# Learning Dynamics and the Geometry of Neural Dynamics in Recurrent Neural Controllers

**Ann Huang**
annhuang@g.harvard.edu
Harvard University

**Satpreet H. Singh**
satpreet_singh@hms.harvard.edu
Harvard Medical School

**Kanaka Rajan**
kanaka_rajan@hms.harvard.edu
Harvard Medical School

## Abstract

Recurrent Neural Networks (RNNs) are versatile and widely used models of complex decision-making behavior across many fields including artificial intelligence (AI) and neuroscience. Understanding how RNNs learn to perform complex tasks through interaction with an environment, i.e., as agents or controllers, is therefore broadly important. A lot of previous work has analyzed RNNs trained using supervised learning, and relatively less attention has been paid to reinforcement learning (RL) in the context of recurrent architectures and their learning dynamics. Here, we take a step towards addressing this gap by thoroughly analyzing the learning dynamics of RNN-based artificial agents trained by reinforcement to solve a classic nonlinear continuous control problem—the Inverted Pendulum. Our analysis found that training gradually sharpened the policy landscape and pruned the recurrent dynamics into a ring to efficiently represent the angle between the pendulum and its goal location, a circular variable. During training, a stable fixed point (FP) emerged and moved across the state space until it approached the goal location. Furthermore, the FP's proximity to the goal location was significantly correlated with the reward obtained by the controller, providing a direct link between the RNN's representational geometry and the agent's task performance. The memory capacity of the agent, quantified by its stimulus integration time, exhibited distinct regimes in its evolution over training. Our framework provides key intuitions on the evolution of the control policy, neural dynamics, representational geometry, and memory in RNN-based agents. In future work, we will extend this framework to investigate more complex environments, with longer evidence integration memory requirements, and more complex sequential decision making and planning.

## 1 Introduction

Recurrent Neural Networks (RNNs) are versatile and widely used models of neural activity and behavior in Neuroscience (Rajan et al., 2016; Barak, 2017). Most current research has focused on analyzing the internal dynamics in fully trained networks (Rajan & Abbott, 2006; Vyas et al., 2020). However, how such dynamics emerge through the process of learning and how changes in learning dynamics affect task performance are relatively less well explored. In recent work (Marschall & Savin, 2023; Hocker et al., 2024; Driscoll et al., 2022; Turner & Barak, 2024), early progress has been made towards understanding how the attractor landscape of RNNs changes during learning in the *supervised learning* setting. Here we consider the less studied setting of *reinforcement learning* (RL) to train RNN *agents* or *controllers*, where the interaction between an artificial agent and an external environment shapes its internal dynamics and behavioral policy throughout learning. We provide a thorough analysis of the inverted pendulum problem as a first step, developing a flexible analysis framework that can be applied to probe the learning dynamics and resulting behaviors of neural network-based agents in more complex or ethologically relevant RL tasks.

## 2 Methods

### 2.1 Agent and training

We trained RNN agents to balance an Inverted Pendulum in the upright position by applying appropriate torque (Fig. 1) in the OpenAI Gym Pendulum environment (Brockman et al., 2016) using policy gradients (Ni et al., 2021). Agents consisted of vanilla RNNs followed by 1-layer feedforward policy and value networks (all 64 units wide, *tanh* nonlinearity).

We considered both partially observable (PO) agents which only received the angular position $\theta$ as inputs, and fully observable (FO) agents, which additionally received the angular velocity $\dot{\theta}$. Training was performed for 100 gradient updates, with 1024 simulation steps per update.

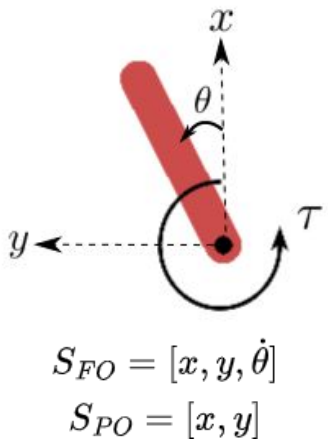


Figure 1: **Schematic representation of the Pendulum**, with different state spaces in fully observable (FO) and partially observable (PO) environments.

### 2.2 Policy landscape analysis

To understand how the controller's policy evolved during training, we visualized the policy in relation to the state variables $(\theta, \dot{\theta})$. At each training checkpoint, we froze the network weights and systematically swept through the observation space, assessing the agent's expected actions for all possible values of the state variables.

### 2.3 Fixed point analysis

The fixed points (FPs) in an RNN provides a mechanistic account of how the network implements a particular computation. We therefore performed fixed point analysis on the network activity of the RNN to shed light on the agent's internal dynamics. The update equation for the RNN is

$$\mathbf{h}_t = F(\mathbf{h}_{t-1}, \mathbf{x}_t) = \tanh\left(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{W}_x \mathbf{x}_t + \mathbf{b}\right),$$

where $\mathbf{h}_t$ is its hidden state, $\mathbf{W}_h$ the recurrent weight matrix, $\mathbf{x}_t$ the input vector, $\mathbf{W}_x$ the input-to-hidden weight matrix, and $\mathbf{b}$ a bias term (Sussillo & Barak, 2013). We define a fixed point in the hidden state space as $h^* = F(h^*, x)$, which can be relaxed to $h^* \approx F(h^*, x)$ to also include slow points. Here, we primarily focus on the network's fixed points when the input $x$ is zero and find the fixed points by minimizing the kinetic energy $\min_h \frac{1}{2}\|h - F(h, x = 0)\|_2^2$.

### 2.4 Stimulus integration time

Linearizing the RNN's update equation around an expansion point $(\mathbf{h}^e, \mathbf{x}^e)$, we obtain a linear dynamical system approximation:

$$\mathbf{h}_t \approx F\left(\mathbf{h}^e, \mathbf{x}^e\right) + \mathbf{J}^{\text{rec}}\big|_{(\mathbf{h}^e, \mathbf{x}^e)} \Delta\mathbf{h}_{t-1} + \mathbf{J}^{\text{inp}}\big|_{(\mathbf{h}^e, \mathbf{x}^e)} \Delta\mathbf{x}_t$$

where $\Delta\mathbf{h}_{t-1} = \mathbf{h}_{t-1} - \mathbf{h}^e$ represents the linearized system's state, $\Delta\mathbf{x}_t = \mathbf{x}_t - \mathbf{x}^e$ denotes the input, $\mathbf{J}^{\text{rec}}$ is the recurrence Jacobian matrix, and $\mathbf{J}^{\text{inp}}$ is the input Jacobian matrix. $\mathbf{J}^{\text{rec}}\big|_{(0,0)} = \mathbf{W_h}$ and $\mathbf{J}^{\text{inp}}\big|_{(0,0)} = \mathbf{W_x}$.

Previous studies have investigated the eigenvalues and eigenvectors of the recurrence matrix and recurrence Jacobian to understand the impact of connectivity on network dynamics (Rajan & Abbott, 2006; Singh et al., 2023). In particular, (Maheswaranathan et al., 2019) derive the stimulus integration timescale $\tau_i$ for a stable eigenvalue $\lambda_i$ (i.e., $|\lambda_i| \leq 1$) by considering the discrete-time

iteration $h_i(t) = \lambda_i^t h_i(0)$, which governs stimulus integration along the direction of the eigenvector $v_i$ corresponding to $\lambda_i$ and then compare this with the equivalent continuous-time equation $h_i(t) = h_i(0)e^{-t/\tau_i}$ to get $\tau_i = |(1/\ln|\lambda_i|)|$. We use 1000 timesteps (5 episodes) to generate estimates of $\tau_i$.

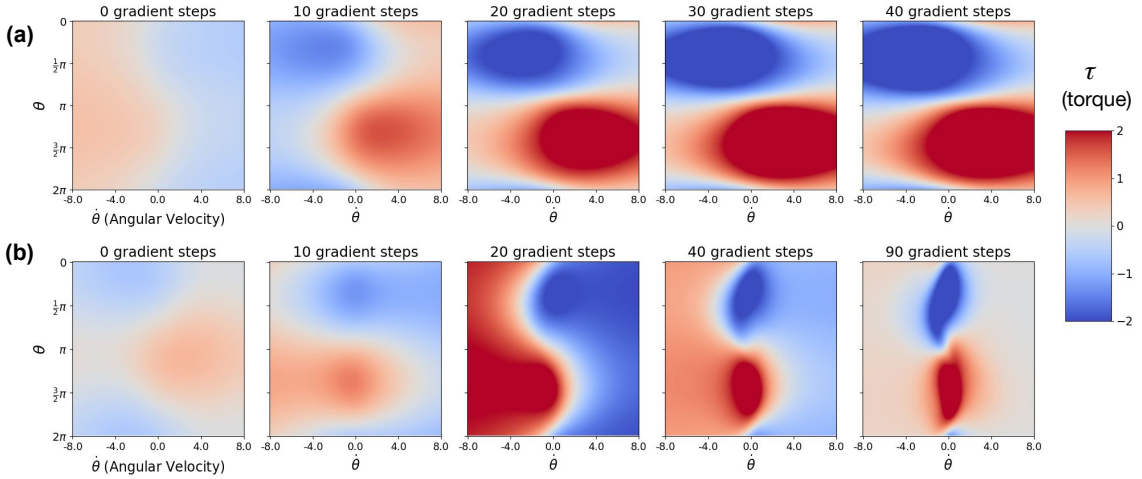


Figure 2: **Policy evolution of the recurrent neural controller**, visualized as a function of $\theta$ and $\dot{\theta}$. (a) Policy evolution in the PO environment; (b) the FO environment. Training gradually sharpens the policy decision boundaries between positive and negative torques, with sharper boundaries observed for the FO environment.

## 3 Results

### 3.1 Policy landscape analysis

We visualized the policy as a function of the state variables $(\theta, \dot{\theta})$ for both the FO and PO environment at different time points during training (Fig. 2). For both PO and FO environments, **training sharpens the policy landscape**, distinguishing between positive and negative torques more clearly. In the FO environment, the controller seems to use the extra $\dot{\theta}$ information to learn a **more precise** control policy, as indicated by the near-zero torques at high $\dot{\theta}$ values.

### 3.2 Geometry of neural dynamics and fixed point analysis

To investigate the internal dynamics of the controller, we then performed both dimensionality reduction and fixed point analysis on the network activities during the test episodes at different points during training (Fig. 3). In the PO case, training prunes recurrent dynamics into a ring to efficiently represent the circular state variable $\theta$. During training, the stable FP moves across the state space, gradually approaching the goal location where $\theta = 0$ (or equivalently, $\theta = 2\pi$). Meanwhile, an unstable FP, representing the free-hanging pendulum, emerges later in the training process and converges to the coordinates corresponding to the vertically-down pendulum position. Together, these FPs help the agent represent both its goal and key aspects of environmental physics within its recurrent dynamics. *Note that the controller's stable FP corresponds to the unstable FP of the physical system, and vice versa.*

Next, we ask if the movement of the stable FP toward the goal location is correlated with the agent's improved policy during learning. To shed light on this question, we superimposed the location of the stable FP across training and colored it by the average episode reward obtained by the corresponding policy (Fig. 4). Interestingly, how close the FP is to its goal location seems to be correlated with reward. We quantify this trend over five seeds to find that indeed **the proximity of the FP to its**

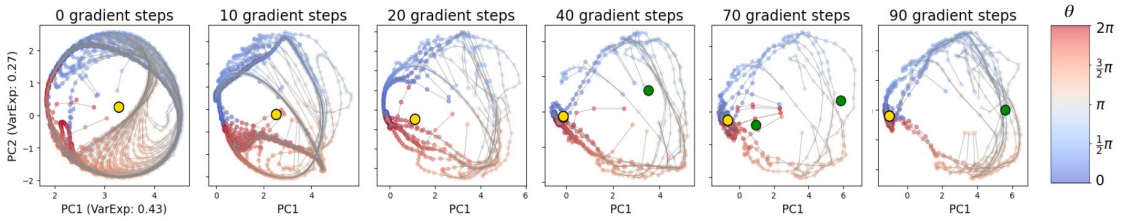


Figure 3: **Recurrent dynamics in the PO environment**, shown in the top two principal components, colored by $\theta$ at different training points. Training prunes the recurrent dynamics into a ring to efficiently represent the circular state variable $\theta$. During training, a stable fixed point (FP) colored in yellow, associated with the upright pendulum, emerges earlier, followed by the appearance of an unstable FP for the free-hanging pendulum state, colored in green.

**final state is correlated with the episode reward-to-go** (the difference between final reward and episode reward at the corresponding time point. This demonstrates a statistically significant linear relationship between reward-to-go and FP-goal location proximity, directly linking the RNN's representational geometry to the agent's behavioral performance. Data is pooled across five seeds to verify the robustness of this relationship.

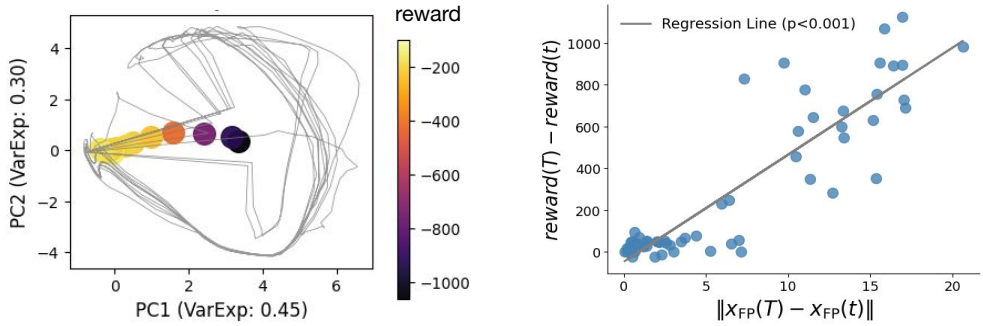


Figure 4: **Trajectory of the stable FP moving across the state space in the PO environment**, colored by 10-episode-average reward at the corresponding time point (left panel). The stable FP's proximity to the goal location is significantly correlated with the reward obtained by the controller (right panel).

### 3.3 Stimulus integration time

To understand the memory capacity of the agent, we plotted the evolution of top-5 stimulus integration times $\tau_i$ over training steps in the PO environment. (Fig. 5). Initially, we see an increase in $\tau$s, suggesting that the agent is extending its period of information integration to gather more stimulus information. This increase is followed by a decrease in $\tau$s, indicating a transition to more efficient memory usage that retains only task-relevant information in the shorter timescale.

This pattern, which we call "**explore then compress**", reflects the balance between thorough information integration during the exploration phase and a more focused, efficient retention of only task-relevant information in the compression phase.

## 4 Discussions

Our study presents a thorough analysis on the evolution of the control policy, network dynamics, and representational geometry of Recurrent Neural Controllers in the classic Inverted-Pendulum environment.

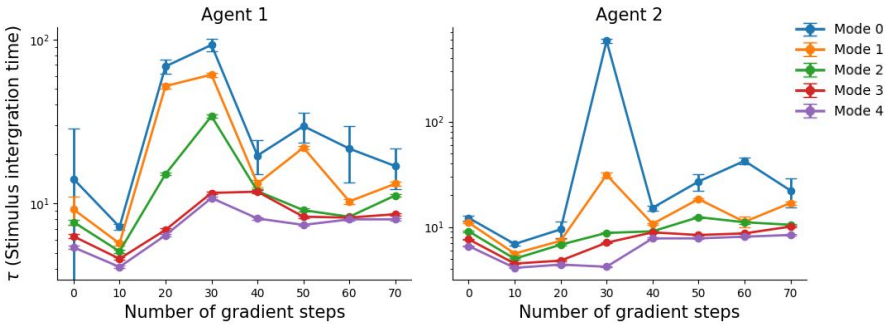


Figure 5: **Evolution of top-5 stimulus integration times $\tau_i$ over training steps in the PO environment.** The stimulus integration timescale $\tau$, calculated at various training stages in the PO environment. Stimulus integration exhibits distinct regimes in its evolution. The error bar indicates 99% confidence interval in the estimation of $\tau$.

We applied techniques from Dynamical Systems theory, such as fixed point analysis, that are popular in the Computational Neuroscience community (Vyas et al., 2020). In the future, we hope to further cross-pollinate interpretability techniques being developed in Computer Science and Computational Neuroscience to further analyze agent behavior and neural-activity.

We also plan to investigate more complex and biologically inspired environments, with longer evidence integration memory requirements, and complex sequential decision making and planning, to test how well our aforementioned findings generalize.

## Broader Impact Statement

Our research contributes to a deeper understanding of neural processes in artifical systems across Computer Science and mathematical models in Neuroscience. We hope this will lead to the development of both more reliable artificial control systems and better theories of brain function.

## References

Omri Barak. Recurrent neural networks as versatile tools of neuroscience research. *Current opinion in neurobiology*, 46:1–6, 2017.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI gym. *arXiv preprint arXiv:1606.01540*, 2016.

Laura Driscoll, Krishna Shenoy, and David Sussillo. Flexible multitask computation in recurrent networks utilizes shared dynamical motifs. *bioRxiv*, pp. 2022–08, 2022.

David Lance Hocker, Christine M. Constantinople, and Cristina Savin. Curriculum learning inspired by behavioral shaping trains neural networks to adopt animal-like decision making strategies, January 2024.

Niru Maheswaranathan, Alex Williams, Matthew Golub, Surya Ganguli, and David Sussillo. Reverse engineering recurrent networks for sentiment classification reveals line attractor dynamics. In *Advances in Neural Information Processing Systems*, pp. 15696–15705, 2019.

Owen Marschall and Cristina Savin. Probing learning through the lens of changes in circuit dynamics. Preprint, Neuroscience, September 2023.

Tianwei Ni, Benjamin Eysenbach, and Ruslan Salakhutdinov. Recurrent model-free rl is a strong baseline for many POMDPs. *arXiv preprint arXiv:2110.05038*, 2021.

Kanaka Rajan and Larry F Abbott. Eigenvalue spectra of random matrices for neural networks. *Physical Review Letters*, 97(18):188104, 2006.

Kanaka Rajan, Christopher D Harvey, and David W Tank. Recurrent network models of sequence generation and memory. *Neuron*, 90(1):128–142, 2016.

Satpreet H Singh, Floris van Breugel, Rajesh PN Rao, and Bingni W Brunton. Emergent behaviour and neural dynamics in artificial agents tracking odour plumes. *Nature Machine Intelligence*, 5 (1):58–70, 2023.

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.

Elia Turner and Omri Barak. The simplicity bias in multi-task rnns: Shared attractors, reuse of dynamics, and geometric representation. *Advances in Neural Information Processing Systems*, 36, 2024.

Saurabh Vyas, Matthew D Golub, David Sussillo, and Krishna V Shenoy. Computation through neural population dynamics. *Annual Review of Neuroscience*, 43:249–275, 2020.