letting gradients flow through the sampling step while still converging to one-hot vectors as $\tau \to 0$ [21] [27].

Sinkhorn Divergence. Because our target sequences are probabilistic, we use a loss function that is appropriate for comparing distributions: the *Sinkhorn divergence* [15][16][38], an optimal transport (OT) distance. Typically, OT-distances find a binary coupling matrix Π linking single samples in the outputs and targets, which minimizes the total Euclidean distance between coupled samples. If the euclidean distance between coupled samples is zero, the output distribution reproduces the target distribution. Finding such a coupling matrix is computationally expensive and non-differentiable. The Sinkhorn divergence overcomes these issues by finding a smoothed coupling matrix allowing non-unique couplings [5]. Details on the training regime are provided in Appendix B.

Performance Metrics. To quantify how closely the trained RNNs replicate their target HMMs, we track four statistics: **(i)** Euclidean distance between "Sinkhorn-aligned" output sequences (global reconstruction error); **(ii)** the 3×3 transition matrix of successive emissions (long-range dynamics); **(iii)** marginal observation frequencies (stationary distribution); and **(iv)** observation volatility (proportion of time steps with output changes). For all metrics, the trained RNNs replicate the emission statistics of the target HMMs. Definitions and results for all combinations of hidden size ($|h| \in 50, 150, 200$) and input dimensionality ($d \in 1, 10, 100, 200$) are shown in Appendix C (Figs. 10, 11). Higher input dimensionality improves convergence, while increasing hidden size beyond 150 yields only marginal benefits. Below, we thus focus on the four configurations with $|h| = 150$ and $d = 100$.

4 Mechanistic Interpretability: Latent Dynamics

To uncover the mechanisms implemented by the trained RNNs, we first consider their global (Section 4.1) and local (4.2) latent dynamics. These analyses reveal how several distinct "dynamical motifs" together contribute to the RNNs' ability to approximate HMM-like emission statistics (Fig. 2).

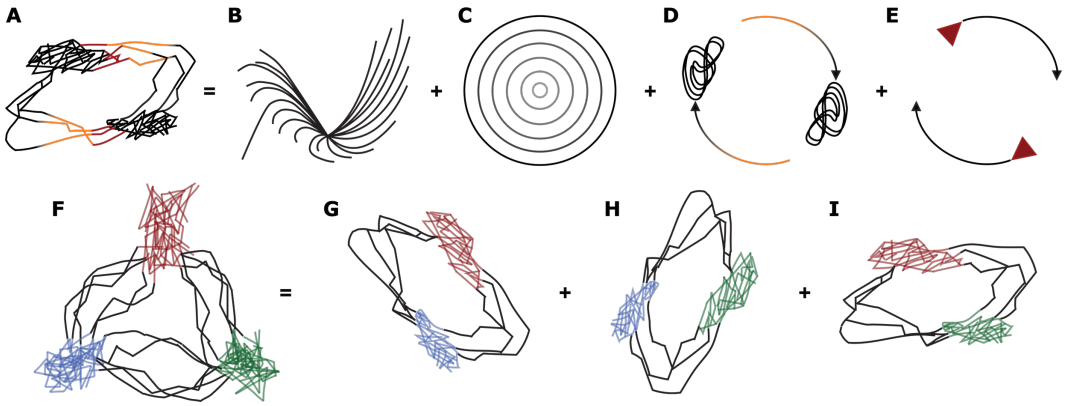

Figure 2: **A compositional dynamical primitive**. Schematic of the dynamical motifs uncovered through reverse-engineering of trained RNNs. **A**: Sample trajectory in the continuous state space. **B**: A single fixed point in the absence of input. **C**: Stochastic input displaces the system from this attractor, sustaining dynamics along closed orbits whose radius grows with input variance. **D**: Distinct slow regions (*clusters*) where trajectories linger, separated by *transition* regions. **E**: Local "*kick-zones*" that trigger transitions between clusters. **F-I**: Schematic of a richer discrete latent structure (**F**) expressed as a composition of three instances of the same dynamical primitive (**G-I**). Each instance preserves the same global and local dynamical properties described in panels (**A-E**).

4.1 Global Latent Dynamics: Noise-sustained Orbital Dynamics

Orbital dynamics under stochastic input. We investigate the global latent dynamics of the trained RNNs by projecting their hidden states onto the first two principal components (PCA, Figure 3). When initialized from random hidden states, and without external input, activity converged to a single fixed point (Figs. 2B, 3B), with no evidence for separate attractors that might be expected to represent the discrete latent states of the target HMMs. However, the latent dynamic changes markedly for RNNs receiving stochastic (Gaussian) inputs, to a regime where trajectories exhibit orbital dynamics (Figs. 2A, 3C). The stochastic input pushes activity outward (Figs. 2C, 3D), while the recurrent component pulls it back, together implementing a stable closed orbit along which activity evolves uni-directionally. Along this orbit, the RNN exhibits regions of slow dynamics (*clusters*) each corresponding to a distinct output class of the reference HMMs, with *transitions* occurring between them (Fig. 2D). Critically, transitions between slow zones result in large changes in the output probabilities. As the number of latent states increases in *linear-chain* HMMs, RNNs do not form additional slow regions along the orbits (App. Fig. 12). Instead, they capture finer emission discretizations by modulating the alignment of readout axes with the plane containing the orbital dynamics (Appendix L, Fig.25). In contrast, for *fully-connected* and *cyclic* architectures, RNNs develop multiple orbits connecting distinct pairs of slow regions, reflecting the richer structure of the underlying HMMs. These dynamics reveal a *compositional dynamical motif* in which the same fundamental unit combines to generate increasingly complex latent structures (Fig. 3G and App. Fig. 13).

Emergence of orbital dynamics during training. PCA projections reveal a clear transition in the latent dynamics across training epochs (Fig. 4A, purple), marking the shift from a stable fixed point to orbital dynamics. As training progresses, unstable eigenvalues emerge, marking the destabilization of the fixed-point regime. This transition coincides with a rise in eigenvalues near the imaginary axis, indicating the onset of oscillatory activity (Fig.4C), and with a characteristic *double-descent* in the loss curve [11] (Fig.16). Functionally, this transition enables sustained quasi-periodic oscillations that encode the temporal dynamics of the target HMM (Fig. 8). The radius of the resulting orbits scales

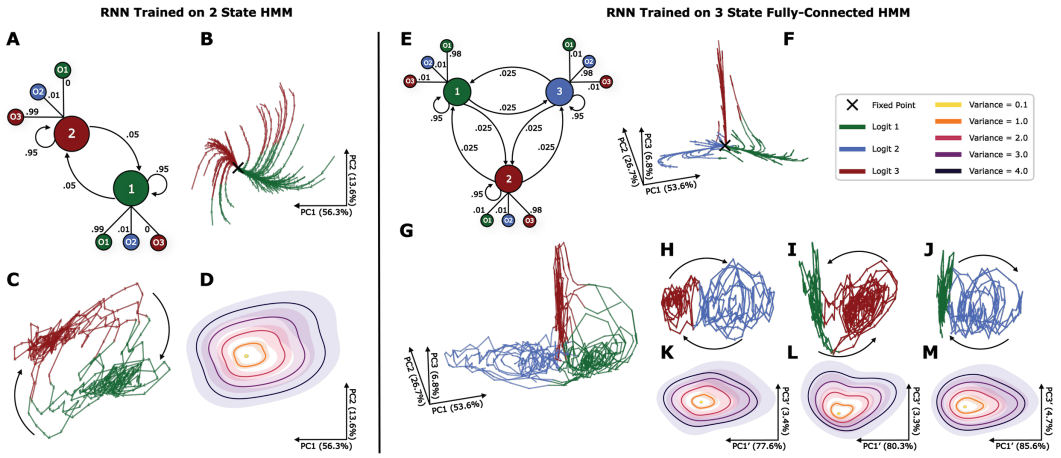

Figure 3: **Global latent dynamics of trained RNNs.** Panels **A** and **E** illustrate the HMM architectures approximated by the corresponding RNNs shown on the left and right. Hidden-state trajectories are projected onto the first two (or three) principal components of the activity and colored by the dominant logit. **B**, **F**: in the absence of input, trajectories from random initial conditions converge to a single fixed point (cross). **C**, **G**: under Gaussian input noise, activity evolves along stable orbits with slow regions associated to distinct outputs; arrows indicate flow. More complex dynamics in **G** can be decomposed into the same fundamental motif observed in **C** by applying a second-level PCA within the first PCA space, shown in **H–J**. **D**, **K–M**: contour plots (95% CI) of hidden-state densities under increasing input variance ($\sigma^2 \in \{0.1, 1.0, 2.0, 3.0, 4.0\}$) reveal a linear scaling between input variance and orbit radius. Results for all HMM structures are provided in Appendix Figures 12, 13.

linearly with input variance (Fig. 3D), a relationship confirmed by a perturbative analysis of the RNN dynamics (Appendix H). Under unbiased Gaussian input, first-order perturbations average out, while second-order terms —scaling linearly with variance — dominate after the transition, explaining how stochastic input shapes the emergent orbital regime (Fig.4D). Over training, the network converges to a stable average rate of transitions between clusters (Fig.4B). In Section 5, we examine how this behavior arises from the interplay between recurrent connectivity and input noise.

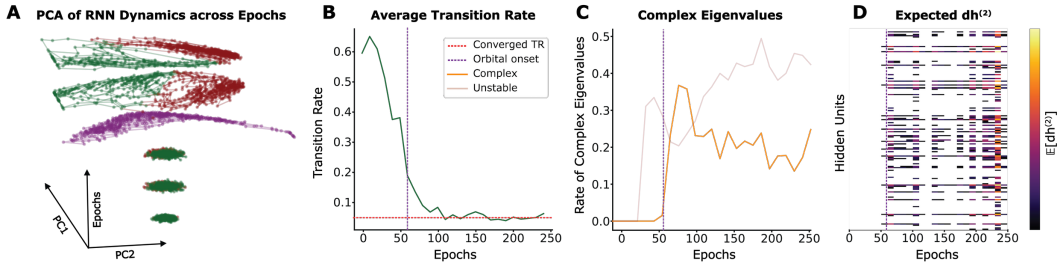

Figure 4: **A:** PCA projections of the latent dynamics across epochs for an RNN trained on a 2-states HMM, green and red indicate the dominant logit across epochs. The model first learns a single-fixed point, then becomes unstable (purple), and finally transitions to orbital dynamics. **B:** The RNN learns to encode emission probabilities of the target HMM, by means of a specific rate of transitions between the two *clusters*. **C:** The RNN becomes unstable just before the transition, then the complex eigenvalues appear on the imaginary axis. **D:** The expected second order perturbation vector $\mathbb{E}[dh^{(2)}]$ emerges in the proximity of the transition. This perturbation approximates how the variance of the noise affects the free recurrent dynamics, capturing the dependency of the orbital dynamics on the input noise variance. Results validating these findings for all HMM structures are provided in Appendix E (Figs. 14, 15).

4.2 Local Latent Dynamics: Clusters, Transitions, and Kick-Zones

Having characterized the global dynamics, we zoom in on the local properties of the orbital dynamics to understand *where* and *how* the network implements transitions between slow zones. Standard linearization techniques around fixed points (Section 2) offer limited insight here, as the observed convergence towards the single fixed point does not explain the full emergent dynamics (Fig. 3). More insights can be obtained by considering how activity evolves in the presence of the input noise on different, short rollouts initialized from a given state-space location (Fig. 5). Specifically, we segment state-space into different "zones" based on three measures computed at each location: the *residency time* (RT), defined as the average number of steps required in a rollout until a change in the dominant logit occurs; the *logit sign-change count*, quantifying how often the dominant logit's gradient flips before a change in the dominant logit occurs, with lower values indicating more directed flow; and the *number of unstable directions*, obtained via Jacobian linearization and Möbius-transformed eigenvalues, which reveals local sensitivity to perturbations (Appendix G).

Three dynamical regimes. Segmenting state space by residency time (RT) reveals three functional zones, each with distinct dynamical signatures (Fig. 5A). We find similar zones in all RNNs, irrespective of the architecture of the target HMM (Appendix I: Figs. 17,18,19).

- **Clusters** (RT > 8): regions where trajectories linger the longest, with frequent logit-gradient sign changes (in the range 5–20), and essentially only contracting eigenvalues. These are locally stable regions, each corresponding to a different probability distribution over the outputs.
- **Kick-zones** ($2 \leq$ RT ≤ 8): located downstream of clusters, these regions exhibit moderate logit-gradient sign changes (around 2–4) and a few unstable directions locally stretch the flow, indicating a local push away from the cluster.
- **Transitions** (RT < 2): Once trajectories cross the kick-zone, they enter short-lived corridors where the system moves nearly deterministically toward the next cluster. These regions exhibit few logit-gradient sign changes (< 1), reflecting a stable and directed flow.

Noise Sensitivity. To further validate the functional relevance of these regions we explicitly probe their sensitivity to noise (Fig. 5B) for the RNN trained on a 2-state HMM; Appendix I, Fig. 20 for all other *linear-chain* RNNs. We sample initial conditions from both *cluster* and *transition* regions and generate trajectories under three noise conditions: *identical* ($\gamma = 0$), *partially resampled* ($\gamma = 0.5$), and *fully independent* ($\gamma = 1$). Transition regions show minimal variability across noise conditions — once the kick-zone is crossed, trajectories proceed quasi-deterministically toward the next cluster. In contrast, cluster regions are noise-sensitive: as γ increases, trajectories diverge, with some crossing the kick-zone and others returning to the cluster. We validated these qualitative differences with quantitative measures of divergence: (i) the trace of the covariance matrix per timestep, capturing dispersion; and (ii) the time-course of the average Euclidean distance between individual trajectories and the mean trajectory (Appendix I, Fig. 21).

5 Mechanistic Interpretability: single neuron computations and connectivity

In the previous section we described a computational mechanism at the level of population-level dynamics, which relies in particular on transitions between slow regions. Here we aim to explain how these transitions emerge based on key features of the RNN connectivity and the resulting single unit properties. We focus on an RNN trained on the two-state HMM and report the analyses for the other configurations in Appendices I, J and K.

Discovering "kick-neurons". Among all RNN units, *two separate triplets of neurons* have pre-activation values (before *ReLU*) with a distinctive spatial profile: pre-activation values are strongly negative within *clusters*, pass through a near-zero regime in the *kick-zones*, and become positive during *transitions* (Appendix J, Fig. 22). The intermediate regime places the units near the *ReLU* activation threshold, where small variations in input can determine the opening of the *ReLU gate* — a mechanism we describe in the next paragraph. Each triplet is linked to one transition direction, firing at the onset of movement between clusters and emerging as the dominant non-zero components of the second-order perturbation vector $\mathbb{E}[dh^{(2)}]$ (described in Section 4.1), which reflects the network's

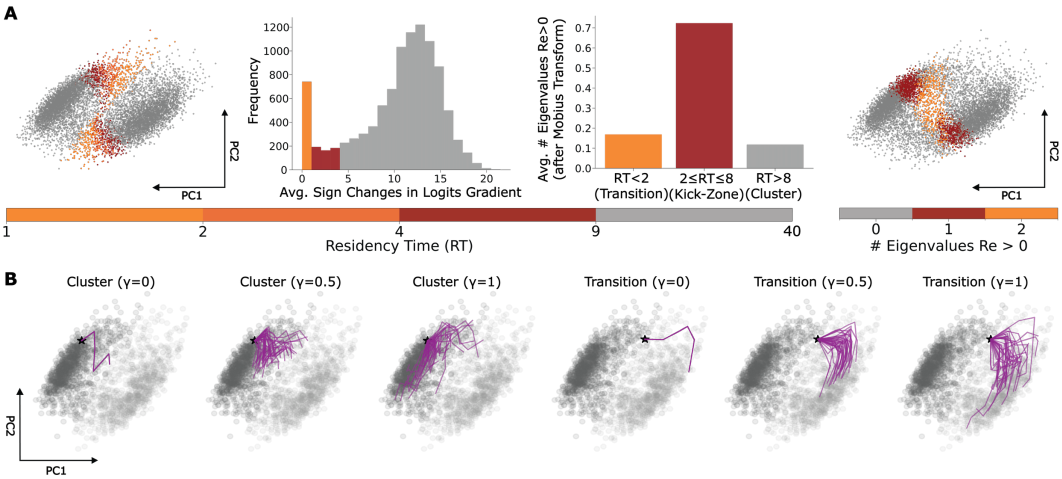

Figure 5: **Clusters, Transitions, and Kick-Zones. A**: Left to right: (i) Residency time (RT) reveals slow (gray) and fast (colored) regions; (ii) logit-gradient sign changes histogram shows a bi-modal distribution, separating stable clusters from transitions; (iii) average number of unstable directions per region (via Möbius-transformed Jacobian), peaks in kick-zones and is lowest in clusters; (iv) spatial distribution of the number of real eigenvalues with positive real part, highlighting localized instability in kick-zones contrasted with the stability of clusters. **B**: Noise sensitivity analysis from sampled states (black star) in clusters (left) and transitions (right) with 30 trajectories (in purple) generated under increasing noise resampling conditions ($\gamma \in \{0, 0.5, 1\}$). Transitions are robust across noise conditions, while clusters exhibit increasing dispersion, indicating higher noise sensitivity. Panels **A**, **B** show results for RNNs trained on 5-state HMMs. Results validating these findings for the remaining configurations are provided in Appendix I (Figs. 17,18,19 and 20).

sensitivity to input variance. Together, these properties suggest a causal role in generating noise-driven "kicks" that initiate state changes, prompting us to term them *kick-neurons*.

Noise as the trigger. To better understand the *kick-mechanism*, we examine the recurrent weight matrix (W_{hh}) and find that *kick-neurons* within each triplet form mutually excitatory connections, while inhibiting the opposing triplet (Fig. 6). Similar to work by [14], showing that input noise integration in RNNs can give rise to independent subpopulations, we observe two larger neuronal groups (comprising ~ 70 neurons), each forming recurrent excitatory loops within themselves while projecting inhibitory connections to the other. These populations interact with the *kick-neurons* through structured excitatory and inhibitory connections, suggesting a role as noise integrators that modulate the *transition gate* opening, hence we refer to them as *noise-integrating populations*.

Causal Interventions. To confirm this mechanism, we performed targeted interventions (Figure 7). We modulated either the neurons' activity directly, or the noise-drive they receive through projections from the associated *noise-integrating populations*. A modulation factor μ controlled both types of perturbations.

Ablation ($\mu = 0$). Silencing the *kick-neurons* or ablating the input noise to the corresponding *integrating population* traps the trajectory within its current cluster, preventing state transitions. Both manipulations mirror each other, confirming that noise-driven activation is required to open the *transition gate*. This intervention leads to a loss of critical eigenvalue pairs from the Jacobian spectrum, reflecting the collapse of the orbital dynamics and reversion to a single fixed-point regime.

Enhancement ($\mu = 2$). Doubling *kick-neuron* activity or amplifying the noise-drive from the corresponding *integrating population* causes overshoots beyond the target cluster. These effects resemble an increase in noise variance, and the number of critical eigenvalue pairs remains unchanged, indicating that enhancing activity preserves the rotational regime in this setting.

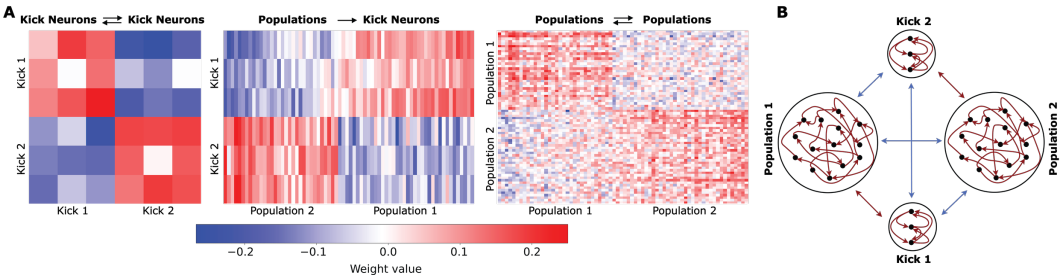

Figure 6: **Recurrent weight matrix and *Kick-circuit*. A**: Left to right: (i) Sub-matrix of W_{hh} restricted to the six *kick-neurons*. Within-triplet weights are positive (red), whereas cross-triplet weights are negative (blue), indicating mutual excitation and reciprocal inhibition. (ii) Weights from *noise-integrating neurons* to *kick-neurons* (sorted). Each integrating population excites one triplet and inhibits the other. (iii) Recurrent weights within the integrating populations show within-population excitation and cross-population inhibition. **B**: *Kick-circuit*: red arrows indicate excitation, blue inhibition. The circuit comprises two self-exciting, mutually inhibiting loops that project to opposing *kick-neuron* triplets, forming a noise-integration mechanism that implements stochastic transitions between slow regions. Results for linear-chain and fully-connected HMMs in App. Figs. 23, 24.

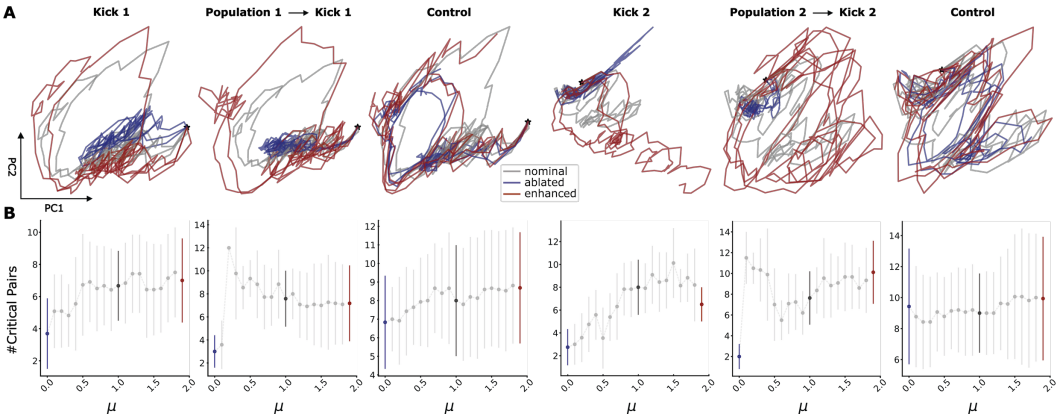

Figure 7: **Causal interventions validate the kick-circuit. A**: Latent trajectories in PCA projection. We sample initial conditions (ICs) from the two cluster regions and let trajectories evolve from the same IC with identical input noise draws in unperturbed (gray), ablated (blue), and enhanced (red) regimes for 60 time-steps. Suppressing either the *kick-neurons* (columns: 1, 4) or their noise-drive from *integrating populations* (columns: 2, 5) prevents transitions between clusters, while enhancement overshoots trajectories past the opposite cluster. Control manipulations on neurons not part of *noise-integrating populations* (columns: 3, 6) preserve *kick-neurons* noise-drive and "cluster-switching". **B**: Stability signature of the interventions. Mean±s.d. number of critical eigenvalue pairs as a function of the modulation factor μ. Ablation ($\mu = 0$, blue) eliminates *critical-pairs*, consistent with reversion to a single fixed-point regime; enhancement ($\mu = 2$, red) maintains the number of *critical-pairs*, preserving the orbital dynamics. Grey points show intermediate values of μ.

6 Stochastic Resonance

The above analyses show that our RNN exhibits noise-sustained orbital dynamics (Figure 3), driven by structured interactions between two functionally distinct neural populations: slow, *noise-integrating* units, and fast-responding *kick-neurons* (Figure 5, 6). As noise accumulates in the slow subsystem, trajectories drift along quasi-stable manifolds until they reach a region — the *kick-zone* — which triggers a rapid, deterministic transition to the next attractor-like cluster. This mechanism results in robust, quasi-periodic alternation between network states, even in the absence of any external periodic input. This phenomenon bears resemblance to a class of noise-induced dynamics known as *self-induced stochastic resonance* (SISR) [33]. In contrast to classical stochastic resonance, which requires an external periodic signal [2], SISR arises intrinsically in systems with time-scale separation. Weak noise applied to a fast subsystem perturbs the system off a slow manifold, initiating

deterministic excursions that form coherent, noise-controlled oscillations [48]. Our RNN appears to operate in an analogous regime: the slow populations accumulate stochastic input over time while in the cluster zones, whereas the fast *kick-neurons* drive sharp transitions once a noise-modulated threshold is reached. This mechanism enables the emergence of stable oscillatory patterns, with their period governed by the interplay between noise variance and slow integration dynamics. The oscillation period, linked in our case to the emission probabilities, can be shaped through learning by adjusting the effective time-constant of the slow subsystem. This interpretation is supported by the population-level dynamics shown in Figure 8, where each cycle reflects a transition between attractor-like states, coordinated through the interaction of slow integrators and fast responders whose activity rises sharply before the transition. In this way, the network effectively harnesses internal noise as a computational signal, leveraging SISR-like dynamics to perform structured, probabilistic inference, thus allowing the RNN to emulate the HMM behavior.

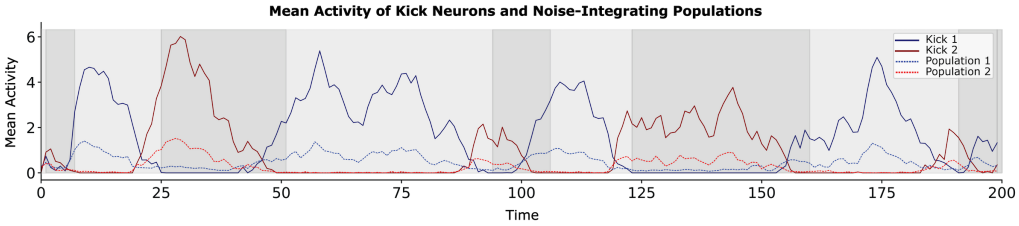

Figure 8: **Oscillatory dynamics.** Mean activity of kick-neuron triplets (solid lines) and noise-integrating populations (dashed lines) as the RNN switches cluster regions (alternating gray bands). These populations exhibit clear anti-phase oscillations: while one integrating group dominates, it reinforces itself and its matching kick-neurons, driving the system toward a transition. This dynamic reflects a *self-induced stochastic resonance* (SISR) regime [33, 48], where slow noise integration and fast resets ("kicks") produce quasi-periodic switching without external periodic input.

7 Conclusion

In this work, we explored how Recurrent Neural Networks (RNNs) can implement discrete, stochastic latent structure through continuous dynamics. We introduced a training pipeline that fits *vanilla* RNNs to various families of Hidden Markov Models (HMMs). Contrary to the expectation of an n-well landscape with one fixed point per HMM state, trained RNNs converge to a single fixed point in the absence of input and, under noise, exhibit noise-sustained orbital dynamics: slow regions that encode distinct emission probabilities, separated by short, deterministic transitions. Mechanistically, it relies on two complementary sub-circuits: large noise-integrating populations and fast kick-neurons, whose interplay converts input variance into quasi-periodic transitions. This connectivity structure harnesses internal noise as a computational signal, enabling RNNs to reproduce HMM-like probabilistic behavior via a reusable dynamical motif and paralleling features of neural activity observed in the brain.

Cortical activity is intrinsically noisy at both micro- and macro-scales, driven by stochastic ion channel dynamics [6], probabilistic synaptic transmission [1], and the high-dimensional, recurrent nature of cortical circuits, which produces ongoing, noisy background activity even in the absence of external input [13], potentially as a consequence of chaotic dynamics [8]. Far from being a nuisance, such variability can enhance information processing — increasing sensitivity to weak or sub-threshold signals and facilitating the coordination of distributed brain regions in the processing of sensory information via *stochastic resonance* [17, 32, 34].

Our findings demonstrate that unconstrained RNNs can uncover both discrete and continuous latent structure directly from data — without any imposed topological priors — and reveal that these networks naturally converge toward biologically plausible circuit motifs. Taken together, these results point to a compositional dynamical primitive in which slow noise integration and fast kick-triggered resets cooperate to generate discrete state transitions, and multiple instances of this motif can combine to produce richer latent dynamics. Overall, this work positions Recurrent Neural Networks as a powerful alternative to Hidden Markov Models for modeling the latent structure and neural mechanisms of natural behaviors.

References

[1] Christina Allen and Charles F Stevens. An evaluation of causes for unreliability of synaptic transmission. *Proceedings of the National Academy of Sciences*, 91(22):10380–10383, 1994.

[2] Roberto Benzi, Alfonso Sutera, and Angelo Vulpiani. The mechanism of stochastic resonance. *Journal of Physics A: mathematical and general*, 14(11):L453, 1981.

[3] Dean V Buonomano and Wolfgang Maass. State-dependent computations: spatiotemporal processing in cortical networks. *Nature Reviews Neuroscience*, 10(2):113–125, 2009.

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

[5] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[6] CE Dangerfield, David Kay, and Kevin Burrage. Stochastic models and simulation of ion channel dynamics. *Procedia Computer Science*, 1(1):1587–1596, 2010.

[7] Sandeep Robert Datta, David J Anderson, Kristin Branson, Pietro Perona, and Andrew Leifer. Computational neuroethology: a call to action. *Neuron*, 104(1):11–24, 2019.

[8] Brian DePasquale, David Sussillo, LF Abbott, and Mark M Churchland. The centrality of population-level factors to network computation is demonstrated by a versatile approach for training spiking networks. *Neuron*, 111(5):631–649, 2023.

[9] Laura N Driscoll, Krishna Shenoy, and David Sussillo. Flexible multitask computation in recurrent networks utilizes shared dynamical motifs. *Nature Neuroscience*, 27(7):1349–1363, 2024.

[10] Daniel Durstewitz, Georgia Koppe, and Max Ingo Thurm. Reconstructing computational system dynamics from neural data with recurrent neural networks. *Nature Reviews Neuroscience*, 24(11):693–710, 2023.

[11] Lukas Eisenmann, Zahra Monfared, Niclas Göring, and Daniel Durstewitz. Bifurcations and loss jumps in rnn training. *Advances in Neural Information Processing Systems*, 36:70511–70547, 2023.

[12] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[13] A Aldo Faisal, Luc PJ Selen, and Daniel M Wolpert. Noise in the nervous system. *Nature reviews neuroscience*, 9(4):292–303, 2008.

[14] Arnaud Fanthomme and Rémi Monasson. Low-dimensional manifolds support multiplexed integrations in recurrent neural networks. *Neural Computation*, 33(4):1063–1112, 2021.

[15] Jean Feydy, Thibault Séjourné, François-Xavier Vialard, Shun-ichi Amari, Alain Trouve, and Gabriel Peyré. Interpolating between optimal transport and mmd using sinkhorn divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2681–2690, 2019.

[16] Aude Genevay, Gabriel Peyré, and Marco Cuturi. Learning generative models with sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics*, pages 1608–1617. PMLR, 2018.

[17] Bruce J Gluckman, Theoden I Netoff, Emily J Neel, William L Ditto, Mark L Spano, and Steven J Schiff. Stochastic resonance in a neuronal network from mammalian brain. *Physical Review Letters*, 77(19):4098, 1996.

[18] Christine Grienberger and Arthur Konnerth. Imaging calcium in neurons. *Neuron*, 73(5):862–885, 2012.

[19] Michael Häusser. Optogenetics: the age of light. *Nature methods*, 11(10):1012–1014, 2014.

[20] Elizabeth Herbert and Srdjan Ostojic. The impact of sparsity in low-rank recurrent neural networks. *PLOS Computational Biology*, 18(8):e1010426, 2022.

[21] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[22] W Jeffrey Johnston and Stefano Fusi. Abstract representations emerge naturally in neural networks trained to perform multiple tasks. *Nature Communications*, 14(1):1040, 2023.

[23] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[24] Renate Krause, Matthew Cook, Sepp Kollmorgen, Valerio Mante, and Giacomo Indiveri. Operative dimensions in unconstrained connectivity of recurrent neural networks. *Advances in Neural Information Processing Systems*, 35:17073–17085, 2022.

[25] Christopher Langdon and Tatiana A Engel. Latent circuit inference from heterogeneous neural responses during cognitive tasks. *Nature Neuroscience*, pages 1–11, 2025.

[26] Soon Hoe Lim. Understanding recurrent neural networks using nonequilibrium response theory. *Journal of Machine Learning Research*, 22(47):1–48, 2021.

[27] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[28] Niru Maheswaranathan, Alex Williams, Matthew Golub, Surya Ganguli, and David Sussillo. Reverse engineering recurrent networks for sentiment classification reveals line attractor dynamics. *Advances in neural information processing systems*, 32, 2019.

[29] Niru Maheswaranathan, Alex Williams, Matthew Golub, Surya Ganguli, and David Sussillo. Universality and individuality in neural dynamics across large populations of recurrent networks. *Advances in neural information processing systems*, 32, 2019.

[30] Valerio Mante, David Sussillo, Krishna V Shenoy, and William T Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *nature*, 503(7474):78–84, 2013.

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

[32] Mark D McDonnell and Derek Abbott. What is stochastic resonance? definitions, misconceptions, debates, and its relevance to biology. *PLoS computational biology*, 5(5):e1000348, 2009.

[33] Cyrill B Muratov, Eric Vanden-Eijnden, et al. Self-induced stochastic resonance in excitable systems. *Physica D: Nonlinear Phenomena*, 210(3-4):227–240, 2005.

[34] Alexander N Pisarchik and Alexander E Hramov. Coherence resonance in neural networks: Theory and experiments. *Physics Reports*, 1000:1–57, 2023.

[35] Reidar Riveland and Alexandre Pouget. Natural language instructions induce compositional generalization in networks of neurons. *Nature Neuroscience*, 27(5):988–999, 2024.

[36] Lee Sharkey, Bilal Chughtai, Joshua Batson, Jack Lindsey, Jeff Wu, Lucius Bushnaq, Nicholas Goldowsky-Dill, Stefan Heimersheim, Alejandro Ortega, Joseph Bloom, et al. Open problems in mechanistic interpretability. *arXiv preprint arXiv:2501.16496*, 2025.

[37] Jimmy Smith, Scott Linderman, and David Sussillo. Reverse engineering recurrent neural networks with jacobian switching linear dynamical systems. *Advances in Neural Information Processing Systems*, 34:16700–16713, 2021.

[38] Christos Sourmpis, Carl Petersen, Wulfram Gerstner, and Guillaume Bellec. Trial matching: capturing variability with data-constrained spiking neural networks. *Advances in Neural Information Processing Systems*, 36:74787–74798, 2023.

[39] Micha E Spira and Aviad Hai. Multi-electrode array technologies for neuroscience and cardiology. *Nature nanotechnology*, 8(2):83–94, 2013.

[40] Nicholas A Steinmetz, Cagatay Aydin, Anna Lebedeva, Michael Okun, Marius Pachitariu, Marius Bauza, Maxime Beau, Jai Bhagat, Claudia Böhm, Martijn Broux, et al. Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings. *Science*, 372(6539):eabf4588, 2021.

[41] 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.

[42] 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.

[43] Jane X Wang, Zeb Kurth-Nelson, Dharshan Kumaran, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Demis Hassabis, and Matthew Botvinick. Prefrontal cortex as a meta-reinforcement learning system. *Nature neuroscience*, 21(6):860–868, 2018.

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

[45] Caleb Weinreb, Jonah E Pearl, Sherry Lin, Mohammed Abdal Monium Osman, Libby Zhang, Sidharth Annapragada, Eli Conlin, Red Hoffmann, Sofia Makowska, Winthrop F Gillis, et al. Keypoint-moseq: parsing behavior by linking point tracking to pose dynamics. *Nature Methods*, 21(7):1329–1339, 2024.

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

[47] Klavdia Zemlianova, Amitabha Bose, and John Rinzel. Dynamical mechanisms of how an rnn keeps a beat, uncovered with a low-dimensional reduced model. *Scientific Reports*, 14(1):26388, 2024.

[48] Jinjie Zhu and Hiroya Nakao. Stochastic periodic orbits in fast-slow systems with self-induced stochastic resonance. *Physical Review Research*, 3(3):033070, 2021.

Contents

A Hidden Markov Model Architectures

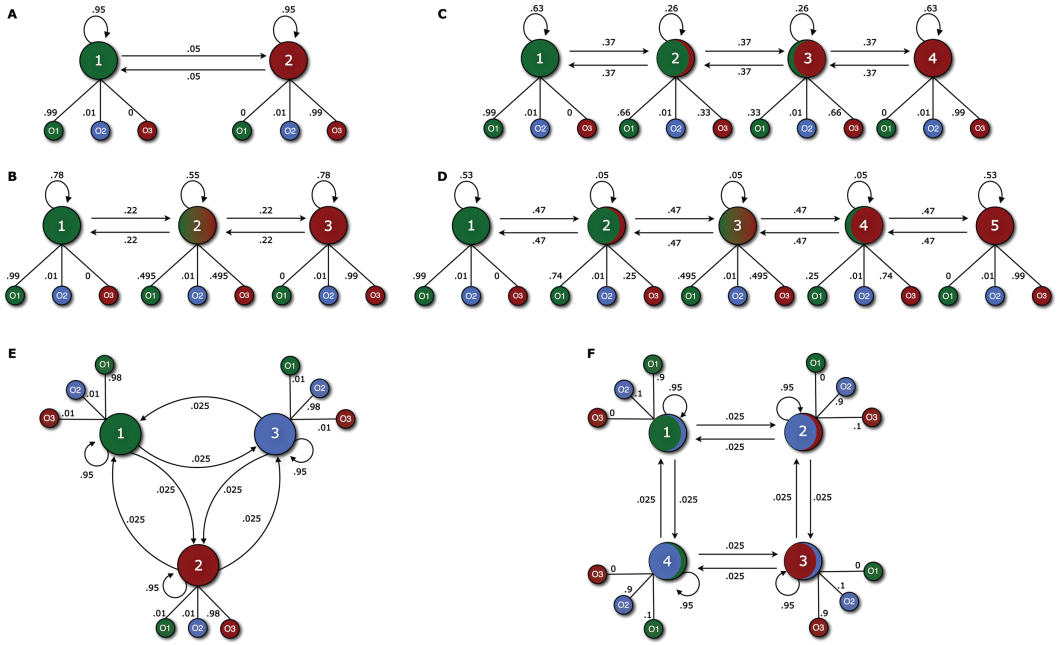

Figure 9: **Hidden Markov Model architectures used for training.** Panels **A-D** illustrate a spectrum of *linear-chain* HMMs with $M \in \{2, 3, 4, 5\}$ latent states (from left to right, top to bottom). Arrows indicate state transitions, with labels showing transition probabilities; the hidden states of the HMM (larger circles) have been colored according to the most likely outcome associated with a specific state, or by a mixture of them. Smaller circles represent emission distributions for each state over the observation alphabet $\mathcal{O} = \{1, 2, 3\}$, with green, blue, and red corresponding to emissions o_1, o_2, and o_3, respectively. All models are constructed to preserve a constant probability $\rho = 0.05$ of reaching the most distant state in $M-1$ steps. As M increases, the latent representation becomes progressively finer, transitioning from highly discrete (**A**) to quasi-continuous (**D**). **E**: *Fully-connected* HMM with three latent states, each capable of transitioning to any other (including self-transitions). Emission probabilities are biased toward one dominant output per state, while retaining smaller weights for the others, producing a symmetric, all-to-all transition structure. **F**: *Cyclic* HMM with four latent states arranged in a bidirectional loop. Each state emits a characteristic pair of outputs, with one output dominant and the other weaker, while adjacent states share one common output.

B Training Regime and Computational Resources

We conducted a systematic sweep across key architectural and task parameters to evaluate how RNNs replicate the emission statistics of different HMM families. Specifically, we trained networks on four *linear-chain* HMMs ($M = 2, 3, 4, 5$ latent states) as well as on the *fully-connected* and *cyclic* architectures described in Section 3.1. For the *linear-chain* HMMs, we train networks with three different hidden state sizes ($|h| = 50, 150, 200$) and four input noise dimensionalities ($d = 1, 10, 100, 200$), yielding 48 unique configurations. Each configuration is trained with three independent random seeds, resulting in a total of 144 models. For the *fully-connected* and *cyclic* architectures, we fixed the hidden size to $|h| = 150$ and input dimensionality to $d = 100$, training three independent seeds per architecture, resulting in six additional models. Only seeds that successfully converged were retained for analysis.

The training regime is kept consistent across all configurations. Each RNN is trained on 30,000 sequences of fixed length, sampled from its corresponding HMM: 100 for $M = 2$, 30 for $M = 3, 4$, 40 for $M = 5$, 30 for the *fully-connected*, and 40 for the *cyclic* architectures. Optimization is performed in batches of 4096 using the Adam optimizer [23] with a learning rate of 0.001. Hidden states are initialized to zero at the start of training, and all weights are drawn from a uniform

distribution $\mathcal{U}(-\sqrt{k}, \sqrt{k})$, where $k = \frac{1}{\texttt{hidden\_size}}$. To stabilize learning and mitigate exploding gradients, we apply gradient clipping with a maximum norm of 0.9 for the *linear-chain* models and 0.3 for the *fully-connected* and *cyclic* architectures. Training proceeded until convergence, typically reached within 1000 epochs. However, convergence becomes less consistent as the number of hidden states in the target HMM increases, in which case shorter training sequences were used to facilitate convergence.

The complete pipeline—including HMM data generation, RNN training, computation of evaluation metrics, PCA analyses, and visualization—was optimized for efficient execution on modern hardware. On an NVIDIA RTX 4090 GPU, each model completed training in approximately 5–20 minutes, depending on sequence length and network size.

C Performance Metrics

To evaluate how well task-optimized RNNs replicate the behavior of the reference HMMs, we developed four performance metrics that capture both global sequence similarity and fine-grained statistical properties. These include: Euclidean distances between matched sequences, squared differences in transition matrices, observation frequencies, and observation volatility. Below we detail the definition, motivation, and implementation of each metric.

***Sinkhorn-aligned* euclidean distance.** To quantify the global discrepancy between HMM and RNN sequences, we compute the Euclidean distance after aligning predicted and reference sequences via the Sinkhorn divergence. This procedure ensures a principled pairing between sequences by solving a soft optimal transport problem, where matched sequences minimize expected pairwise distances under an entropy-regularized transport plan.

Given two sequences of categorical outputs, $\boldsymbol{y}_{\text{rnn}}$ and $\boldsymbol{y}_{\text{hmm}}$, represented as one-hot vectors in $\mathbb{R}^{T \times C}$ (where T is sequence length and C the number of output categories), we flatten them into vectors of dimension $T \cdot C$ and compute the Euclidean distance:

$$d(\boldsymbol{y}_{\text{rnn}}, \boldsymbol{y}_{\text{hmm}}) = \sqrt{\sum_{i=1}^{T \cdot C} \left(y_{\text{rnn}}^{(i)} - y_{\text{hmm}}^{(i)} \right)^2}$$

Distances are computed across batches of $N = 5000$ sequences, averaged over random seeds and configurations. As a baseline, the same metric is computed between pairs of HMM-generated sequences. These distributions are visualized in Fig. 10, demonstrating that RNN outputs closely match the HMM baseline.

Transition matrix squared differences. To assess how well the RNN captures the temporal dependencies between successive outputs, we compute empirical transition matrices from the sequences. For each model, we extract the most likely output at each time step and count the empirical frequency of transitions from output i to j:

$$T_{ij} = \frac{\#\text{ transitions from output } i \text{ to } j}{\#\text{ total transitions from } i}$$

We compute T^{rnn} and T^{hmm}, and compare them via element-wise squared differences:

$$\Delta_{ij} = \left(T_{ij}^{\text{rnn}} - T_{ij}^{\text{hmm}} \right)^2$$

The resulting matrices are averaged across models and shown in Fig. 11, providing a detailed account of how accurately each RNN reproduces the internal transition structure of its target HMM.

Observation frequencies. This metric assesses the RNN's ability to reproduce the stationary distribution of HMM outputs. We count the frequency of each output class across all time steps and sequences, yielding a probability vector $p^{\text{rnn}} \in \mathbb{R}^C$. We compare this to the ground-truth distribution p^{hmm} and, although not shown in figures, this analysis confirms that most trained networks match long-term HMM statistics closely.

Observation Volatility. To quantify short-term dynamics, we measure how frequently the RNN changes its output across time. Volatility is then averaged across sequences and compared with that of the HMM. This metric complements observation frequency by detecting under- or over-smoothing in the RNN's emission process.

C.1 Sinkhorn-Aligned Euclidean Distances

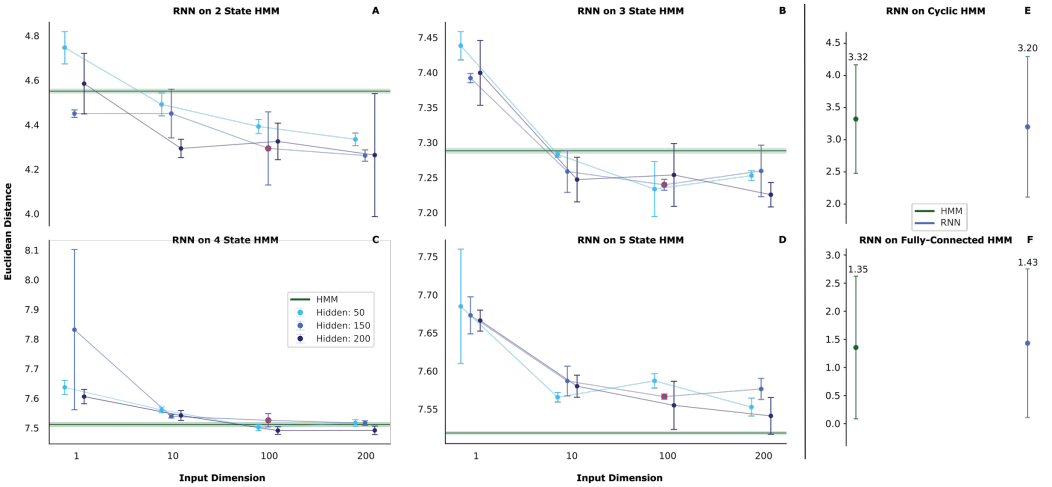

Figure 10: **Sinkhorn-aligned Euclidean distances.** Mean $\pm$ s.d. Euclidean distances between Sinkhorn-aligned output sequences generated by trained RNNs and their corresponding HMM references, across all combinations of hidden-state size ($|h| \in 50, 150, 200$) and input-noise dimensionality ($d \in 1, 10, 100, 200$). Panels **A–D** show results for *linear-chain* HMMs with $M \in 2, 3, 4, 5$ latent states, while panels **E–F** correspond to the *cyclic* and *fully-connected* architectures, respectively. Error bars indicate variability across three random seeds per configuration. Horizontal green bands represent the baseline distance obtained by comparing HMM-generated sequences against themselves. Red-circled points mark the selected RNN configuration ($|h| = 150$, $d = 100$) used in the mechanistic analyses presented throughout the paper. For *linear-chain* HMMs, increasing input dimensionality systematically improves alignment with the reference emissions, approaching the HMM baseline across hidden-state sizes. For *fully-connected* and *cyclic* architectures, RNNs trained with the same configuration ($|h| = 150$, $d = 100$) achieve similarly close correspondence to their HMM targets.

C.2 Transition Matrices Squared Differences

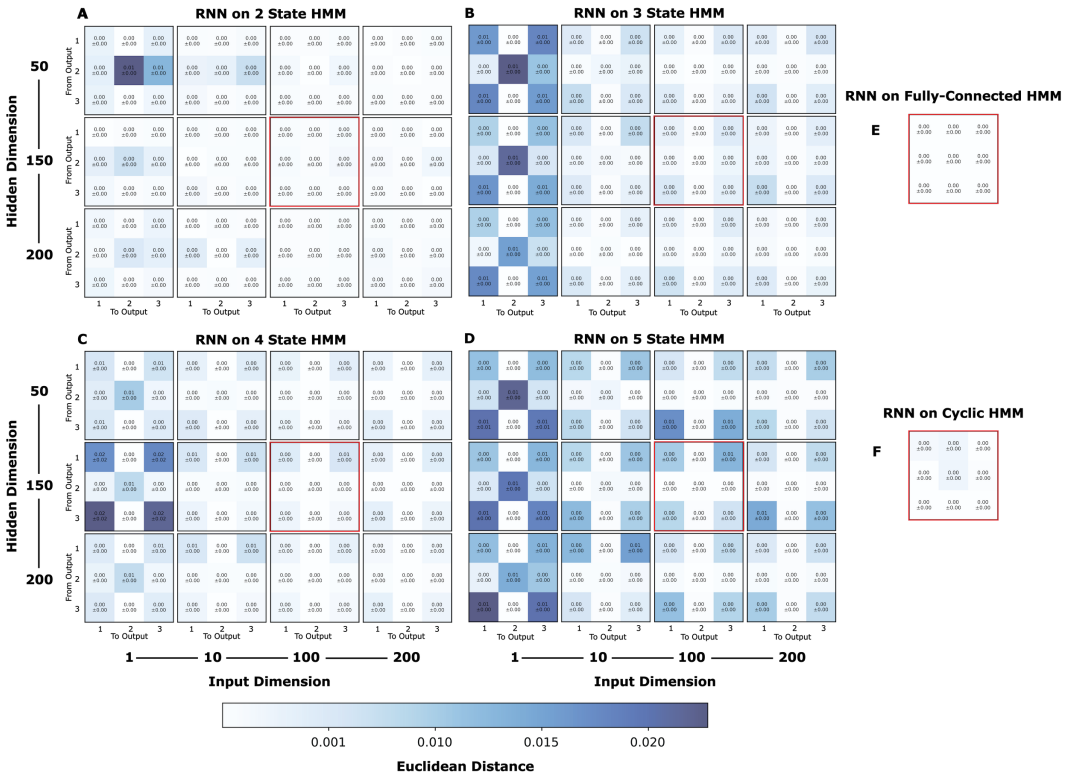

Figure 11: **Transition matrices squared differences.** Mean ± s.d. of the element-wise squared differences between empirical transition matrices computed from RNN outputs and those of their reference HMMs, across all combinations of hidden-state size ($|h| \in 50, 150, 200$) and input-noise dimensionality ($d \in 1, 10, 100, 200$). Panels **A–D** correspond to *linear-chain* HMMs with $M \in 2, 3, 4, 5$ latent states, while panels **E–F** report results for the *fully-connected* and *cyclic* architectures, respectively. Errors are averaged over three random seeds per configuration. Darker colors indicate larger deviations from the reference transition matrices, while lighter tones reflect closer alignment. Red-outlined boxes highlight the selected RNN configurations ($|h| = 150$, $d = 100$) used in the mechanistic analyses throughout the paper. For the *linear-chain* architectures, transition-matrix alignment improves systematically with increasing input dimensionality, approaching near-perfect correspondence at $d \geq 100$. The *fully-connected* and *cyclic* models, trained with the same configuration, achieve comparably low deviation levels.

D Trajectories for RNNs trained on HMMs

D.1 Trajectories for Linear-Chain HMMs

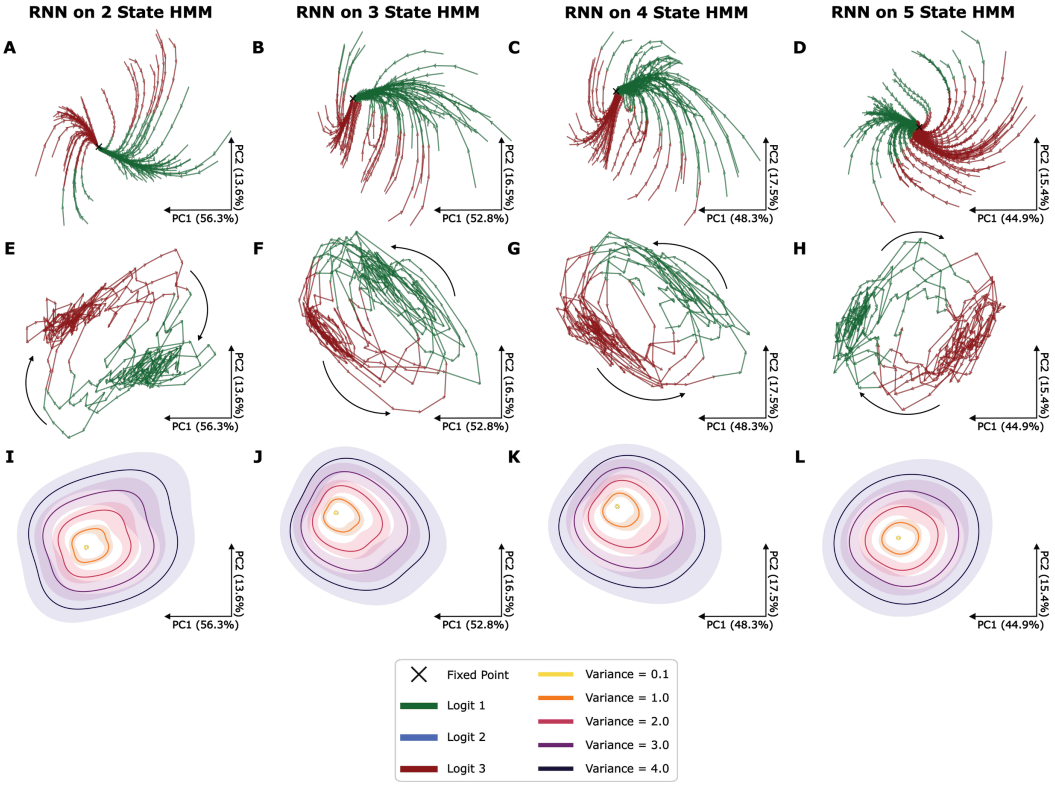

Figure 12: **Trajectories for linear-chain HMMs.** Each column shows results from RNNs ($|h| = 150$) trained to reproduce the emission statistics of *linear-chain* HMMs with $M \in \{2, 3, 4, 5\}$ latent states. Hidden-state trajectories are projected onto the first two principal components (same axes across rows). **A–D**: trajectories from random initial conditions converge to a single fixed point (cross) in the absence of input. **E–H**: under Gaussian input noise, trajectories evolve along stable orbits with distinct regions corresponding to different dominant logits (colors). Arrows indicate average flow direction. **I–L**: hidden-state density contours (95% CI) under increasing input variance ($\sigma^2 \in \{0.1, 1.0, 2.0, 3.0, 4.0\}$) reveal a linear scaling between orbit radius and input variance, while preserving overall shape.

D.2 Trajectories for Fully-Connected and Cyclic HMMs

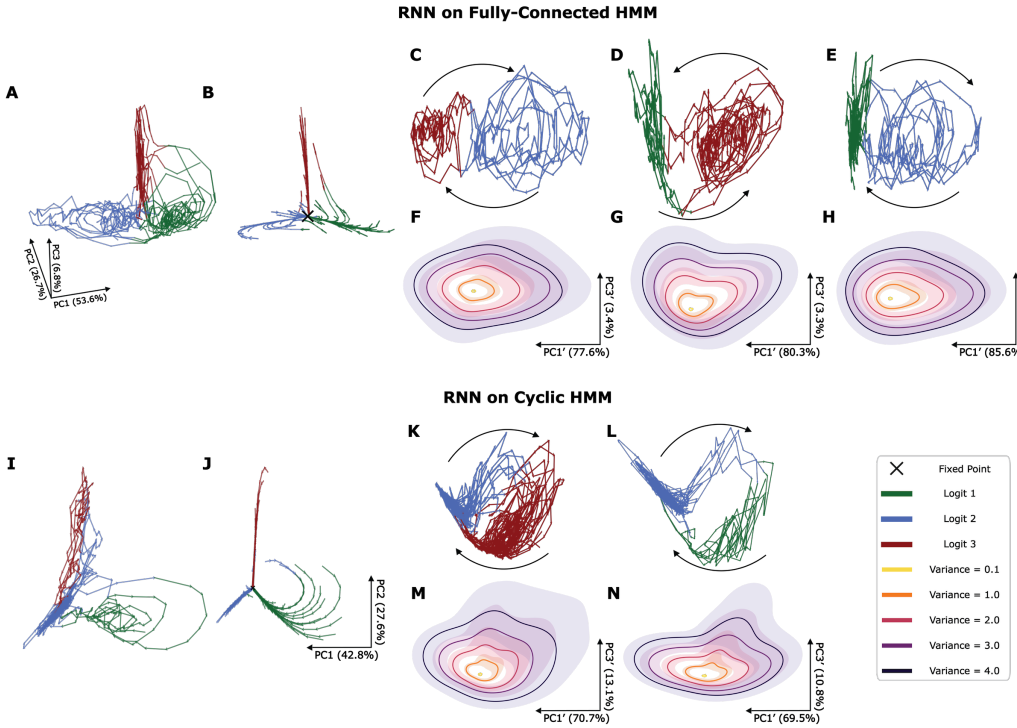

Figure 13: **Trajectories for fully-connected and cyclic HMMs.** Hidden-state dynamics of RNNs ($|h| = 150$) trained to reproduce the emission statistics of *fully-connected* (top) and *cyclic* (bottom) HMMs. Hidden-state trajectories are projected onto the first three principal components in **A–B** and **I–J**. To visualize orbital dynamics linking any given pair of clusters, within the subspace in **A–B** and **I–J** we computed a second set of PC component separately for activity restricted to each pair, shown in **C–H** and **K–N**. **B, J**: Under Gaussian input noise, trajectories evolve along multiple stable orbits connecting slow regions dominated by distinct logits (colors). These orbital trajectories can be decomposed into the same fundamental dynamical primitives described in the text, shown in **C–E** and **K–L**, respectively. **A, I**: In the absence of input, trajectories from random initial conditions converge to a single fixed point (cross). **F–H** and **M–N**: Hidden-state density contours (95 % CI) under increasing input variance ($\sigma^2 \in \{0.1, 1.0, 2.0, 3.0, 4.0\}$) reveal that the linear scaling between input variance and orbit radius is preserved.

E Learning Trajectories Across Training Epochs

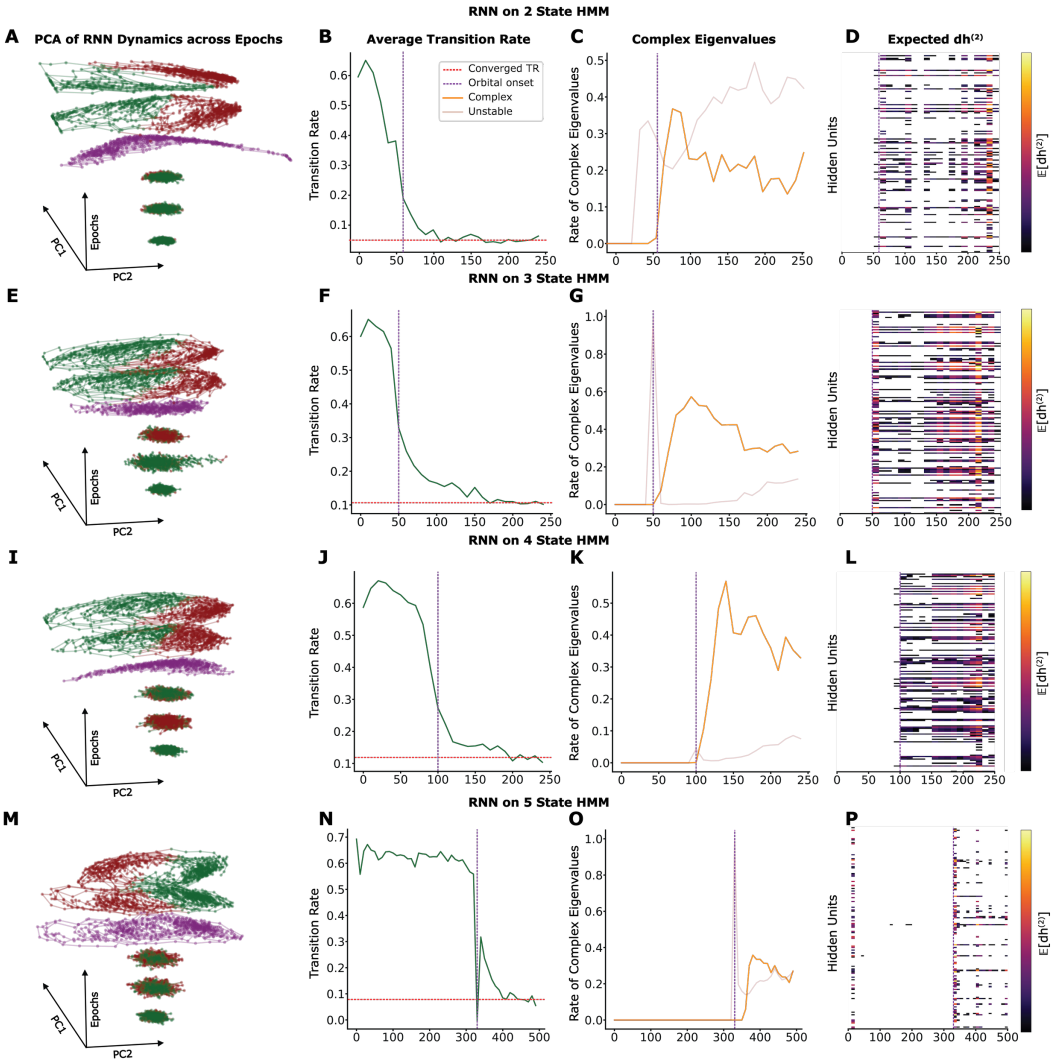

Figure 14: **Learning trajectories across training epochs for linear-chain HMMs.** Each row corresponds to an RNN trained on a *linear-chain* HMM with $M \in \{2, 3, 4, 5\}$ latent states (top to bottom). **A, E, I, M**: PCA projections of hidden-state dynamics across training epochs reveal the emergence of structured orbital dynamics: networks initially converge to a single fixed point, undergo a transient unstable regime (purple), and ultimately form stable, noise-sustained orbital dynamics (colored by dominant logit). **B, F, J, N**: the average transition rate between output clusters, showing a clear shift after the transition epoch (purple dashed line) aligning with the converged transition rate (red dotted line). **C, G, K, O**: the fraction of unstable and complex Möbius-transformed eigenvalues of the Jacobian rises sharply around the transition, reflecting, respectively, the destabilization of the fixed point and the onset of oscillatory dynamics. **D, H, L, P**: the expected second-order perturbation vector $\mathbb{E}[dh^{(2)}]$ emerges after this transition, capturing how input noise variance drives the recurrent dynamics and sustains the orbital regime. Together, these results demonstrate a consistent learning trajectory across architectures, in which the network transitions from stable to oscillatory dynamics.

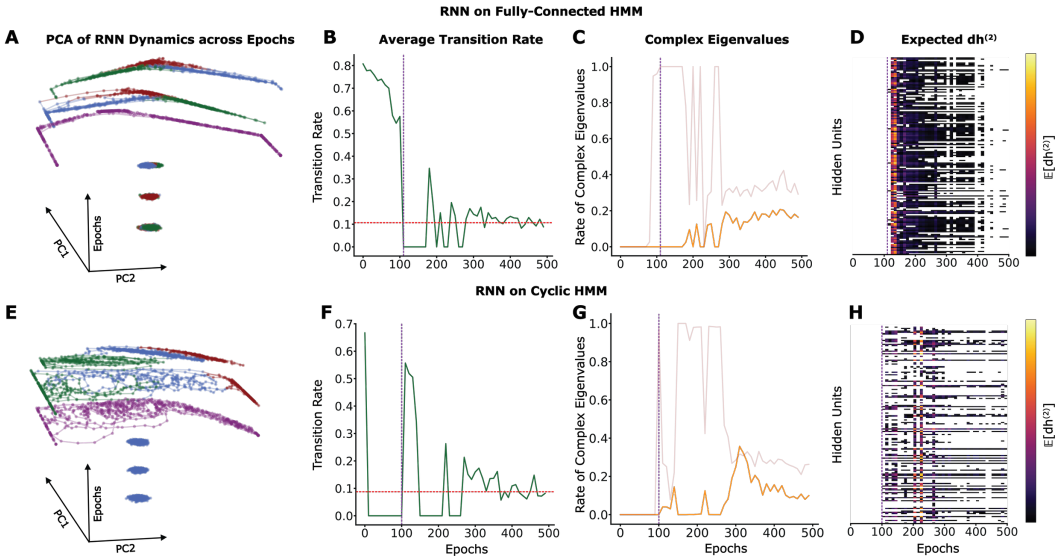

Figure 15: **Learning trajectories across training epochs for fully-connected and cyclic HMMs.** Each row corresponds to an RNN ($|h| = 150$) trained to reproduce the emission statistics of a *fully-connected* (top) or *cyclic* (bottom) HMM. **A, E**: PCA projections of hidden-state dynamics across training epochs show the progressive emergence of structured orbital dynamics: networks initially converge to a single fixed point, undergo a transient unstable regime (purple), and ultimately form stable, noise-sustained orbital dynamics (colored by dominant logit). **B, F**: the average transition rate between output clusters, showing a clear shift after the transition epoch (purple dashed line) aligning with the converged transition rate (red dotted line). **C, G**: the fraction of unstable and complex Möbius-transformed eigenvalues of the Jacobian rises sharply around the transition, reflecting, respectively, the destabilization of the fixed point and the onset of oscillatory dynamics. **D, H**: the expected second-order perturbation vector $\mathbb{E}[dh^{(2)}]$ emerges after this transition, capturing how input noise variance drives the recurrent dynamics and sustains the orbital regime. These results demonstrate that RNNs trained on more complex HMMs follow the same qualitative learning trajectory observed for linear-chain models, transitioning from a single attractor to oscillatory dynamics.

F Loss Curves

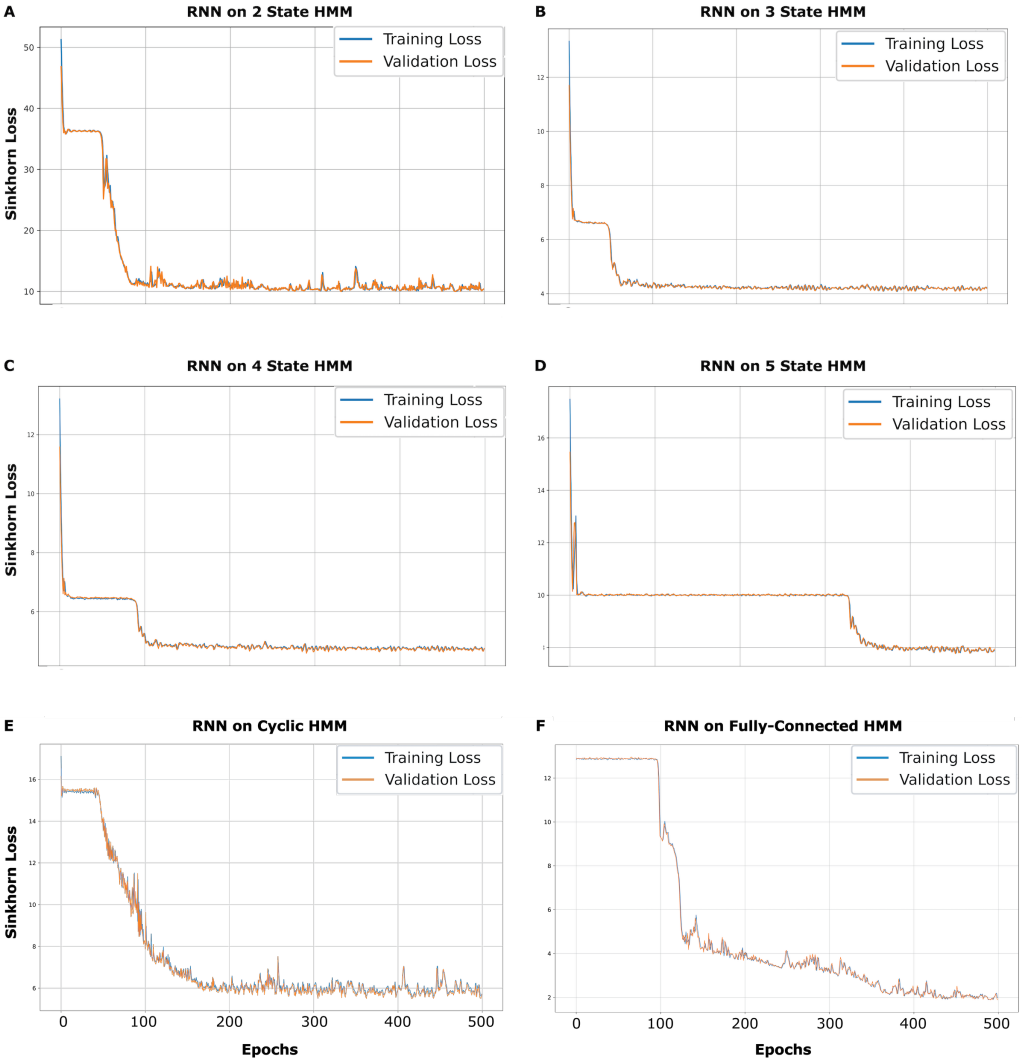

Figure 16: **Training and validation loss curves.** Sinkhorn loss over training epochs for the six RNN models used in the main analyses ($|h| = 150$, $d = 100$), each trained to reproduce the emission statistics of an *HMM* with different transition structures: *linear-chain* $M \in \{2, 3, 4, 5\}$, *fully-connected*, and *cyclic*. Training and validation losses (blue and orange) closely overlap across all configurations, indicating no signs of overfitting. All models converge within 500 epochs, exhibiting a characteristic *double-descent* profile. The second loss drop coincides with the transient unstable regime identified in Figures 14 - 15, marking the emergence of noise-sustained orbital dynamics and the transition from fixed-point to rotational behavior. This correspondence confirms that the onset of orbital dynamics constitutes the dynamical solution through which the networks minimize the Sinkhorn loss.

G Jacobian Linearization and Stability Analysis via Möbius Transformation

In a vanilla RNN with ReLU non-linearity the hidden state evolves as

$$h_t \;=\; \phi\big(x_t\, W_{\text{in}}^T + h_{t-1}\, W_{hh}^T\big), \qquad \phi(z) = \max(0, z). \tag{A.1}$$

Because the derivative of the ReLU is either 0 or 1, the Jacobian that propagates an infinitesimal perturbation δh_{t-1} to δh_t takes a particularly simple form:

$$J_t \;=\; W_{hh}^T D_t, \qquad D_t \;=\; \operatorname{diag}\big[\mathbb{1}_{z_t>0}\big], \qquad z_t = x_t\, W_{\text{in}}^T + h_{t-1}\, W_{hh}^T. \tag{A.2}$$

The diagonal *gating matrix* D_t switches rows of W_{hh} on or off depending on whether the corresponding pre-activation is positive.

Discrete-time stability and the need for a spectral map. In discrete time a fixed point is locally stable iff all eigenvalues satisfy $|\lambda_i| < 1$. To interpret this spectrum with the intuition of continuous-time systems (where the relevant boundary is $\operatorname{Re}\lambda_i = 0$) we apply the classical Möbius transformation

$$\mu(\lambda) \;=\; \frac{1+\lambda}{1-\lambda}, \qquad \lambda \neq 1. \tag{A.4}$$

This bijection maps the unit circle to the imaginary axis and preserves complex conjugacy, giving the correspondence

Condition in λ	After mapping	Interpretation
$\lvert\lambda\rvert < 1$	$\operatorname{Re}\mu(\lambda) < 0$	contraction
$\lvert\lambda\rvert = 1$	$\operatorname{Re}\mu(\lambda) = 0$	neutral / Hopf boundary
$\lvert\lambda\rvert > 1$	$\operatorname{Re}\mu(\lambda) > 0$	expansion

H Second Order Perturbation

Inspired by the perturbative approach on RNN of [26], here we compute two simple terms to estimate the impact of the first and second order effects of the input noise on the stable dynamics. Without loss of generality, we reformulate the RNN dynamics as:

$$\hat{h}_{t+1} = \phi(\hat{h}_t)W_{hh}^T + x_t W_{\text{in}}^T \epsilon$$

With ϕ representing the ReLU activation, $\hat{h}_t \in \mathbb{R}^r$ the pre-activation recurrent state, $x_t \in \mathbb{R}^n$ with $x_t \sim \mathcal{N}(0, \sigma^2 \mathbb{I}_n)$, and z_i the $i-th$ pre-activation neuron. We recover the formulation of equation (2) in Section 3.2 with $h_t = \phi(\hat{h}_t)$. Let's define the unperturbed dynamics

$$\hat{h}_t^o = \phi(\hat{h}_{t-1}^o)W_{hh}^T$$

Such that $\hat{h}_{t_c}^0 = \hat{h}_{t_c-1}^0$ for some $t_c > 0$. We consider the first order perturbation around it

$$\hat{h}_t = \hat{h}_t^0 + \delta h_t$$

By substituting we obtain the following:

$$\hat{h}_{t+1}^0 + \delta h_{t+1} = \phi(\hat{h}_t'^0 + \delta h_t)W_{hh}^T + x_t W_{\text{in}}^T \epsilon,$$

$$\delta h_{t+1} = \phi(\hat{h}_t^0 + \delta h_t)W_{hh}^T - \phi(\hat{h}_t^0)W_{hh}^T + x_t W_{\text{in}}^T \epsilon.$$

Given the ReLU activation, we approximate for small δh_t:

$$\phi(\hat{h}_t^0 + \delta h_t) \approx \phi(\hat{h}_t^0) + \delta h_t D_\phi(\hat{h}_t^0),$$

where $D_\phi(\hat{h}_t^0)$ is diagonal, with entries 1 if $(\hat{h}_t^0)_i > 0$, 0 otherwise. Thus:

$$\delta h_{t+1} \approx \delta h_t D_\phi(\hat{h}_t^0)W_{hh}^T + x_t W_{\text{in}}^T \epsilon.$$

Naming $A_t = D_\phi(\hat{h}_t^o)W_{hh}^T$ and $M_{t,s} = A_{s+1}A_{s+1}...A_t = \prod_{k=s+1}^{t+1} A_k$, with $A_{t+1} = \mathbb{I}_r$.
By iterating we obtain:

$$\delta h_t = \epsilon \sum_{s=0}^{t} x_s W_{\text{in}}^T \left(\prod_{k=s+1}^{t+1} A_k \right) = \epsilon \sum_{s=0}^{t} x_s W_{\text{in}}^T M_{t,s}$$

$$\hat{h}_t = \hat{h}_t^o + \epsilon \sum_{s=0}^{t} x_s W_{\text{in}}^T M_{t,s}$$

We observe that, given the unbiased Gaussian noise, $\mathbb{E}[\delta h] = 0$. Therefore, the RNN integration mechanism must rely on higher order terms. Henceforth, we herby derive the second order term as well. To make the higher-order term explicit, Instead of $\delta h = \delta h^{(1)} + O(\sigma^2)$ we express δh_t into orders of noise:

$$\delta h_t = \delta h_t^{(1)} + \delta h_t^{(2)} + O(\sigma^3),$$

where $\delta h_t^{(1)}$ is the first-order term and $\delta h_t^{(2)}$ is the second-order term. For an element-wise activation ($\frac{\partial^2 \phi_i}{\partial z_j^2} = 0$ for $i \neq j$), we get the following recurrent form

$$\delta h_{t+1} = \delta h_t D_\phi(\hat{h}_t^0)W_{hh}^T + \frac{1}{2}\sum_{i=1}^{r} \left[(\delta h_t)_i^2 \cdot \frac{\partial^2 \phi_i}{\partial z_i^2}(\hat{h}_t^0) \right] e_i \cdot W_{hh}^T + x_t W_{\text{in}}^T + O(\|\delta h_t\|^3)$$

Given that $\delta h_t^{(2)}$ captures only the second order effects, it must satisfy the following relation

$$\delta h_{t+1}^{(2)} \approx \delta h_t^{(2)} D_\phi(\hat{h}_t^0) W_{hh}^T + \frac{1}{2} \sum_{i=1}^{r} \left[(\delta h_t^{(1)})_i^2 \cdot \frac{\partial^2 \phi_i}{\partial z_i^2}(\hat{h}_t^0) \right] e_i \cdot W_{hh}^T$$

with $\delta h_0^{(2)} = 0$. This is a linear recursion, driven also by a quadratic term in $\delta h_t^{(1)}$. Solving it we obtain:

$$\delta h_t^{(2)} = \frac{\epsilon^2}{2} \sum_{s=0}^{t} \sum_{i=1}^{r} \left[(\delta h_s^{(1)})_i^2 \cdot \frac{\partial^2 \phi_i}{\partial z_i^2}(\hat{h}_s^0) \right] e_i \cdot M_{t,s}$$

Which leads to the following expectation, depending only on the noise variance.

$$\mathbb{E}[\delta h_t^{(2)}] = \frac{\epsilon^2}{2} \sum_{s=0}^{t} \sum_{i=1}^{r} \left[\sigma^2 \left(\sum_{k=0}^{s} W_{\text{in}}^T M_{t,s} \right)_i^2 \cdot \frac{\partial^2 \phi_i}{\partial z_i^2}(\hat{h}_s^0) \right] e_i \cdot M_{t,s}$$

Unfortunately, the ReLU activation does not possess a second derivative. Henceforth, we approximate the impact of the second order perturbation with the square of the first order perturbation over several trials whenever the pre-activation neurons activity z_i is below 0:

$$dh_t^{(2)} = \sum_{k}^{t} \frac{1}{2} \left(dh_k^{(1)} \right)^2 \odot f_k \cdot M_{t,k}$$

Where $\odot$ is the Hadamard product and

$$(f_k)_i = \begin{cases} 1 \text{ if } (z_k)_i < 0 \\ 0 \text{ otherwise} \end{cases}$$

for t = 10 and averaging with respect to 100 different trajectories. This vector emerges after the orbital dynamics onset (Figure 4A,D) and scales linearly with the variance, analogous to the scaling of the orbits observed in Figure 3D, K-M. Furthermore, empirically, we observe that kick neurons comprise some of the top non-zero components of this vector, suggesting a deeper connection between variance and the firing mechanism.

I Residency times and noise sensitivity

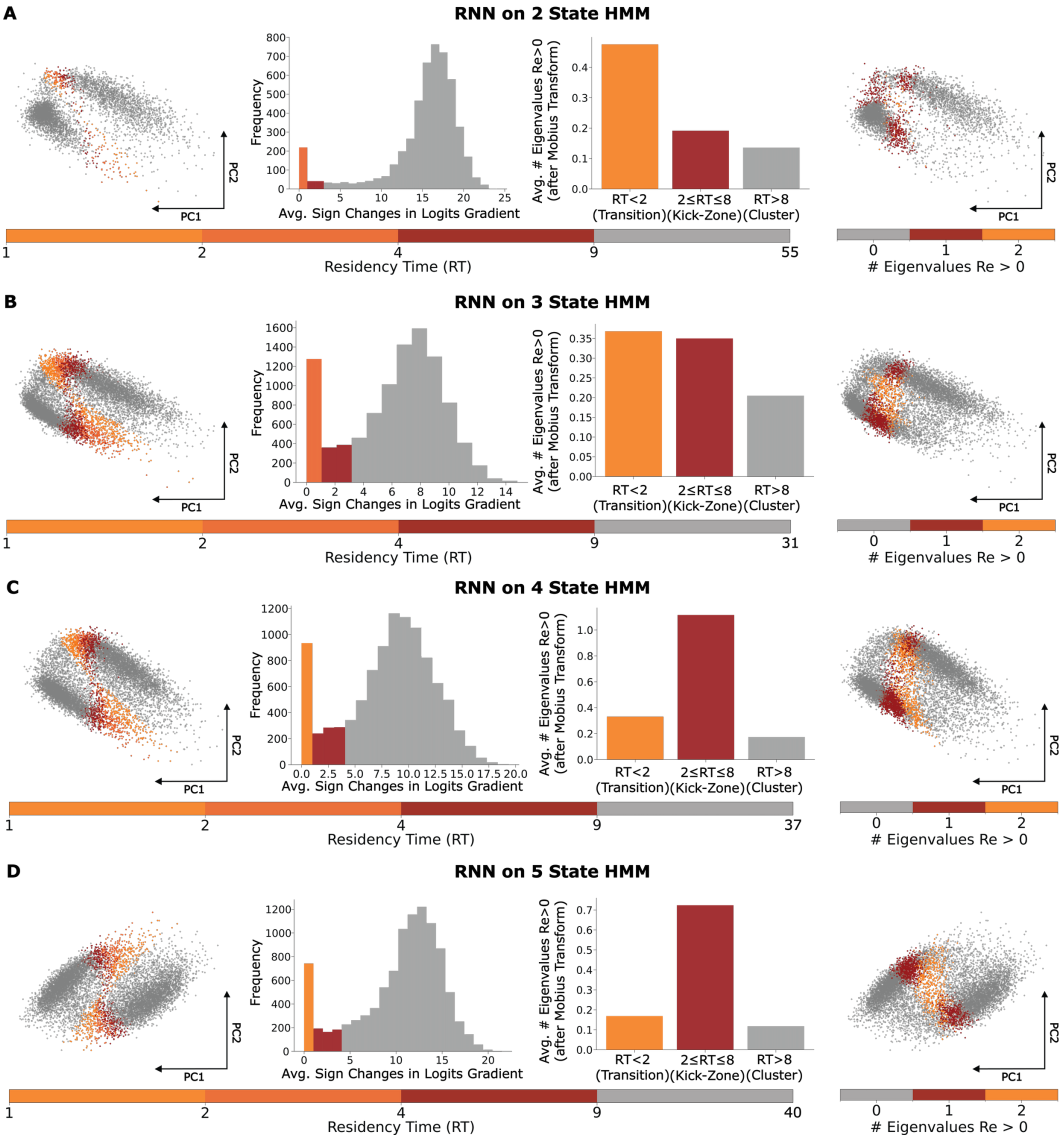

Figure 17: **Residency analyses for linear-chain HMMs.** Each row corresponds to an RNN trained on a *linear-chain* HMM with $M \in \{2, 3, 4, 5\}$ latent states (panels **A–D**). **Left**: PCA projections of the latent trajectories colored by residency time (RT) reveal distinct dynamical regimes, with slow regions (*clusters*, dark gray) where trajectories linger and fast regions (*transitions*, orange) where rapid output switching occurs. **Center-left**: distributions of average sign changes in the logit gradient exhibit a robust bimodal structure separating stable clusters from directed transition flows. **Center-right**: the average number of unstable directions (via Möbius-transformed Jacobian eigenvalues) peaks in intermediate RTs, identifying *kick-zones* that mediate transitions. **Right**: spatial maps of the number of eigenvalues with positive real part confirm local instability confined to these zones. The coherent structure across all HMM configurations demonstrates that the RNNs converge on a common solution: orbital dynamics divided into slow clusters, unstable kick-zones, and rapid transitions, suggesting a generic mechanism by which RNNs can emulate discrete stochastic HMMs emissions through continuous dynamics.

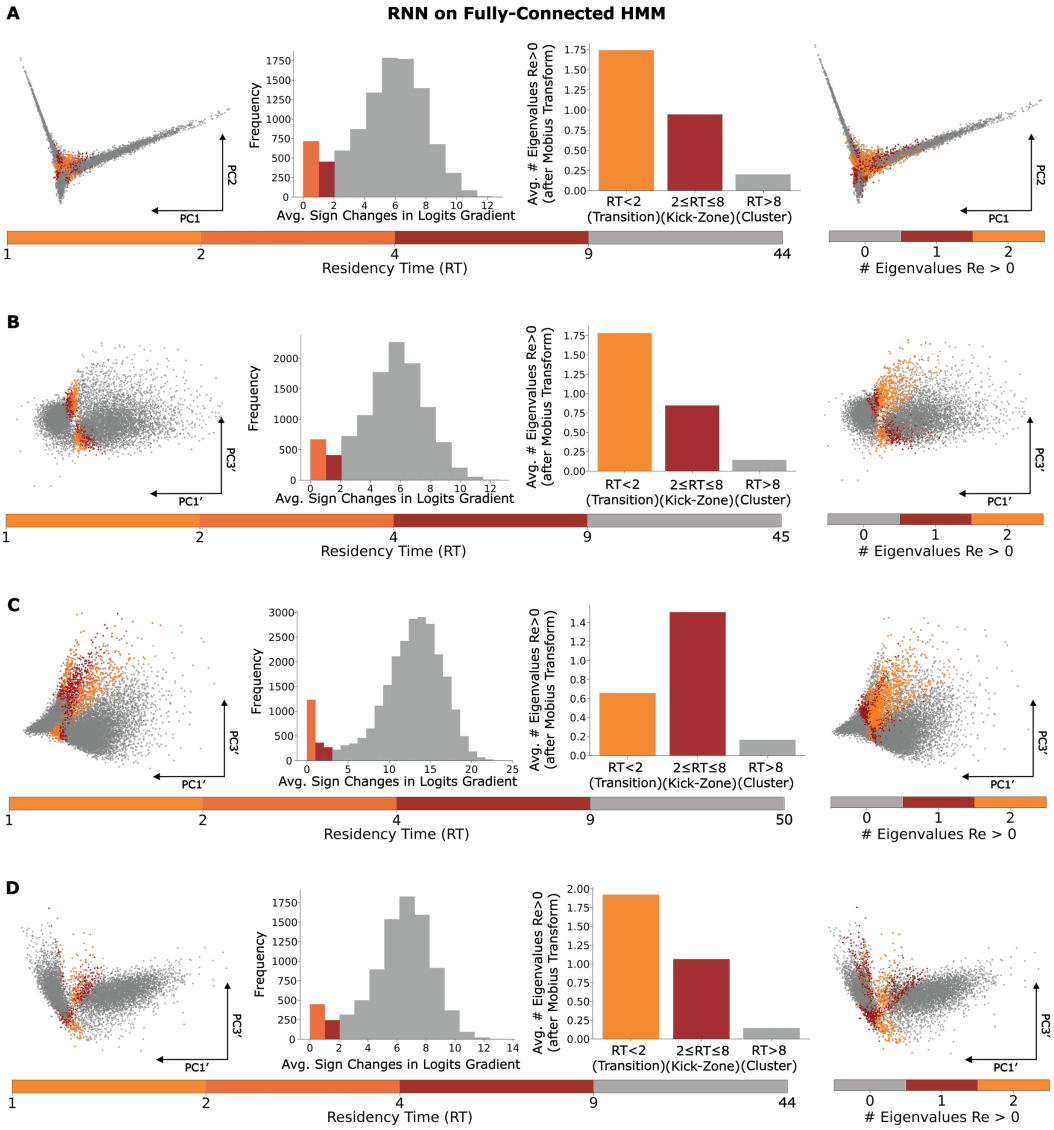

Figure 18: **Residency analyses for fully-connected HMMs.** Each row corresponds to the principal space or one of the three principal subspaces identified in the RNN trained to reproduce the emission statistics of a *fully-connected* HMM (panels **A–D**). **Left**: PCA projections of the latent trajectories colored by residency time (RT) reveal distinct dynamical regimes, with slow regions (*clusters*, dark gray) where trajectories linger and fast regions (*transitions*, orange) where rapid output switching occurs. **Center-left**: distributions of average sign changes in the logit gradient exhibit a robust bimodal structure separating stable clusters from directed transition flows. **Center-right**: the average number of unstable directions, computed from Möbius-transformed Jacobian eigenvalues, peaks in intermediate RTs, identifying locally unstable *kick-zones* that mediate transitions. **Right**: spatial maps of the number of eigenvalues with positive real part confirm that instability is confined to kick-zones while clusters remain locally stable. Together, these results show that the RNN decomposes the fully-connected HMM into three coupled dynamical subspaces, each expressing the same tripartite organization of *clusters*, *kick-zones*, and *transitions* described for the linear-chain architectures — supporting the compositional reuse of the same dynamical primitive across HMM families.

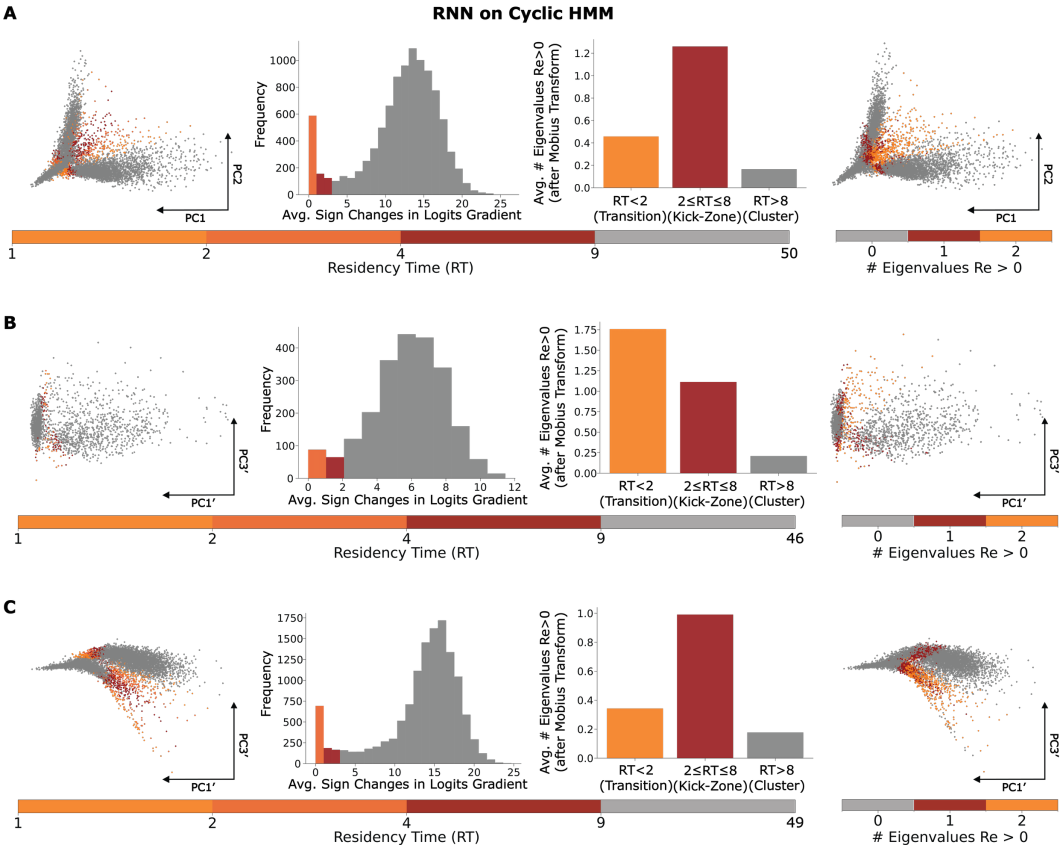

Figure 19: **Residency analyses.** Each row corresponds to the principal space or one of the two principal subspaces identified in the RNN trained to reproduce the emission statistics of a *cyclic* HMM (panels **A–C**). **Left**: PCA projections of the latent trajectories colored by residency time (RT) reveal distinct dynamical regimes, with slow regions (*clusters*, dark gray) where trajectories linger and fast regions (*transitions*, orange) where rapid output switching occurs. **Center-left**: distributions of average sign changes in the logit gradient exhibit a robust bimodal structure separating stable clusters from directed transition flows. **Center-right**: the average number of unstable directions, computed from Möbius-transformed Jacobian eigenvalues, peaks in intermediate RTs, identifying locally unstable *kick-zones* that mediate transitions. **Right**: spatial maps of the number of eigenvalues with positive real part confirm that instability is confined to kick-zones while clusters remain locally stable. Together, these results show that cyclic architectures preserve the same dynamical primitive observed in simpler HMMs.

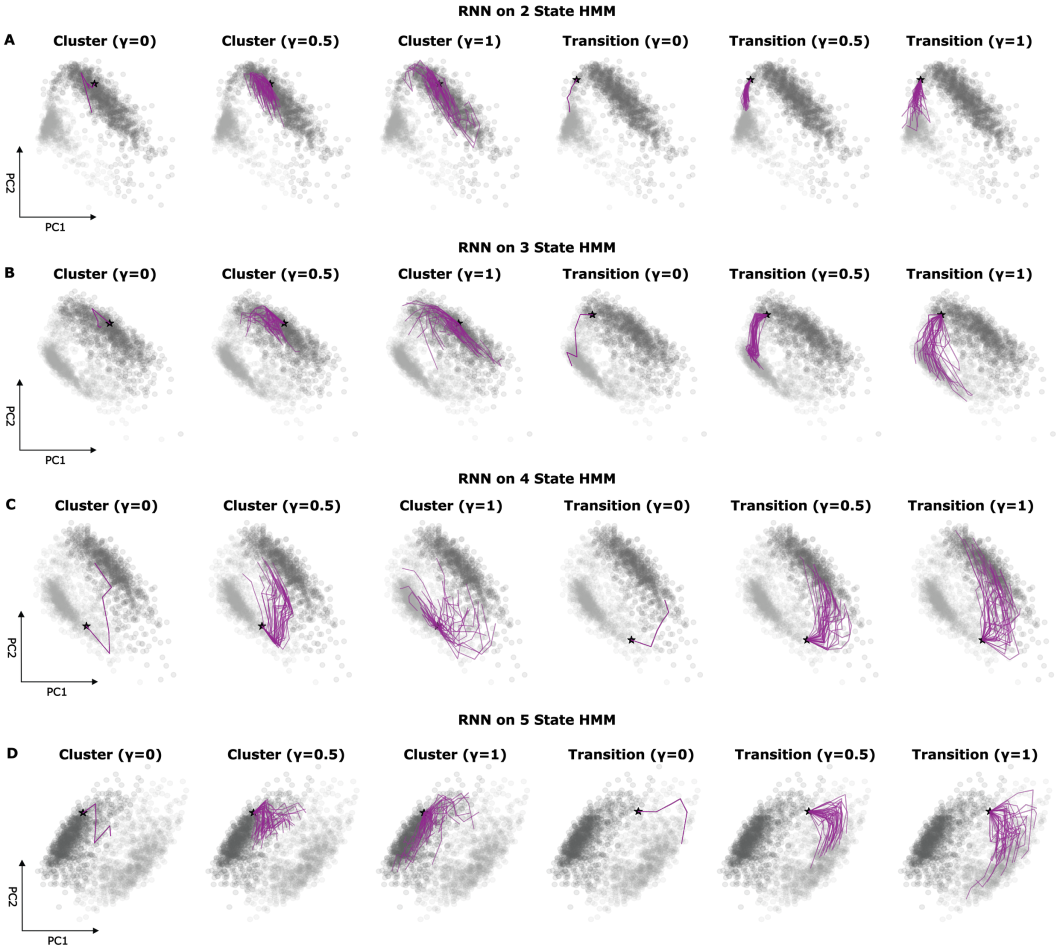

Figure 20: **Noise sensitivity analyses for linear-chain HMMs.** Each row corresponds to an RNN trained to reproduce the emission statistics of a *linear-chain* HMM with $M \in \{2, 3, 4, 5\}$ latent states (panels **A–D**). **Left to right:** Sampled initial conditions (black stars) are taken from representative *cluster* (left triplets) and *transition* (right triplets) regions in the latent space. From each location, 30 trajectories (magenta) are simulated under three levels of input-noise resampling: identical ($\gamma = 0$), partially resampled ($\gamma = 0.5$), and fully independent ($\gamma = 1$). In all configurations, transitions remain compact and exhibit robust, quasi-deterministic flow once the kick-zone is crossed, whereas clusters show increasing divergence with noise resampling, reflecting high noise sensitivity.

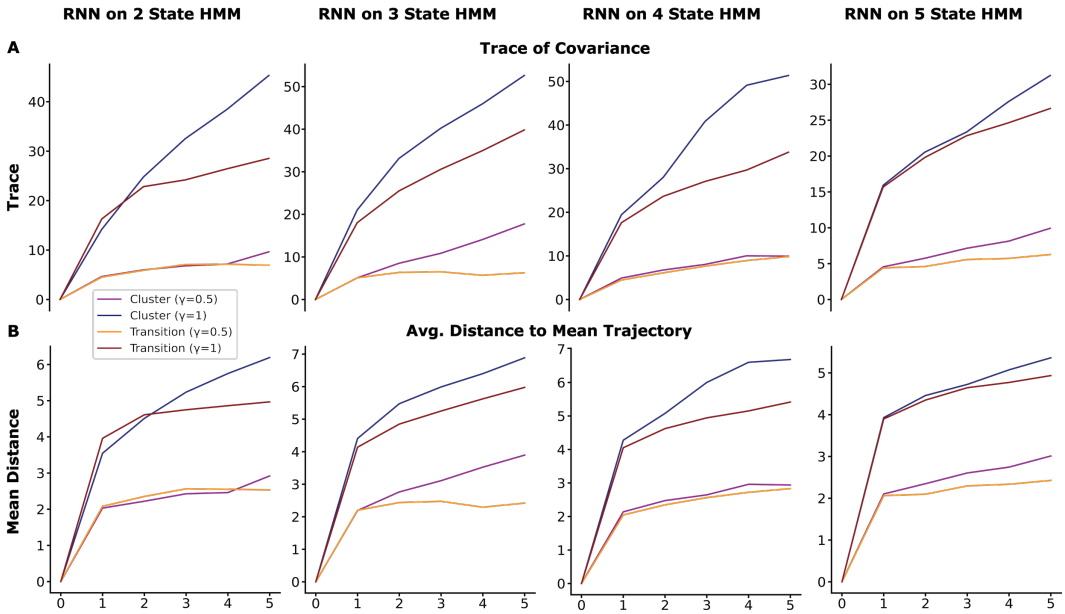

Figure 21: **Quantification of noise sensitivity for linear-chain HMMs.** Each column corresponds to an RNN trained to reproduce the emission statistics of a *linear-chain* HMM with $M \in \{2, 3, 4, 5\}$ latent states (panels **A–D**). For each configuration, we quantify the dispersion of latent trajectories initialized from representative *cluster* (purple and blue) and *transition* (orange and red) regions under two levels of noise resampling ($\gamma = 0.5,\ 1$). **A**: the *trace of the trajectory covariance matrix* captures the overall spread of trajectories in latent space. **B**: the *average Euclidean distance to the mean trajectory* reflects deviations across trials. In all configurations, transitions remain compact and exhibit robust, quasi-deterministic flow once the kick-zone is crossed, whereas clusters show progressively larger divergence with increasing noise resampling, reflecting high noise sensitivity.

J Kick-Neurons Activity

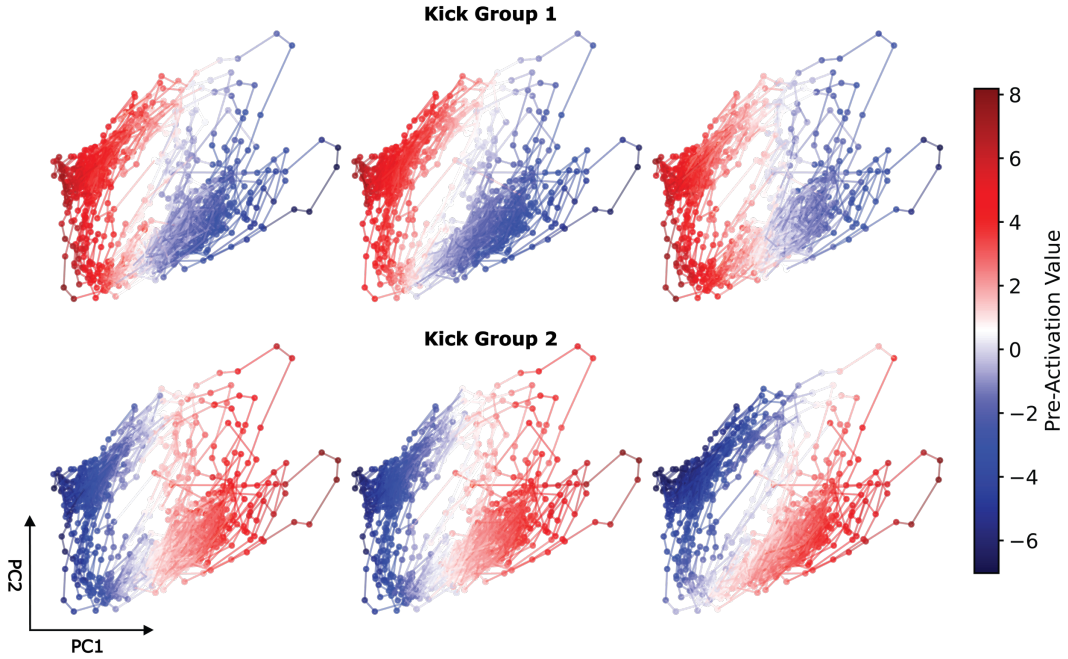

Figure 22: **Pre-activation values of kick-neurons.** Latent trajectories projected onto the first two principal components (PC1 and PC2) of the RNN trained on a 2-state HMM. Each point along a trajectory corresponds to one time step and is colored by the pre-activation value of a specific neuron. The first row shows the three neurons forming *Kick Group 1*, while the second row displays the three neurons of *Kick Group 2*, which mediate transitions in the opposite direction. In both groups, pre-activations are strongly negative within one of the two clusters, increase to near-zero values in the kick-zones, and become positive during transitions and in the subsequent cluster. These dynamics reveal a clear functional role: within clusters, the neurons are fully suppressed (ReLU outputs zero), while in the kick-zones, pre-activation values fluctuate near the ReLU threshold. This intermediate regime places the units near the *ReLU* activation threshold, where small variations in input can determine the opening of the *ReLU gate*.

K Recurrent weight matrices and Kick-circuits

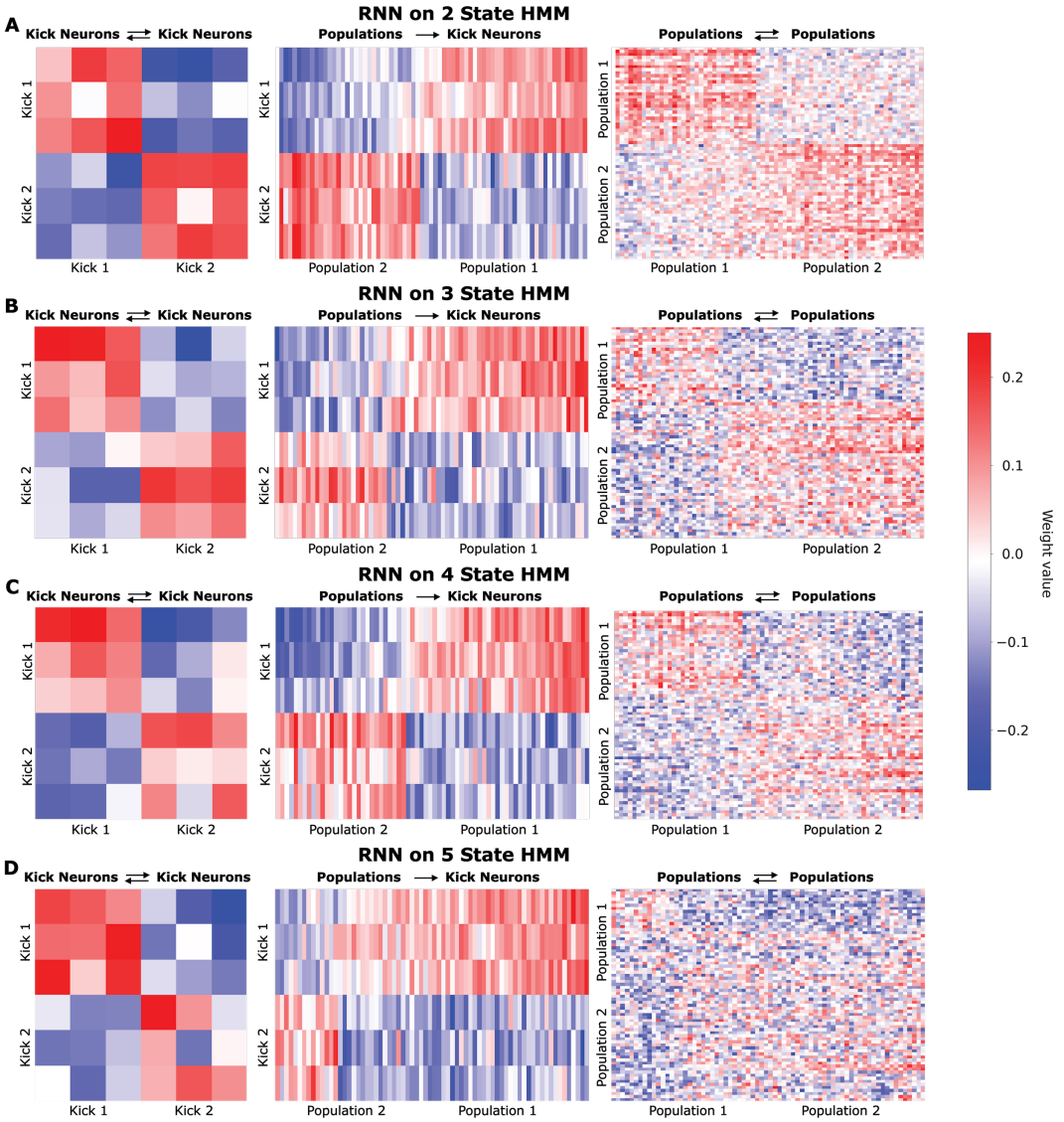

Figure 23: **Recurrent weight matrices and *Kick-circuits* for linear-chain HMMs.** Each row (**A–D**) corresponds to an RNN trained to reproduce the emission statistics of a *linear-chain* HMM with $M \in \{2, 3, 4, 5\}$ latent states. **Left**: Sub-matrices of the recurrent weights W_{hh} restricted to the identified *kick-neurons*. Within-triplet connections are predominantly excitatory (red), while cross-triplet connections are inhibitory (blue), indicating mutual excitation and reciprocal inhibition between opposing kick groups. **Center**: Weights from the *noise-integrating populations* to the *kick-neurons* (sorted). Each population excites one kick-neuron triplet while inhibiting the other, implementing selective gating of transition direction. **Right**: Recurrent weights within and across the two *noise-integrating populations*, showing strong within-population excitation and cross-population inhibition. Across all architectures, this structured connectivity forms a *kick-circuit*: two self-exciting, mutually inhibiting loops that project to opposing *kick-neurons*, enabling noise-driven, direction-selective transitions between cluster regions.

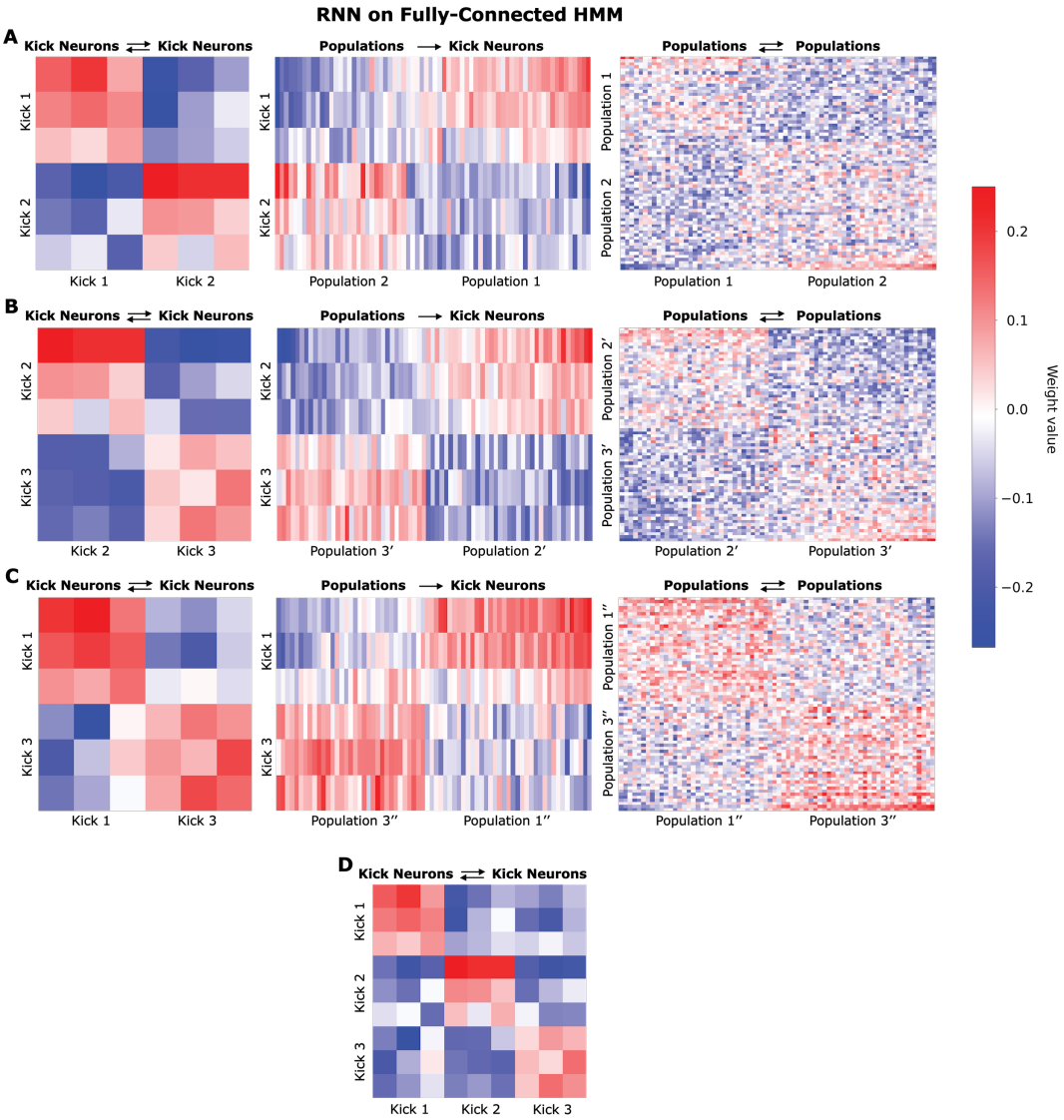

Figure 24: **Recurrent weight matrices and *Kick-circuits* for fully-connected HMMs.** Each row (**A–C**) corresponds to one of the three principal subspaces identified in the RNN trained to reproduce the emission statistics of a *fully-connected* HMM. Within each subspace, we isolate the *kick-neurons* and associated *noise-integrating populations* that together implement a self-contained instance of the dynamical primitive described for the *linear-chain* architectures. **Left**: sub-matrices of the recurrent weights W_{hh} restricted to the identified *kick-neurons*. Within-group connections are predominantly excitatory (red), while cross-group connections are inhibitory (blue), indicating mutual excitation and reciprocal inhibition between opposing kick groups. **Center**: weights from the two *noise-integrating populations* to the corresponding *kick-neurons* (sorted). Each population excites one kick group while inhibiting the other, implementing selective gating of transition direction. **Right**: recurrent weights within and across *noise-integrating populations*, showing strong within-population excitation and cross-population inhibition, mirroring the architecture observed in *linear-chain* models. Together, these subspaces (**A–C**) represent three instances of the same *kick-circuit* motif, confirming that the RNN decomposes the fully-connected HMM into compositional dynamical primitives that each generate noise-driven, direction-selective transitions. The notation $\mathrm{population}'$ (panel **B**) and $\mathrm{population}''$ (panel **C**) indicates that population groups are re-identified independently for each subspace, whereas the *kick-groups* remain fixed across panels. **D**: at the higher compositional level, the recurrent weights among the three *kick-groups* show the same organization — self-excitation within each group and cross-inhibition between groups — revealing that the same circuit principle recurs hierarchically across levels, from single-neuron motifs to multi-orbit compositions.

L Alignment of Output Dimensions with Orbits Plane

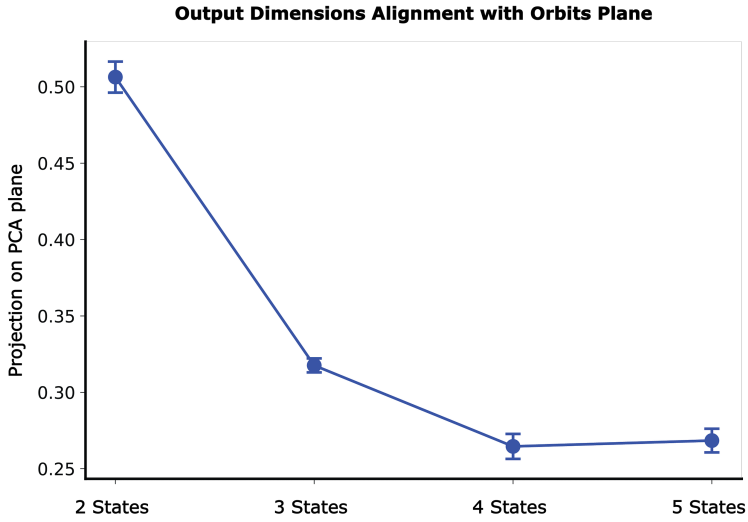

Figure 25: **Alignment of output dimensions with orbit plane.** Mean $\pm$ s.d. projection of the three output readout axes onto the principal plane spanned by the orbital dynamics, computed across three independently trained RNNs ($|h| = 150$, $d = 100$) for each *linear-chain* HMM with $M \in \{2, 3, 4, 5\}$ latent states. As the number of latent states increases, the alignment between the readout axes and the orbit plane systematically decreases, indicating that the networks progressively encode output probabilities through subtler modulations within a shared low-dimensional dynamical manifold rather than by forming additional slow regions or distinct attractors. This geometric reorientation provides a mechanistic account of how RNNs approximate increasingly fine latent discretizations — transitioning from discrete to quasi-continuous regimes — by reusing the same dynamical primitive while generating smoother, less skewed output distributions.

M Limitations and Broader Impact

M.1 Discussion of Limitations

This work focuses on RNNs trained to emulate relatively simple HMM structures. Despite the diversity of HMM families examined (*linear-chain, fully-connected* and *cyclic*), the target models share symmetric transition graphs and emission probabilities constrained to two or three output classes. While this design allows for clear interpretation and mechanistic analysis, natural behaviors are often governed by richer latent structures featuring asymmetric transitions and more diverse emission profiles. Extending the present framework to capture these more complex dynamics remains an important direction for future work. Nevertheless, preliminary analyses on naturalistic animal behavior suggest that the core dynamical primitive and its underlying mechanisms identified here generalize beyond the simplified settings considered, providing a potential foundation for modeling spontaneous behavioral sequences in biological systems.

M.2 Broader Impact

This work aims to advance our understanding of how Recurrent Neural Networks (RNNs) can represent discrete, stochastic latent structure through continuous dynamics. By reverse-engineering trained RNNs, we uncover a mechanistic account of how such representations can emerge in the absence of explicit structural constraints. This contributes to a growing body of work exploring how neural architectures can serve as data-driven models of behavior.

A key implication is that RNNs may offer a powerful alternative to traditional models like Hidden Markov Models (HMMs) for inferring latent structure in naturalistic behaviors. Unlike HMMs, which impose strong assumptions about the topology of the latent state space, RNNs allow for both discrete and continuous structures to emerge directly from the data. This flexibility has the potential to yield more biologically plausible representations of the computations underlying behavior.

More broadly, the dynamical mechanisms we identify — particularly how structured connectivity and noise interactions give rise to stochastic, quasi-periodic transitions — may inspire novel hypotheses about the neural representations of spontaneous, sequential behaviors in biological systems.

We do not foresee any direct negative societal impacts arising from this work.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction in this paper accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations in the Appendix M.1.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All the results in this paper can be reproduced.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specify all the training and test details necessary to understand the results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments in this paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper provide sufficient information on the computer resources needed to reproduce the experiments in B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discuss both potential positive societal impacts and negative societal impacts of the work performed in Appendix M.2.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The creators or original owners of assets used in the paper are properly credited and the license and terms of use are explicitly mentioned and properly respected.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.