First-Explore PPO : Learning Meta-Exploration with Proximal Policy Optimization

Ben Norman, Ryan Sullivan, Jeff Clune

Keywords: Reinforcement Learning, Exploration, Meta Learning

Summary

Exploration is essential for effective learning, yet many learning methods rely on random action selection, manually designed heuristics, or priors from imitation learning. These strategies are often insufficient in complex environments with deceptive rewards, which can incentivize agents to learn suboptimal policies. A more general approach is to train agents to *learn how to explore*. First-Explore addresses this problem through meta-learning, but its original instantiation is sample-inefficient. We introduce First-Explore PPO (FE-PPO), a PPO-based implementation of the First-Explore meta-exploration objective. Across challenging meta-learning benchmarks with deceptive rewards, FE-PPO significantly outperforms RL^2, a strong meta-RL baseline, which fails entirely in one of these environments. FE-PPO also matches or exceeds the performance of the original First-Explore in every environment tested with $10 - 40\times$ fewer samples. These results show that First-Explore PPO is a practical and effective starting point for future work on meta-exploration. Our code is available at https://github.com/RyanNavillus/First-Explore-PPO

Contribution(s)

1. This paper introduces First-Explore Proximal Policy Optimization (FE-PPO), an instantiation of First-Explore that applies the First-Explore meta-exploration objective to PPO. We evaluate FE-PPO on three deceptive-reward meta-learning benchmarks and find that FE-PPO outperforms RL^2 on all three environments. FE-PPO even outperforms the original First-Explore on one of the three environments and matches performance on the others, while being dramatically more sample efficient.
   Context: Prior work introduced PPO (Schulman et al., 2017), First-Explore (Norman & Clune, 2024), and RL^2 (Duan et al., 2016). The original First-Explore method optimized trajectories using a cross-entropy-loss-based approach without a value function. FE-PPO instead adapts the high-level concepts of First-Explore to PPO; this combination is non-obvious, as we describe in the paper.
2. We introduce a stronger RL^2 baseline by tuning PPO-based RL^2 with a Transformer-XL architecture. This baseline outperforms the RL^2 results reported by Norman & Clune (2024), yielding a more competitive comparison for FE-PPO.
   Context: Norman & Clune (2024) evaluate PPO-based RL^2 with a RNN based architecture and report lower scores. Our tuned Transformer-XL implementation significantly improves this baseline while preserving the same core meta-RL objective.
3. We provide clean, open-source implementations of FE-PPO in JAX and CleanRL to support future research on meta-exploration. The codebase also includes the benchmarks used in this paper and a JAX implementation of a Transformer-XL PPO-based RL^2.
   Context: Open-source implementations of PPO, First-Explore, and RL^2 exist, but no open-source implementation of First-Explore with PPO previously existed.

First-Explore PPO : Learning Meta-Exploration with Proximal Policy Optimization

Ben Norman[1,2,†], **Ryan Sullivan**[1,2,†], **Jeff Clune**[1,2,3]
btnorman@cs.ubc.ca, ryan.sullivan@ubc.ca

[1]**Department of Computing Science, University of British Columbia, Canada**
[2]**Vector Institute**
[3]**CIFAR AI Chair**

[†] equal contribution

Abstract

Exploration is essential for effective learning, yet many learning methods rely on random action selection, manually designed heuristics, or priors from imitation learning. These strategies are often insufficient in complex environments with deceptive rewards, which can incentivize agents to learn suboptimal policies. A more general approach is to train agents to *learn how to explore*. First-Explore addresses this problem through meta-learning, but its original instantiation is sample-inefficient. We introduce First-Explore PPO (FE-PPO), a PPO-based implementation of the First-Explore meta-exploration objective. Across challenging meta-learning benchmarks with deceptive rewards, FE-PPO significantly outperforms RL^2, a strong meta-RL baseline, which fails entirely in one of these environments. FE-PPO also matches or exceeds the performance of the original First-Explore in every environment tested with $10 - 40\times$ fewer samples. These results show that First-Explore PPO is a practical and effective starting point for future work on meta-exploration. Our code is available at https://github.com/RyanNavillus/First-Explore-PPO

1 Introduction

In complex environments, discovering the highest-performing behaviors often requires substantial exploration. For example, agents may need to try novel strategies, navigate to unseen locations, or attempt new puzzle solutions. Each environment requires different methods of exploration, without which agents may never discover critical information necessary to enable the strongest behaviors. As a result, exploring through trial and error is a central challenge in online reinforcement learning.

Most methods use simple exploration, such as ϵ-greedy sampling (Sutton & Barto, 1998; Mnih et al., 2015; Osband et al., 2016) or entropy-regularized training (Schulman et al., 2015a; 2017; Haarnoja et al., 2018). Other approaches directly incentivize exploration with heuristic objectives (Sukhbaatar et al., 2017; Burda et al., 2018; Henaff et al., 2022). However, these approaches often fall short in hard-exploration domains (Matthews et al., 2024) or perform well only in the domains they were developed for (Taiga et al., 2021; Sullivan et al., 2024). Rather than hand-coding exploration, agents can learn exploratory behaviors from experience. This approach is promising because learned solutions can outperform hand-coded ones (Duan et al., 2016) and generalize across domains, rather than depending on domain expertise to specify a sensible exploration heuristic. Meta reinforcement learning (Duan et al., 2016; Team et al., 2023) trains agents to learn effective behaviors and, in principle, can learn exploratory behavior. However, these approaches are susceptible to deceptive rewards, such as penalties for moving or nearby positive but suboptimal reward sources (Norman &

Clune, 2024). In these environments, agents may converge to suboptimal strategies and fail to fully explore.

First-Explore addresses this challenge by directly training agents to collect information that improves later-episode performance (Norman & Clune, 2024). First-Explore separates behavior into exploration and exploitation modes. Agents receive no environment rewards during exploration episodes; instead, each is assigned the *return* from the subsequent exploitation episode in which the agent uses information gathered during prior exploration. Norman & Clune (2024) demonstrate that this dual-mode approach teaches agents to explore before exploiting rewards, enabling strong performance in environments where prior meta-learning methods fail. Despite these strong results, First-Explore is highly sample-inefficient. First-Explore does not train a value function, so credit assignment relies on high-variance realized returns rather than bootstrapped value estimates. This poor sample efficiency limits First-Explore's applicability to complex environments and real-world problems, where learning often requires many interactions.

We present First-Explore Proximal Policy Optimization (FE-PPO), which combines First-Explore's meta-learning objective with the Proximal Policy Optimization (PPO) algorithm. PPO trains a value function to bootstrap future returns from partial rollouts, allowing efficient optimization of long-horizon meta-learning objectives. We evaluate FE-PPO on the three deceptive-reward meta-learning domains introduced by Norman & Clune (2024) and show that it matches or outperforms First-Explore in all three environments. Importantly, FE-PPO does so using $10 - 40\times$ fewer environment samples. We release open-source implementations of FE-PPO, First-Explore, and other meta-learning baselines in both JAX (Bradbury et al., 2018) and PyTorch (Paszke et al., 2019) CleanRL (Huang et al., 2022). FE-PPO is a simple, efficient, and effective baseline for meta-exploration and will be a foundation for future research.

2 Background

2.1 Meta Reinforcement Learning

Meta-reinforcement learning studies agents trained over a distribution of trial-level POMDPs, $p(\mathcal{M})$, where each trial samples an environment instance $\mathcal{M}_i = (\mathcal{S}, \mathcal{A}, \mathcal{O}, P_i, R_i, \Omega_i)$. A trial consists of multiple episodes from the same sampled environment instance: the environment state is reset at the start of each episode, but the underlying transition, reward, and observation functions remain fixed throughout the trial. The objective is to maximize cumulative reward across the entire trial, $G_{\text{trial}} = \sum_{t=0}^{H-1} r_t$, where H is the total number of timesteps in the trial.

The agent's policy is implemented using a sequence model that conditions on the within-trial interaction history. At each timestep, the policy receives the current observation together with auxiliary inputs including the previous action and reward, allowing the sequence model to use past experience for adaptation throughout a trial.

In the environments studied in this paper, each trial has an initially unknown reward structure. To achieve high returns across the trial, agents must identify the reward structure during earlier episodes and use this experience to obtain more reward in later episodes of the same trial.

2.2 First-Explore

First-Explore separates agent behavior into *exploration* and *exploitation* modes (Norman & Clune, 2024). First-Explore maintains an exploration context c_i that accumulates information from previous exploration-mode rollouts, Figure 1. This context contains the actions selected by the policy, as well as observations and rewards encountered during exploration, and is provided to both the exploration and exploitation policies.

In exploration mode, the agent observes environment rewards, which may inform its belief state about the current environment instance, but the policy is not trained to optimize these rewards.

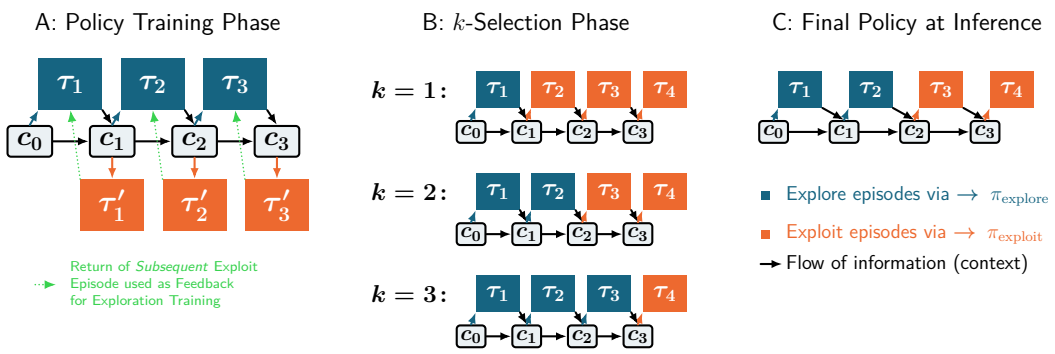

Figure 1: The three phases of First-Explore. **A:** Two modes are trained: one to explore, producing informative episodes, and one to exploit, maximizing current-episode return. τ_i is the ith exploration-mode episode and τ_i' the ith exploitation-mode episode. **B:** *After training*, we sweep over the number of exploration-mode episodes k to find the value that maximizes performance, $k_{\max}$. **C:** The final policy first performs $k_{\max}$ exploration episodes followed by exploitation episodes. In the diagram, $k_{\max} = 2$.

Instead, exploration-mode actions are scored using returns obtained during subsequent exploitation. This objective encourages the agent to gather information that is useful for exploitation, even when the rewards encountered during exploration are deceptive or locally suboptimal. First-Explore uses realized exploitation returns to train the exploration policy and does not learn a value function, so its policy-gradient estimates can have high variance. This makes the method sample-inefficient, motivating the bootstrapped value-estimation approach introduced in FE-PPO.

3 Method

3.1 First-Explore PPO

FE-PPO replaces First-Explore's Monte-Carlo gradient estimator with a bootstrapped PPO update (Schulman et al., 2017) computed jointly over exploration and exploitation. A single Transformer-XL network (Dai et al., 2019) with an actor head and a value head that is shared across both modes and trained with the standard PPO loss. We append a mode flag to each observation to allow the policy to behave differently in each mode. The augmented observation also contains the previous action (as a one-hot vector) and the previous reward, providing the policy and value function with the immediate history of each step.

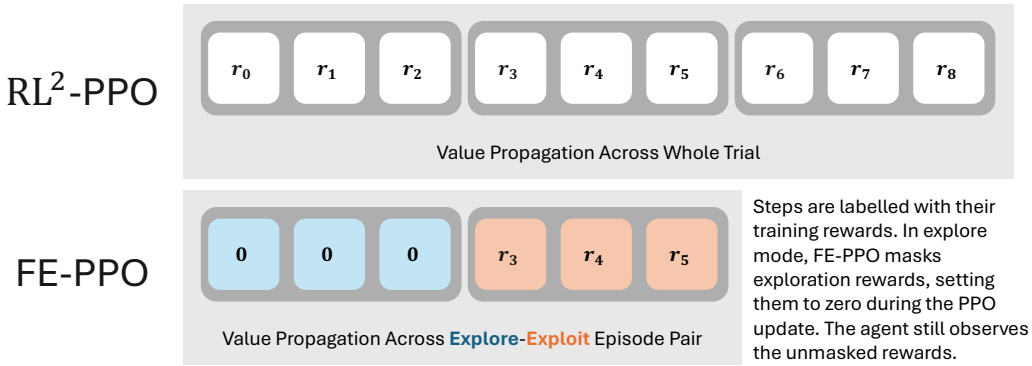

Figure 2: The value propagation used to train FE-PPO and RL^2-PPO.

To enable a critic to correctly value states in the exploration mode, we treat each explore–exploit pair as a single episode during the loss calculation. We mask the exploration episode termination signal such that PPO sees a terminal transition only at the end of exploitation episodes. Under this structure, Generalized Advantage Estimation (Schulman et al., 2015b) propagates value targets backwards from the exploitation episode into the preceding exploration episode without interruption. Exploration rewards are masked, so only the subsequent exploitation return bootstrapped through the value function contributes to an exploration step's advantage estimation. However, the agent observes the unmasked exploration rewards so they inform later actions. Masking the exploration rewards implements First-Explore's principle of training exploration actions according to their effect on later exploitation, and our introduction of a value function reduces the variance by replacing Monte Carlo estimates with bootstrapping from a learned critic.

Consistent with First-Explore, each episode (whether exploration-mode or exploitation-mode) is initialized with the context from the end of the previous exploration episode. After K explore–exploit pairs the environment is reset to a new task instance and the recurrent memory is reinitialized, completing one trial.

4 Experiments

We implement First-Explore using PureJaxRL (Lu et al., 2022) and based on the original open-source implementation. For each method, we perform a random hyperparameter search and train the final models using five independent random seeds for 500M timesteps. Training details are explained in Appendix D. Each trained policy is evaluated on $10{,}000$ independently sampled environments for 1 trial each, and we report the mean and standard deviation of the cumulative trial returns. As in Norman & Clune (2024) we evaluate the final FE-PPO models by sweeping over the number of exploration episodes for each run, and report scores for the highest performing number of exploration episodes. RL^2 scores are a simple cumulative sum of returns across a trial. For First-Explore, we use the results reported by Norman & Clune (2024), since retraining the method is computationally prohibitive; in Ray Maze, a single run takes approximately 75 hours.

4.1 Environments

We evaluate our approach on the deceptive reward meta-RL environments introduced by Norman & Clune (2024). Each trial begins by sampling an environment instance for the entire trial, so the agent must use its interaction history to infer the reward structure of that instance. In all three environments, effective exploration requires forgoing immediate reward maximization to enable higher returns during later exploitation.

Bandits with One Fixed Arm. The first environment is a ten-armed Gaussian bandit. The agent selects an arm $a \in \{1, \ldots, 10\}$ at each timestep. The first arm always gives the fixed reward $\mu_1 = 0.5$, while the remaining arms have instance-specific means $v_a \sim \mathcal{N}(0, 1)$. Pulling one of these arms produces reward $v_a + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \frac{1}{2})$. The arm means are fixed for the duration of the trial, but the noise term is resampled on each pull.

This environment is deceptive because arm 1 provides a reliable positive reward but reveals no information about the remaining arms. Since the other arms have a mean reward of zero across instances, sampling them for exploration has lower expected reward than repeatedly pulling arm 1. However, high trial return requires identifying an arm whose sampled mean exceeds μ_1, then exploiting that arm for the remainder of the trial.

Dark Treasure Rooms. The second environment is a 9×9 grid world containing hidden treasures and traps. The agent starts in the center of the grid and observes only its current (x, y) coordinate. It has five actions: move in one of the four cardinal directions or remain still. Each environment contains eight objects placed uniformly at random. When the agent reaches an object, the object is consumed for that episode and the agent receives its associated reward. Object reward values

are sampled uniformly from the range $[-4, 2]$, and overlapping objects have additive rewards. This environment is deceptive because visiting an unseen object has negative expected reward. The agent must nevertheless move blindly into previously unvisited coordinates to discover which locations contain positive rewards.

Ray Maze. The third environment is a partially observable navigation task in a randomly generated maze with impassable walls and three goal locations. The agent has three actions: turn left, turn right, or move forward. It observes the environment through 15 LiDAR-like rays, each of which reports the distance to the nearest wall along that ray, the wall orientation, and whether the ray intersects a goal location. These observations reveal where goals are, but not whether a goal is beneficial or harmful.

Each goal independently gives reward $+1$ with probability 0.3 and reward -1 otherwise. Only the first goal reached in an episode produces reward, preventing the agent from evaluating multiple goals within a single episode. Since an unknown goal has negative expected value, the agent must initially risk visiting goals in order to determine their rewards. High return then requires remembering which goals are positive and navigating back to them in later episodes, while avoiding both known negative goals and untested goals.

4.2 Results

Table 1 summarizes the main experimental results. FE-PPO achieves the highest scores in all three environments, doubling the performance of First-Explore in Dark Treasure Room and achieving the highest score in Ray Maze where RL^2 scores 0. In Bandits with One Fixed Arm, FE-PPO outperforms First-Explore using only (1/10) of the environment interactions reported by Norman & Clune (2024), and in Dark Treasure Room and Ray Maze it uses only (1/40). As a result of this improved sample-efficiency, our models trained in under 10 hours for Dark Treasure Room, and 6 hours for Ray Maze

Table 1: Mean cumulative reward of each method under *greedy action sampling* in hard-to-explore meta-RL domains. FE-PPO outperforms First-Explore on the first two domains while using significantly fewer samples, and has partial performance on the third domain. Across all domains, FE-PPO outperforms RL^2, though, under greedy action sampling, this difference is only significant on Dark Treasure Room and Ray Maze.

Method	Environment Samples	Bandit with One Fixed Arm	Dark Treasure Room	Ray Maze
FE-PPO	**0.5 B**	$\mathbf{131.43 \pm 4.53}$	$\mathbf{4.00 \pm 0.09}$	$\mathbf{0.517 \pm 0.01}$
FE	5 B / 20 B / 20 B†	127.70 ± 2.00	2.00 ± 0.30	0.47 ± 0.01
RL^2 (PPO)	**0.5 B**	129.19 ± 1.24	3.46 ± 0.10	0.000 ± 0.000

† FE uses ~5 billion environment steps for the bandit environment and ~20 billion for Dark Treasure Room and Ray Maze.

The performance of RL^2 depends strongly on greedy action sampling, whereas FE-PPO's remains comparable under stochastic evaluation, as indicated by the contrast between Table 1 and Figure 3. When sampling actions stochastically, we find that FE-PPO outperforms RL^2 in every environment according to a Mann-Whitney U-Test (p=0.0079 for all, full test results in Appendix E). We also compare results for each sampling strategy side by side in Table 2 in Appendix B. Some environments require stochastic policies for high performance (Jaakkola et al., 1994), so FE-PPO's robustness under stochastic evaluation suggests that its advantage over RL^2 in these hard-exploration domains may widen when deterministic action selection is suboptimal.

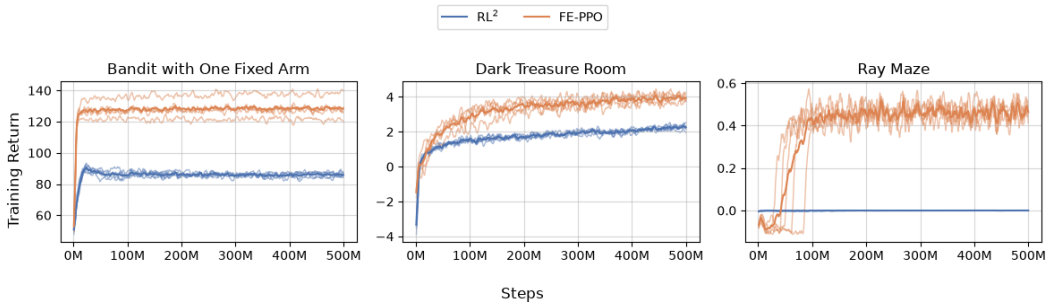

Figure 3: Cumulative trial-training return achieved by FE-PPO (orange) and RL^2 (blue) throughout training *under stochastic action sampling*. For each environment, we report the number of exploration episodes that gives the highest mean score: 9 for Bandit, 5 for Dark Treasure Room, and 1 for Ray Maze. Individual runs are shown in lighter colors. Unlike the results in Table 1, which use greedy action selection by always taking the highest-probability action, these training returns are computed by sampling actions directly from the learned policy distribution. Under this sampling, FE-PPO significantly outperforms RL^2.

5 Conclusion

Meta-reinforcement learning is a useful framework for learning exploration strategies, but it is expensive to study. These methods require many interactions across many task instances, a problem which is exacerbated by expensive transformer-based policies. Progress therefore depends not only on better objectives, but also on implementations that make large-scale meta-RL experiments fast enough to support scientific iteration.

This paper introduced First-Explore PPO, a PPO-based implementation of the First-Explore objective. FE-PPO preserves the central idea of First-Explore: exploration actions are optimized according to their effect on later exploitation. By expressing this objective as an actor-critic problem, FE-PPO uses value estimates to propagate exploitation returns back into exploration episodes, substantially improving sample efficiency while retaining strong performance on deceptive meta-RL benchmarks.

More broadly, FE-PPO turns First-Explore from a specialized, sample-intensive procedure into a practical and familiar PPO-style baseline. This opens up exciting opportunities to integrate First-Explore with the extensive research building on PPO, including intrinsic motivation methods and curriculum learning. Together with our optimized JAX and CleanRL implementations, environments, and baselines, FE-PPO can enable faster and more reproducible evaluation of meta-exploration methods under stronger baselines and at larger scales.

References

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

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

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 2978–2988, 2019.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Mikael Henaff, Roberta Raileanu, Minqi Jiang, and Tim Rocktäschel. Exploration via elliptical episodic bonuses. *Advances in Neural Information Processing Systems*, 35:37631–37646, 2022.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL http://jmlr.org/papers/v23/21-1342.html.

Tommi Jaakkola, Satinder Singh, and Michael Jordan. Reinforcement learning algorithm for partially observable markov decision problems. *Advances in neural information processing systems*, 7, 1994.

Chris Lu, Jakub Kuba, Alistair Letcher, Luke Metz, Christian Schroeder de Witt, and Jakob Foerster. Discovered policy optimisation. *Advances in Neural Information Processing Systems*, 35:16455–16468, 2022.

Michael Matthews, Michael Beukman, Benjamin Ellis, Mikayel Samvelyan, Matthew Jackson, Samuel Coward, and Jakob Foerster. Craftax: A lightning-fast benchmark for open-ended reinforcement learning. *arXiv preprint arXiv:2402.16801*, 2024.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Ben Norman and Jeff Clune. First-explore, then exploit: Meta-learning to solve hard exploration-exploitation trade-offs. *Advances in Neural Information Processing Systems*, 37:27490–27528, 2024.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. *Advances in neural information processing systems*, 29, 2016.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

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

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

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

Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*, 2017.

Ryan Sullivan, Ryan Pégoud, Ameen Ur Rehman, Xinchen Yang, Junyun Huang, Aayush Verma, Nistha Mitra, and John P Dickerson. Syllabus: Portable curricula for reinforcement learning agents. *arXiv preprint arXiv:2411.11318*, 2024.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, 1998.

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.

Adaptive Agent Team, Jakob Bauer, Kate Baumli, Satinder Baveja, Feryal Behbahani, Avishkar Bhoopchand, Nathalie Bradley-Schmieg, Michael Chang, Natalie Clay, Adrian Collister, et al. Human-timescale adaptation in an open-ended task space. *arXiv preprint arXiv:2301.07608*, 2023.

Supplementary Materials

The following content was not necessarily subject to peer review.

A Experimental Details

For each environment, rather than sampling from the full distribution of environments for each episode, we sample 10,000 unique environment instances and use these throughout training. This is a practical necessity because certain environment resets can be expensive, namely in the Ray Maze environment. Additionally, JAX code executes all possibilities in parallel rather than conditionally branching, so these resets must be executed at every step.

Each method is trained on a single NVIDIA L40s GPU and 8 CPUs. The main experiments required roughly 10 hours per run for FE-PPO and 8 hours per run for RL^2, for a total of 270 GPU hours. We used roughly double this, or an additional 300 GPU hours, for hyperparameter sweeps and testing while developing our method.

B Greedy vs. Stochastic Action Sampling

As noted in section 4 we notice a significant difference in performance when sampling greedily vs. stochastically from an RL^2 policy. To make this distinction more clear, we've plotted the evaluation scores using each sampling method in Table 2. We find that FE-PPO performs largely the same with each sampling strategy while the performance of RL^2 depends heavily on greedy action selection.

Table 2: Mean cumulative reward $\pm$ standard deviation under greedy and stochastic action sampling.

Algorithm	Sampling	Bandits with One Fixed Arm	Dark Treasure Rooms	Ray Maze
FE-PPO	Greedy	131.65 ± 4.54	3.92 ± 0.07	0.517 ± 0.01
	Stochastic	131.29 ± 4.47	3.92 ± 0.08	0.48 ± 0.01
RL^2	Greedy	129.19 ± 1.24	3.46 ± 0.1	0 ± 0
	Stochastic	86.69 ± 1.62	2.24 ± 0.08	0 ± 0

C Sweeping over the Number of Explore Episodes

After training FE-PPO, we sweep over the number of explore episodes to find the optimal tradeoff between exploration and exploitation. As we see in Figure 4, additional exploration episodes improve performance up to a point. However, each episode spent searching for information decreases the amount of time the agent can spend exploiting previously discovered rewards. This tradeoff is most visually apparent in the Bandit environments, where each extra exploration episode results in exactly 1 less pull of the best-known arm. This leads to a nearly linear decrease in cumulative reward as the number of exploration episodes increases past the optimal number.

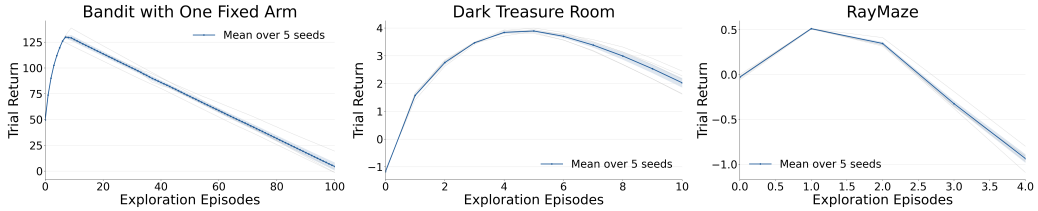

Figure 4: Cumulative return achieved by the agent across all episodes in a trial for different numbers of exploration episodes. The mean and standard deviation over 5 models is shown in blue and the returns for individual models are show in gray. This shows the tradeoff between collecting more information and exploiting currently known sources of rewards.

D Hyperparameters

Hyperparameters for RL^2 and FE-PPO were found through a randomized hyperparameter sweep of 50 trials, each run on 1 seed for 100 million environment steps. The hyperparameter ranges are shown in Table 3 and the selected values for each method and environment are shown in Table 4. Note that while we perform the exact same sweep for RL^2 and FE-PPO, we find that the best hyperparameters for RL^2 are the same across all environments. This is largely because all hyperparameter choices achieve the same high score in the Bandit with One Fixed Arm environment and score 0.0 in Ray Maze, so the best hyperparameters for Dark Treasure Room were as good as any alternative in the remaining environments. As a result, we chose the hyperparameter set that performed best in Dark Treasure Room to give the algorithm the best possible chance of succeeding.

Table 3: Hyperparameter sweep ranges.

Hyperparameter	Sampling distribution	Range / Choices
PPO hyperparameters		
Learning rate	Log-uniform	$[5 \times 10^{-5}, 5 \times 10^{-4}]$
Entropy coefficient	Log-uniform	$[10^{-3}, 10^{-1}]$
Clipping ratio	Uniform	$[0.1, 0.3]$
Discount factor γ	Discrete	$\{0.99, 0.995, 0.999\}$
GAE λ	Uniform	$[0.9, 0.99]$
Transformer-XL architecture		
Embedding size	Discrete	$\{64, 128, 256\}$
Number of layers	Discrete	$\{1, 2, 3\}$
Number of attention heads	Discrete	$\{4, 8\}$
Window size	Discrete	$\{32, 64, 128\}$
Hidden size	Discrete	$\{128, 256\}$
Attention gating	Discrete	$\{$`False`, `True`$\}$

Table 4: Chosen RL^2 and FE-PPO hyperparameters by environment.

Hyperparameter Environment	RL^2 All	FE-PPO Bandit	FE-PPO Dark Treasure Room	FE-PPO Ray Maze
PPO hyperparameters				
Learning rate	2.11×10^{-4}	1.75×10^{-4}	2.68×10^{-4}	1.69×10^{-4}
Anneal learning rate	`True`	`True`	`True`	`True`
Entropy coefficient	4.93×10^{-2}	9.15×10^{-2}	2.85×10^{-3}	6.67×10^{-2}
Value-function coefficient	0.5	0.5	0.5	0.5
Clipping ratio	0.275	0.131	0.278	0.158
Discount factor γ	0.999	0.999	0.995	0.995
GAE λ	0.926	0.940	0.926	0.950
Maximum gradient norm	0.5	0.5	0.5	0.5
Update epochs	4	4	4	4
Number of minibatches	4	4	4	4
Minibatch size	25600	25600	25600	25600
Number of environments	512	512	512	512
Steps per rollout	200	200	200	200
Total timesteps	5×10^8	5×10^8	5×10^8	5×10^8
Transformer-XL architecture				
Activation function	ReLU	ReLU	ReLU	ReLU
Embedding size	256	256	256	128
Number of layers	3	1	2	2
Number of attention heads	8	8	8	8
Window size	32	32	64	64
Hidden size	256	128	128	128
QKV feature size	256	256	256	128
Attention gating	`False`	`True`	`False`	`False`
Gating bias	0.0	0.0	0.0	0.0
Meta-learning hyperparameters				
First-Explore horizon K	–	100	10	4

E Statistical Tests

We conduct separate Mann–Whitney U tests comparing FE-PPO and RL^2 in each environment under both greedy and stochastic action selection. The results are shown in Table 5.

Table 5: Mann–Whitney U test results comparing FE-PPO and RL^2. Bolded p-values are statistically significant for $\alpha = 0.05$

Sampling	Environment	U_{FE}	p-value
Greedy	Bandit	20.0	0.1508
Greedy	Dark Treasure Room	25.0	**0.0079**
Greedy	Ray Maze	25.0	**0.0079**
Stochastic	Bandit	25.0	**0.0079**
Stochastic	Dark Treasure Room	25.0	**0.0079**
Stochastic	Ray Maze	25.0	**0.0079**