Exploration Behavior of Untrained Policies

author names withheld

Under Review for the Workshop on High-dimensional Learning Dynamics, 2025

Abstract

Exploration remains a fundamental challenge in reinforcement learning (RL), particularly in environments with sparse or adversarial reward structures. In this work, we study how the architecture of deep neural policies implicitly shapes exploration before training. We theoretically and empirically demonstrate strategies for generating ballistic or diffusive trajectories from untrained policies in a toy model. Using the theory of infinite-width networks and a continuous-time limit, we show that untrained policies return correlated actions and result in non-trivial state-visitation distributions. We discuss the distributions of the corresponding trajectories for a standard architecture, revealing insights into inductive biases for tackling exploration. Our results establish a theoretical and experimental framework for using policy initialization as a design tool to understand exploration behavior in early training.

1. Introduction

Effective exploration is crucial for reinforcement learning agents operating in high-dimensional or sparse-reward environments. While much focus has been placed on designing explicit exploration bonuses or strategies, we shift attention to a more implicit source of exploration: the architecture of the policy network. Importantly, studying the effect of policy architecture does not necessitate the training of additional networks [3], or the introduction of exploration-dependent rewards [7, 12].

Untrained policies are responsible for determining the initial data distribution from which deep RL agents begin training. We hypothesize that the choice of architecture and initialization distribution implicitly determines the entropy and geometry of exploration in state space. As a step toward understanding this phenomenon, we provide initial theoretical and experimental results to support this hypothesis. We leverage the infinite-width theory and a continuous-time limit to understand random policy behavior in a toy model with corresponding simulations.

Contributions: Our results are as follows: we first show that fixed policies lead to ballistic trajectories (where correlations and smoothness in the network dominate), before demonstrating that random policy initializations at each step can be used to generate diffusive trajectories with heavy-tailed steady-state distributions. Lastly, we show that combining these methods can lead to interesting effects, providing new pathways for structured exploration in deep RL.

Related Work: The exploration problem in RL has been studied from many angles [5], but combining with analytically tractable models of deep RL appears to be novel. The infinite-width limit [4, 9] and analysis of the Fokker-Planck equation, which characterize our results have also proven to be fruitful avenues of research. Our work is also somewhat related to the vast research on neural architecture search (NAS) which attempts to optimize policy architecture directly [15],

e.g. for better exploration strategies, which we instead approach through the lens of closed-form solutions.

2. Background

2.1. Reinforcement Learning

As we focus on the effects of untrained policies, we only need the basic and usual definitions: a state space, $\mathcal{S}$, action space $\mathcal{A}$, deterministic transition dynamics (a function mapping state and action to successor-state) $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. Let $\pi_\theta : \mathcal{S} \to \mathcal{A}$ be a policy represented by a feedforward neural network with parameters $\theta \in \mathbb{R}^n$ randomly initialized according to some specified scheme (e.g. Xavier-Glorot). We will not introduce any (intrinsic or exploration-dependent) reward functions and instead focus on the reward-free Markov processes that govern the agent's data collection processes.

Operating in the policy-based RL setting, the network π_θ (whose initialization effects we study) is trained to maximize returns (e.g. REINFORCE [14], TRPO [10], PPO [11]). Value-based algorithms related to DQN may yield similar results, but the additional non-linear mapping from value to policy space can complicate the analysis, and is left to future work.

2.2. Smooth Networks Yield Ballistic Trajectories

For short times, sufficiently smooth neural networks can induce trajectories with a dominating drift term: "ballistic motion". Loosely speaking, the structure of a neural net induces similar outputs (actions) for nearby states, and when compounded with smooth dynamics, this results in agents with very smooth (non-diffuse) trajectories. This result is visualized in Figures 1 and 2 and is formulated precisely below.

Assumption 1 (Transition Dynamics Locality) *The environment dynamics are "local": there exists a $\delta > 0$ such that for all $s \in \mathcal{S}$ and successor states $s' = f(s, \cdot)$ the gap between two successive states is upper-bounded $|s' - s| < \delta$.*

Lemma 1 (Short-Time Ballistic Behavior of Lipschitz Neural Policies) *Let $\pi_\theta : \mathbb{R}^d \to \mathbb{R}^d$ be a stochastic policy sampled from a neural network prior, such that the realized sample function is L_π-Lipschitz. Let the dynamics satisfy Assumption 1 for some $\delta > 0$, and define the deterministic linear dynamics:*

$$s_{t+1} = s_t + \pi_\theta(s_t).$$

Let the constant $c = \pi_\theta(s_0)$ denote the initial action. Then for any initial state s_0 the agent's trajectory approximately follows the straight-line path ct, up to a bounded error:

$$\|s_t - ct\| \leq \frac{1}{2}\delta L_\pi t^2.$$

Remark 2 *Although the previous lemma holds for all times, a linear approximation holds only for short timescales, $t \ll \frac{1}{\delta L_\pi}$, beyond which, deviations caused by curvature (resulting from non-linearities in the deep neural net) become large enough to significantly affect the effective mean velocity, c.*

The Lipschitz constant of a neural policy can be computed [1, 6, 13] or roughly bounded *a priori* to make Lemma 1 implementable. Although the assumptions and structure of the above dynamics appear limiting, they provide a starting point to develop intuition for exploration behaviors. We will focus on generalizing these results in future work. Since the same neural network is often used at the beginning of training (e.g. to fill up a replay buffer), a (frozen) policy will not supply diverse trajectories. As an alternative, we will now consider initializing a *new* policy (from a fixed distribution, say) and study its effects on the agent's trajectories.

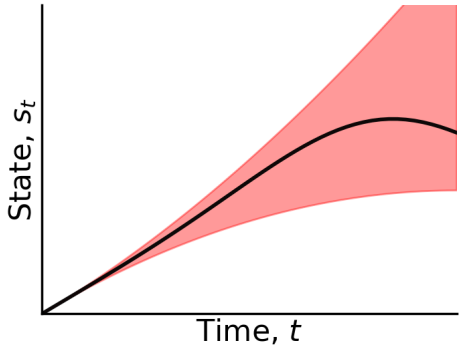

Figure 1: Though trajectories can change direction (as evidence in the plot on the right), on short timescales, the trajectories can be well-approximated with linear drift.

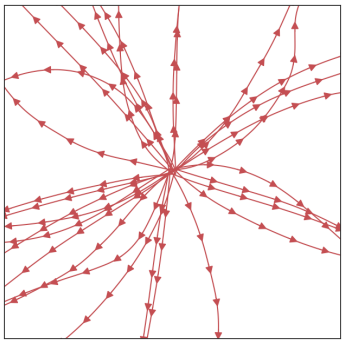

Figure 2: Standard MLPs used for deep RL (with two hidden layers, 256 hidden nodes, ReLU activation) produce ballistic trajectories.

2.3. Random Policy Networks as Gaussian Processes

Rather than a single policy generating an entire trajectory (as is typically the case), we seek to obtain a more diffusive (and hence potentially more exploratory) trajectory distribution. To this end, the control policy π_θ is now re-sampled sequentially:

Definition 3 (Trajectory Under a Random Policy Ensemble) *Let $\mathcal{T}_\theta = \{s_0, s_1, \ldots, s_T\}$ denote a trajectory generated by:*

$$s_{t+1} = f\left(s_t, \pi_{\theta_t}(s_t)\right), \quad \theta_t \sim \mathcal{P}_\theta$$

where each θ_t is independently drawn from the initialization distribution $\mathcal{P}_\theta$, and π_{θ_t} is the corresponding deterministic policy network.

With this setup, we can leverage the following closed-form expression for the policy in the infinite-width limit (cf. also Eq. 1 of [8] and surrounding discussion for more details):

Theorem 4 (Neal [9], Infinite-Width Limit as Gaussian Process) *Let π_θ be a feedforward neural network with nonlinear activation $\phi(0) = 0$ (e.g., ReLU or Tanh), and i.i.d. weights and biases, with zero mean. In the limit as the width of all hidden layers tends to infinity, the distribution over functions π_θ converges to a Gaussian Process:*

$$\pi_\theta \xrightarrow{d} \mathcal{GP}(0, K(s, s')),$$

where K is an architecture-dependent kernel.

The GP then produces an action covariance when a fresh policy is sampled at each timestep: let $s, s' \in \mathcal{S}$, then the covariance between actions at two states is given by: $\text{Cov}\left[\pi_\theta(s), \pi_\theta(s')\right] = K(s, s') \in \mathbb{R}^{d\times d}$. Thus, the kernel corresponding to the chosen architecture (and choice of parameter initialization) is responsible for giving structure to action correlations across state-space. Below, we show an example of this structure in a policy with one (infinitely wide) hidden layer and ReLU activation, for its simplicity and popularity in the literature.

Note that these effects are complementary to that of a single policy, whose architecture is typically Lipschitz, implying ballistic trajectories, as shown above.

2.4. Case Study: ReLU Networks and Induced Kernels

For ReLU networks with Gaussian weight initialization, the infinite-width kernel is given by:

$$K(s, s') = \sigma_w^2 \frac{1}{\pi}|s||s'|\left(\sin\theta + (\pi - \theta)\cos\theta\right) + \sigma_b^2, \tag{1}$$

where $\cos\theta = \frac{\langle s,s'\rangle}{|s||s'|}$. This kernel is not stationary but is radial and induces a diffusion term that grows with $|s|^2$. Thus, far from the origin, exploration becomes more diffuse. Stationary kernels (e.g. Radial Basis Functions [2]) depending on $|s - s'|$ on the other hand would result in constant diffusion coefficient, which may be of interest in some environments.

In the special case of the one hidden layer, infinitely wide ReLU, we take weights and biases drawn from centered Gaussians with variance σ_w, σ_b, respectively. In this case, the policy is mean zero and has diffusion coefficient $K(s, s) = \Sigma(s) = \sigma_b^2 + \frac{\sigma_w^2}{\pi}\|s\|^2$. For the simplest linear dynamics $s_{t+1} = s_t + \pi(s_t)$, the stochastic policy induces a random walk in state space that can be modeled by a Fokker-Planck equation:

$$\frac{\partial p}{\partial t} = -\mu_0^\top \nabla p + \frac{1}{2}\nabla^2 : (\Sigma(s)p), \tag{2}$$

which in the long-time limit (assuming stationarity) can be written as:

$$\nabla^2\left[\left(\sigma_b^2 + \frac{\sigma_w^2}{\pi}\|s\|^2\right)p_\infty(s)\right] = 0.$$

With radial symmetry $p_\infty(s) = f(r)$, $r = \|s\|$, we obtain the solution:

$$p_\infty(s) \propto \left(\sigma_b^2 + \frac{\sigma_w^2}{\pi}\|s\|^2\right)^{-d/2}. \tag{3}$$

This describes a heavy-tailed stationary distribution (resembling Cauchy or power-law distributions) and is normalizable if $\sigma_b > 0$. Note that as $\sigma_b^2 \to 0$, the tails become increasingly heavy, and exploration is more likely to reach distant regions of state space.

2.5. Experiments

Exploration behavior depends strongly on the nature of the policy architecture. As illustrated in Figure 3, placing a barrier in state space with a narrow hallway drastically changes the likelihood of successful exploration. Ballistic trajectories, generated by a fixed MLP (red), often do not pass through the hallway: they follow straight paths, and even small errors result in failed exploration. In contrast, trajectories from networks re-initialized at every step (green) exhibit diffusive,

fat-tailed distributions that may slowly explore across the barrier. A hybrid switching strategy, where a fixed MLP is used for the first n steps before switching to per-step re-initialization, capturing both initial momentum and later diffusion, enabling more effective exploration in this challenging scenario. Choosing a value of $n \ll (\delta L_\pi)^{-1}$ (cf. Lemma 1) ensures that trajectories will linearly escape the initial state, while $n > 2\Delta/\delta$ can ensure trajectories reach states of interest (a distance Δ from the origin). Studying this trade-off in more complex environments is the focus of future work.

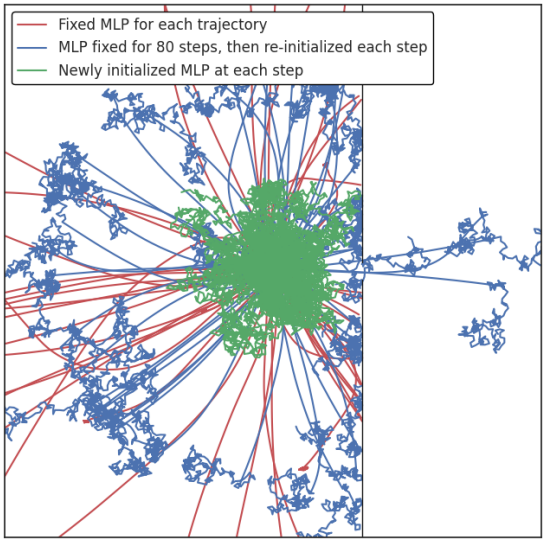

Figure 3: **Exploration through a narrow hallway.** Ballistic trajectories from a fixed MLP struggle to pass through the barrier; stepwise re-initialization produces overly-diffusive motion, while a hybrid strategy leverages both behaviors for efficient exploration.

3. Discussion

In this work, we explored how the architecture and initialization of policy networks can shape the exploration dynamics of reinforcement learning agents. By modeling untrained policies as draws from Gaussian Processes (GPs) in the infinite-width limit, we demonstrated that distinct neural architectures and parameterizations implicitly induce nontrivial priors over agent behavior. For simple dynamics models, these behaviors are analytically tractable, giving insight into the exploration distribution. These priors manifest themselves as specific patterns in trajectory geometry, ranging from ballistic drift to heavy-tailed diffusive exploration.

Our analysis showed that Lipschitz continuity in deterministic policy networks leads to smooth, directional (ballistic) trajectories which may limit early exploration. In contrast, policies sampled at each timestep from an architectural prior induce highly stochastic (but structured) trajectory distributions. We studied these trajectories in the continuous-time limit, employing a Fokker–Planck equation, revealing a connection between a network's kernel and steady-state distribution. We found that ReLU-based policies generate quasi-Cauchy state distributions in free space, encouraging broad exploration. Notably, the magnitude and shape of the induced kernel dictate the correlation between actions at different states, determining whether trajectories are locally consistent or rapidly de-correlating, when using the per-step policy sampling scheme.

These findings suggest a new lens for understanding exploration. Rather than a process to be tuned through learning or incentivized via external bonuses, one can envisage exploration as an architectural and initialization design problem. By tailoring policy architectures to match the topology or reward sparsity of a target environment, one can influence early-stage exploration "zero-shot". This discussion aligns with a broader incentive to align network architectures with environments, an area in deep reinforcement learning that has not yet been explored.

References

[1] Aritra Bhowmick, Meenakshi D'Souza, and G Srinivasa Raghavan. Lipbab: Computing exact lipschitz constant of relu networks. In *Artificial Neural Networks and Machine Learning–ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part IV 30*, pages 151–162. Springer, 2021.

[2] Martin Dietrich Buhmann. Radial basis functions. *Acta numerica*, 9:1–38, 2000.

[3] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[4] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[5] Pawel Ladosz, Lilian Weng, Minwoo Kim, and Hyondong Oh. Exploration in deep reinforcement learning: A survey. *Information Fusion*, 85:1–22, 2022.

[6] Fabian Latorre, Paul Rolland, and Volkan Cevher. Lipschitz constant estimation of neural networks via sparse polynomial optimization. *arXiv preprint arXiv:2004.08688*, 2020.

[7] Sam Lobel, Akhil Bagaria, and George Konidaris. Flipping coins to estimate pseudocounts for exploration in reinforcement learning. In *International Conference on Machine Learning*, pages 22594–22613. PMLR, 2023.

[8] Lassi Meronen, Martin Trapp, and Arno Solin. Periodic activation functions induce stationarity. *Advances in Neural Information Processing Systems*, 34:1673–1685, 2021.

[9] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[10] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[11] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[12] Adrien Ali Taiga, William Fedus, Marlos C Machado, Aaron Courville, and Marc G Bellemare. On bonus-based exploration methods in the arcade learning environment. *arXiv preprint arXiv:2109.11052*, 2021.

[13] Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. *Advances in Neural Information Processing Systems*, 31, 2018.

[14] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

[15] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

Appendix

We first prove a useful inequality for obtaining our main result, Lemma 1.

Lemma 5 *The sequential inequality*

$$\epsilon_{t+1} \leq \epsilon_t + kt \tag{4}$$

has iterates upper bounded by $\epsilon_t \leq kt^2/2 + \epsilon_0$.

Proof The proof is straightforward by induction. For the base case, note that $\epsilon_0 \leq k(0)^2/2 + \epsilon_0$. From the inductive hypothesis,

$$\begin{aligned}
\epsilon_{t+1} &\leq \epsilon_t + kt \\
&\leq \frac{kt^2}{2} + \epsilon_0 + kt \\
&= \frac{k(t^2+2t)}{2} + \epsilon_0 \\
&= \frac{k(t+1)^2 - k}{2} + \epsilon_0 \\
&= \frac{k(t+1)^2}{2} + \epsilon_0 - \frac{k}{2} \\
&\leq \frac{k(t+1)^2}{2} + \epsilon_0
\end{aligned}$$

which is the form of iterate $(t+1)$, thus completing the proof. ■

We now prove Lemma 1:

Proof First, denote $\widetilde{s}_t = ct + s_0$, the linear approximation of the trajectory. We examine the iterates of the error, $\epsilon_t \doteq |s_t - \widetilde{s}_t|$ to verify the bound presented in the lemma.

$$\begin{aligned}
\epsilon_{t+1} &= |s_{t+1} - \widetilde{s}_{t+1}| && (5)\\
&= |s_{t+1} - \widetilde{s}_t - c| && (6)\\
&= |s_t + \pi_\theta(s_t) - \widetilde{s}_t - c| && (7)\\
&\leq |s_t - \widetilde{s}_t| + |\pi_\theta(s_t) - c| && (8)\\
&\leq \epsilon_t + |\pi_\theta(s_t) - c| && (9)\\
&\leq \epsilon_t + |\pi_\theta(s_t) - \pi_\theta(s_0)| && (10)\\
&\leq \epsilon_t + L_\pi |s_t - s_0| && (11)\\
&\leq \epsilon_t + \delta L_\pi t && (12)\\
& && (13)
\end{aligned}$$

Several notes are in order: In the last three lines we (a) made the substitution of $c = \pi_\theta(s_0)$ as an estimate of the mean velocity, (b) used the Lipschitz property of the neural network π_θ, and (c) used the locality of dynamics iteratively to bound the distance in state-space. For (c), we have iterated the bound in Assumption 1 for multiple timesteps, making use of the triangle inequality.

Note that for (a), this choice leads naturally to the desired result and is easy to compute (once the first action is drawn, the estimate of the drift is complete). However, it may not lead to the tightest bounds in practice. Empirically, we have found that using a mean of actions along the trajectory can smooth out the estimate for velocity, somewhat improving the bound. Similar steps can be taken to arrive at a linear upper bound, using that e.g. $|\pi_\theta(s_t) - \langle \pi_\theta(s_t) \rangle| < g\varepsilon$. where g is a geometric factor and ε is the diameter of a ball over which the average $\langle \pi_\theta(s_t) \rangle$ is computed. In any case, the resulting bound is linear.

Using Lemma 5, we see that $\epsilon_t \leq \delta L_\pi t^2/2$, neglecting the initial error term $\epsilon_0 = 0$. Ensuring the error does not accumulate past the linear prediction results in the upper bound on timesteps presented in the main text.

■