Does Novelty-Based Exploration Maximize Novelty?

Srinath Mahankali, Zhang-Wei Hong, Pulkit Agrawal
Improbable AI Lab, Massachusetts Institute of Technology

Abstract

Exploration is a central challenge in reinforcement learning. While novelty-based intrinsic rewards outperform random exploration in hard-exploration tasks, the mechanism by which they improve exploration is unclear. The dominant intuition is that such methods measure and incentivize novelty, resulting in superior exploration. Counterintuitively, we find that state-of-the-art exploration methods such as *Random Network Distillation* (RND) neither measure novelty accurately nor do they optimize novelty well. Based on the results, we hypothesize that RND's performance gain can be explained by an alternative perspective to novelty: state-dependent perturbation of the reward function. To test this insight, we develop a simple scheme for randomly generating intrinsic rewards that do not measure novelty. An experimental study on ATARI games, the standard benchmark for exploration, reveals that randomly generated intrinsic rewards match the performance of RND on most games. Our findings question the underlying assumption of intrinsic reward functions needing to represent novelty to help exploration, and provides motivation for further investigation into alternative forms of intrinsic rewards that might be more suited to high-dimensional observation spaces.

1 Introduction

Exploration is a central problem in reinforcement learning (RL) [31], which is even more challenging in sparse-reward scenarios where the learning signal is rarely available, impeding policy improvement. Inspired by the principle of *optimism in the face of uncertainty*, various exploration methods have been developed to incentivize the agent to visit novel states (i.e., uncertain states) [25, 21, 12, 22, 34, 2, 3, 10, 4] via exploration bonuses. We will refer to these exploration methods as *novelty-based exploration*. The standard paradigm is to augment the task reward (r_t^E; also called *extrinsic reward*) with an exploration bonus (r_t^I; also called *intrinsic reward*) and optimize the policy to maximize for the sum of rewards, $r_t^E + r_t^I$. Such methods have demonstrated impressive performance gains over naive exploration strategies such as ϵ-greedy on well-known hard exploration tasks (e.g., ATARI games like *Venture*, *Montezuma's Revenge*, and *Gravitar* [32, 4]). However, we find a gap between the theory and practice of why exploration bonuses improve performance, a topic we investigate.

A critical aspect of novelty-based exploration is the choice of the intrinsic reward. Suppose one is optimizing for worst-case regret in a tabular Markov Decision Process (MDP). In that case, the theoretically optimal choice for intrinsic reward is state novelty computed as the inverse state-action visitation counts $1/\sqrt{N(s,a)}$ [14], where $N(s,a)$ denotes the number of times the state-action pair, (s,a), occured in agent's experience. However, when the state space is continuous and high-dimensional, state-action visitation counts are ill-defined [2]. In such a scenario, one could consider measuring the state-action density, but it is challenging to estimate [18]. Consequently, many prior works investigated tractable alternatives or approximations to density estimation [2, 20, 22, 3, 23].

A state-of-the-art method for measuring novelty and, thereby encouraging exploration is *Random Network Distillation* (RND) [3]. RND computes novelty as the error in predicting a randomly generated low-dimensional embedding of a state. After visiting the state s, the agent also updates its prediction. Intuitively, the more times the agent visits a particular state, the more accurate its

Preprint. Under review.

prediction is going to be, and thus prediction error should be inversely proportional to how novel the state is. However, we found that this intuition does not always hold true. Even in simple grid-world environments, we found that RND rewards do not decrease monotonically at visited states during training (Section 3.1), which implies that RND may not measure novelty accurately. Despite the inaccuracy in measuring state novelty, RND has demonstrated significant performance improvement on challenging exploration tasks. This leads to a critical question: *"Is maximizing novelty the primary reason for the efficacy of novelty-based exploration?"*

One possibility is that despite the novelty measure being inaccurate, it might still provide a reasonably accurate gradient for policy optimization encouraging the agent to explore. To investigate if this the case, we studied the question: *"Can current policy optimization methods learn a policy that maximizes the novelty measure?"* It is standard practice within novelty-based exploration to simultaneously train the policy and the novelty estimator. As the novelty estimator changes every episode, the intrinsic reward function also changes over time. Because the policy is usually represented as a deep neural network, it cannot be perfectly optimized with respect to the updated reward function in one or a few gradient steps. Therefore, at any given time, the current policy is likely an imperfect optimizer of the intrinsic rewards [35]. To test how sub-optimal this optimization might be, we trained a policy using the state-of-the-art policy gradient method, PPO [26], to optimize only the RND reward. Quite surprisingly, we found that even in a simple grid-world environment, at different stages of training, a random policy achieved higher RND rewards than the trained PPO policy (see Section 3).

The above observations suggest that the intuition behind methods like RND improving exploration being that they measure and optimize novelty is incomplete. Neither does RND accurately measure novelty, nor can current methods optimize novelty. An alternate hypothesis is that intrinsic rewards like novelty measures perturb the policy optimization objective function, which encourages the agent to explore. To test this hypothesis, we constructed an alternative intrinsic reward, *random reward functions* (shorthand RPF), that perturb the policy optimization objective but do not measure novelty. Comparison against RND on ATARI games where RND significantly outperforms PPO trained only with game rewards reveals that most improvement obtained by RND over PPO can also be obtained by random reward functions (i.e., RPF; see Figure 1). On games where RND significantly outperforms PPO, RND outperforms RPF on some (e.g., *Montezuma's Revenge*), whereas RPF outperforms RND on others (e.g., *Venture*). Overall, RPF outperforms or matches RND in the majority of games. Note that the goal of this paper is not to develop a state-of-the-art exploration method, but to dissect the mechanism by which current intrinsic reward methods improve exploration.

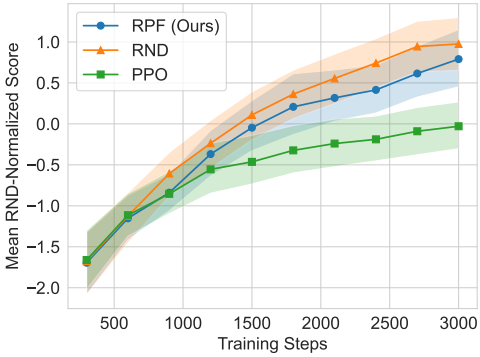

Figure 1: Random reward functions (denoted as RPF) that use random noise as intrinsic rewards are competitive to intrinsic reward methods measuring novelty (denoted as RND) and substantially outperform the policy trained with only task rewards (denoted as PPO). Performance is measured as the interquartile mean (IQM) [1] of normalized return.

In light of our findings, one way forward to improve exploration is to develop better measures of novelty and superior policy optimization strategies for maximizing exploration bonuses. A second view is that given the lack of theoretical justification for using novelty in continuous state spaces and the known difficulty of measuring novelty in high-dimensional spaces, perhaps an alternative worth considering is intrinsic rewards that are not based on novelty. Such intrinsic rewards *might* be more suitable for high-dimensional and continuous state-action spaces. However, this is not a well-studied topic in the literature. The experimental results from using random reward functions as intrinsic rewards provide hope that such a construction might be possible.

2 Preliminaries

Reinforcement learning (RL) optimizes an agent's policy in sequential decision-making problems [31]. The agent starts from an initial state s_0. At each timestep t, the agent perceives the state s_t, takes

an action $a_t \sim \pi(.|s_t)$ with its policy π, receives a reward $r_t = r(s_t, a_t, s_{t+1})$, and transitions to a next state s_{t+1} until reaching terminal states. The goal of RL is to learn a policy π to maximize the expected return yielded by the policy π:

$$\max_{\pi} J(\pi) = \mathbb{E}_{\pi}\Big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1})\Big]. \tag{1}$$

As the expected return accounts for long-term accumulated rewards, taking greedy action a that maximizes immediate reward $r(s, a)$ does not necessarily lead to high return. Thus, RL algorithms require "exploration" actions that lead to low immediate rewards but potentially high return in long run. We refer to this process as *exploration* throughout this paper. Many existing works have shown that exploration bonuses [3, 32, 22, 2] that encourage the policy to visit novel states are crucial for learning a good policy in hard-exploration tasks [2]. Exploration bonuses (also known as intrinsic rewards) are introduced into the reward function as follows: $r(s_t, a_t, s_{t+1}) = r^E(s_t, a_t, s_{t+1}) + r^I(s_t, a_t, s_{t+1})$, where r^E and r^I denote reward from the environment (i.e., task rewards) and exploration bonuses (i.e., intrinsic rewards), respectively. Our analysis in this paper is established on the intrinsic reward function given by Random Network Distillation (RND) [3]. RND is among the state-of-the-art exploration methods that estimates novelty in the following way: A neural network Φ is randomly initialized, while a predictor neural network $\widehat{\Phi}$ parameterized by θ is trained to minimize the loss via sampling data from the experiences (s, a, s') collected by the agent:

$$\min_{\theta} L(\theta) = \mathbb{E}_{(s,a,s')\sim\pi}\left[\left\|\widehat{\Phi}(s';\theta) - \Phi(s')\right\|_2^2\right], \tag{2}$$

where s' denotes next state after taking action a at state s. The intrinsic reward r^I is set to be:

$$r^I(s_t, a_t, s_{t+1}) = \underbrace{\|\widehat{\Phi}(s_{t+1};\theta) - \Phi(s_{t+1})\|_2^2}_{\text{Prediction error}} \tag{3}$$

This choice of r^I can be viewed as state novelty since errors of the predictor network are expected to be larger for states s_{t+1} which the agent has not frequently seen. In our experiments, we consider learning a policy using Proximal Policy Optimization (PPO) [26] with intrinsic rewards from RND.

3 Analyzing Why Novelty-Based Intrinsic Motivation Helps Exploration

The common intuition for why RND helps exploration is that (i) the RND reward function measures state novelty and (ii) a policy trained to optimize RND rewards successfully optimizes the RND reward function. We observe that these intuitions may not accurately align with the underlying mechanism of how intrinsic rewards facilitate exploration.

Experimental Setup To demonstrate this, we conduct experiments in a grid world environment with implementations based on the `cleanrl` codebase [13]. We train an agent in the grid world using PPO with *only* RND rewards (Equation 3) for one million timesteps and monitor how it explores during training. The agent receives no extrinsic rewards to allow us to test how well an agent that is trained using RND optimizes purely the RND reward function. The grid world, shown in Figure 2, has bounds $[-5, 5]$ in both the x and y-axes, the agent spawns at the top right with coordinates $(5, 5)$, and the agent's observation is its (x, y) coordinates. The agent can move up, down, left, or right by $\frac{1}{5}$ units per timestep and can move for 1000 timesteps in total per episode. The details of the experiments, including network architectures and hyperparameters, can be found in Appendix A.

We also analyze the behavior of an RND agent in more complex ATARI games, which are commonly used to benchmark exploration methods [33]. The implementation of RND we use is based on the `rlpyt` codebase [28], and uses Proximal Policy Optimization (PPO) [27] as the base algorithm for all experiments. We use the same PPO hyperparameters as in [4], which are used across all games, and use their hyperparameters for RND. We also use the same network architectures for the policy and value network as in [4]. We consider a set of thirteen ATARI games where RND significantly outperforms PPO, since it is likely that good performance in these games requires exploration. We obtained this set of games by considering the games where the mean performance of RND is significantly higher than PPO based on results in [4]. The list of thirteen games can be found in Appendix C.1. We train one RND agent on each of these games for 50 million frames, except for Montezuma's Revenge, where we train the RND agent for 100 million frames. The agent does not receive any extrinsic rewards in these experiments as well.

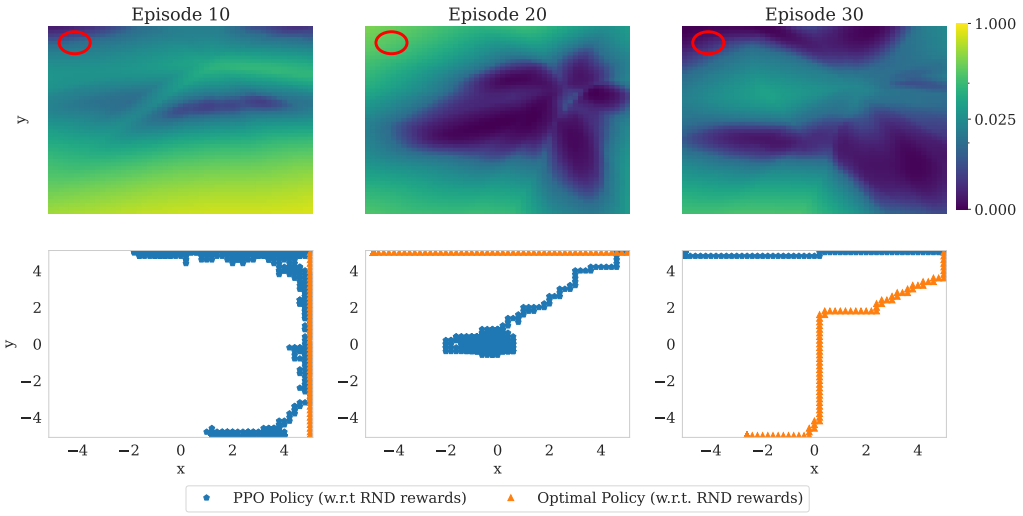

Figure 2: We train an agent in a grid world with no extrinsic rewards using RND. The agent observes the full state, which is its (x, y) coordinates and spawns at the top right. **Top row**: Heatmap visualization of the RND reward (in log scale) taken at different points throughout training. At many states, the RND reward function does not decrease monotonically during training, implying that RND is an imperfect representation of novelty. **Bottom row**: Comparison between the trajectory taken by the RND agent during training and the trajectory taken by the optimal agent with respect to the corresponding RND reward function in the top row. The trajectory taken by the optimal agent differs significantly from the RND agent's trajectory. Since the RND agent is rarely close to maximizing the RND reward function, it may not matter how accurate the novelty estimate is.

3.1 Does Novelty-Based Intrinsic Motivation Accurately Estimate Novelty?

Novelty-based exploration strategies such as RND are inspired by the principle of optimism in the face of uncertainty, which originally motivated exploration bonuses in tabular settings. In the tabular case, the optimal exploration bonus for a state is a monotonically decreasing function of the number of times the agent visits that state [29]. Thus, any accurate measure of novelty should also be monotonically decreasing. We find that the RND exploration bonus does not monotonically decrease, based on experiments in our simple grid world environment and challenging ATARI games.

Evaluating the Novelty Measure in a Grid World The top row of Figure 2 visualizes the evolution of the RND reward at every state in the grid world at episodes 10, 20, and 30 respectively with a symlog scale. Notice that in episode 10, the RND rewards are 0 at the top left corner (circled in red) of the grid, while in episode 20, the RND rewards have a magnitude close to 0.2 at the top left, where the reward is highest among all states. Similarly, in episode 20, the RND rewards are close to 0.05 near the bottom middle of the grid, but in episode 30, the RND rewards are maximized at the bottom middle with a value of approximately 0.1. One objection to this result could be that the capacity of the predictor network of RND may be too small, causing it to forget the correct predictions in states that have not been seen recently. To address this, we provide more experiments showing that this can happen even when the predictor network has increased capacity in Appendix B.

Evaluating the Novelty Measure in ATARI Games We train one agent using RND without extrinsic rewards for each ATARI game listed in Appendix C.1, and keeping track of the RND reward function during training. We again find that it is likely for the RND reward function to not monotonically decrease at all states. To see this, we consider the expected return of a uniform random policy with respect to the RND reward function at different points of training. Ideally, this expected return should decrease during training since the distribution of states visited by the random policy is constant. However, in the majority of games we considered, the expected return does not monotonically decrease. Plots of the expected return during training can be found in Appendix D.

3.2 Is Intrinsic Reward Actually Maximized?

Next, we investigate how well an agent trained to maximize RND performs at optimizing the RND reward. As our simple grid world environment is tabular, we can compute the optimal policy with respect to the RND reward function using value iteration.

We compare the RND policy trained using PPO to the optimal policy. We find that the two policies are rarely similar, showing that the agent does not optimize the RND rewards well. In the bottom row of Figure 2, the agent's trajectory (blue) is displayed alongside the optimal agent's trajectory (orange). The two trajectories rarely align, despite both agents starting at the same state.

Table 1: A random policy can achieve higher returns than the RND agent in terms of the RND reward function in our grid world environment.

Episode	Random Policy	RND Agent
10	41.01	**218.77**
15	**27.34**	14.71
20	**21.50**	0.62
25	**12.03**	1.36
30	**10.64**	0.19
35	**5.95**	0.25

We can confirm that the RND rewards are not optimized well by comparing the expected return of the agent to the expected return of a uniform random policy with respect to the RND reward function throughout training. We take the expectation over 100 evaluation episodes, and we find that a random policy can achieve higher return than the agent which is trained for the RND rewards, as shown in Table 1. We present further experiments across a wider range of hyperparameters in Appendix B.

4 Can Random Reward Functions Explore Similarly to RND?

Since the RND reward function might not accurately represent novelty, and since the RND agent rarely obtains an effective policy with respect to the RND reward function, we hypothesize that it may not be important for intrinsic rewards to accurately represent state-novelty to improve exploration. To test this, we consider an alternate reward function that does not measure novelty, but instead injects state-dependent noise in the reward function.

4.1 Generating Intrinsic Rewards with Random Potential Functions (RPF)

Intrinsic motivation methods for exploration add bias to the objective of maximizing the extrinsic reward [35, 4]. Novelty-based rewards tends to decay over training, mitigating this bias. On the other hand, the process by which the novelty-based reward decays is complex and thus difficult to emulate using random reward functions. To reduce the bias from random reward functions, we refer to [19], where it is shown that using a reward shaping term of the form $\gamma\phi(s') - \phi(s)$ for a transition (s, a, s'), does not change the optimality of a policy in the case of an infinite horizon MDP.

Random Potential Functions (RPF) Our algorithm, with pseudocode shown in Appendix E, periodically randomly initializes an intrinsic reward function. First, we initialize a neural network Φ which takes in an observation s as input. In our implementation, Φ has the same architecture as the target network in RND from [4]. For a transition of the form (s, a, s') where s is the current observation, a is the action, and s' is the next observation, RPF defines the intrinsic reward as $r_i(s, a, s') = \gamma\Phi(s') - \Phi(s)$, which is motivated by the potential-based reward shaping in [19]. Every n iterations, the last layer of the neural network Φ is re-initialized, where n is a hyperparameter.

Normalization Scheme Because the neural network Φ is untrained, we normalize the observations s and s' using a running mean and standard deviation as done by [3] before feeding them into Φ. We also normalize the rewards by dividing by a running estimate of their standard deviation. Finally, we normalize the extrinsic rewards using a running mean and standard deviation, which was previously found to improve the performance of RND in [4].

Visualizing the RPF and RND Agents in the Grid World We visualize the distributions of states visited by the RPF and RND agents in Figure 3 as a heatmap, and find that the distributions of visited states appear very similar in terms of their support. Specifically, if the agent visited state s a total of n times during training, we plot $\log(n + 1)$ for that state. This motivates the question: can we

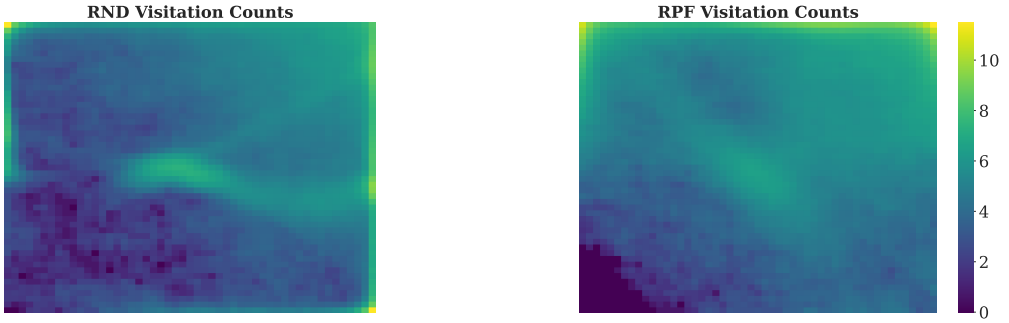

Figure 3: We visualize the state visitation counts of our RND agent in the grid world on the left. We also train an agent using RPF, which randomly samples potential-shaped reward functions [19] to generate intrinsic rewards, and visualize its state visitation counts in the heatmap on the right. Even though the intrinsic rewards are from randomly sampled reward functions and do not represent novelty, the distribution of visited states is similar between both methods in terms of their support.

achieve similar improvements in exploration, previously attributed to better novelty measurements, from choosing the intrinsic reward function randomly?

4.2 Evaluating the Exploration Performance of RPF

We test whether RPF, along with other approaches for randomly generating rewards can improve performance on ATARI games where exploration is necessary for good performance. Each intrinsic reward method has its own hyperparameters, shown in Appendix F, which are kept the same across different games. We train each algorithm for five seeds on each game for 50 million frames, except for Montezuma's Revenge, where we train each algorithm for 100 million frames. Different from Section 3, the agent is now trained to maximize the sum of extrinsic and intrinsic rewards.

To test whether random intrinsic reward functions can improve exploration, we test their performance on the thirteen games listed in Appendix C.1, where RND significantly outperforms PPO. To see if our method behaves in a similar way to RND in general, we also test whether our method hurts performance more than RND does on a subset of eight ATARI games where PPO significantly outperforms RND. This list of games can be found in Appendix C.2. We also obtain this list based on the results in [4] by choosing games where the mean score of PPO is significantly higher than that of RND. While we obtain these two sets of games using mean scores, we report aggregate performances using the interquartile mean (IQM) as it is robust to outliers while still being sensitive to the magnitude of many datapoints [1].

Measuring Exploration Performance with RND-Normalized Score To estimate how much of the improvement from PPO to RND can be captured by RPF, we need a metric which can be aggregated across different games. Motivated by past works which use a human-normalized score to do this [1, 17], we also consider a normalized score. However, for our work, we need to estimate the improvement compared to PPO and RND, so it is helpful to normalize the score of PPO to be 0 and the score of RND to be 1. Because of the set of games we choose to evaluate the improvement in exploration of RPF (Appendix C.1), it is possible to use such a metric.

For a given score x, and a game g, the RND-normalized score is defined as:

$$x_{RND,g} = \frac{x - \text{Score}_{PPO,g}}{\text{Score}_{RND,g} - \text{Score}_{PPO,g}}$$

where $\text{Score}_{PPO,g}$ and $\text{Score}_{RND,g}$ are the final performances of PPO and RND, respectively, on game g. We estimate $\text{Score}_{PPO,g}$ and $\text{Score}_{RND,g}$ by computing the median game score over the last 10 optimization epochs for each of the 5 runs and taking their mean. To aggregate training runs over all thirteen games with 5 runs for each game, we first normalize each training run using the RND-normalized score and compute the IQM over all training runs, as done in Figure 1.

4.3 Can Random Rewards Account for Performance Gains From RND?

We now test the hypothesis that most of the improvement in exploration from novelty-based intrinsic motivation can be explained by the simple strategy of periodically randomly sampling intrinsic reward functions. To test this, we consider the thirteen ATARI games listed in Appendix C.1 where RND significantly outperforms PPO.

Table 2: RND-normalized scores of various methods on ATARI games where RND significantly outperforms PPO (higher is better) in terms of RND-normalized score, where the score of RND is normalized to 1 and the score of PPO is normalized to 0. RPF matches or exceeds RND in performance in a majority of games, which occurs when its RND-normalized score on a game is greater than or equal to 1. Periodically reinitializing the intrinsic reward function is essential to the performance of RPF. Potential-based shaping is necessary for random reward functions to capture the majority of the improvement in performance of RND over PPO: making the reward function only a function of the next state (RSF) does not perform as well. However, applying potential-based shaping to the RND bonus actually *decreases* performance on these games compared to RND, suggesting that the reason for improved exploration with RPF is not solely due to potential-based shaping.

GAME	RPF (OURS)	RPF–RESETS	PRND	RSF
ASSAULT	**1.29**	0.57	0.68	0.77
BERZERK	**1.18**	0.81	0.94	-0.05
BOWLING	**2.29**	0.61	0.57	0.59
DEMONATTACK	0.30	0.44	0.06	**0.48**
FROSTBITE	1.91	1.80	**4.27**	2.41
ICEHOCKEY	**1.56**	1.44	1.31	1.32
KANGAROO	1.09	0.93	0.92	**1.24**
MONTEZUMAREVENGE	0.22	0.0	**0.56**	0.0
PHOENIX	-0.03	-0.03	-0.42	**0.84**
ROAD RUNNER	-0.80	-0.27	**0.50**	-0.10
SEAQUEST	0.55	0.19	**0.70**	0.40
TENNIS	-0.10	**0.94**	-0.02	-0.02
VENTURE	**1.02**	0.64	0.98	0.61
OVERALL SCORE	**0.73**	0.37	0.65	0.40

RPF Achieves Most of the Performance Gains From RND We plot the RND-normalized score, aggregated across all thirteen games, in Figure 1. From Figure 1, it is clear that RPF significantly improves over PPO in games where RND significantly improves over PPO. Figure 1 also shows that RPF captures most of the improvement which RND has over PPO, which suggests that RPF is competitive in exploration compared to RND.

Shown in Table 2 is the mean of the RND-normalized score of RPF for each game, along with an estimate of the IQM over all 65 runs (5 runs per game, 13 games). We show the RND-normalized score so that performance can be compared across different games. The raw scores can be found in Appendix G. Based on the estimated IQM over all 65 runs in Table 2, we see that over 70% of the improvement from RND is captured by RPF, with RPF exceeding RND in performance on a majority of these games. In particular, RPF exceeds RND on seven of the games: `assault`, `berzerk`, `bowling`, `frostbite`, `ice_hockey`, `kangaroo`, `venture`. RPF also improves over PPO on ten out of thirteen games, which shows how RPF is capturing a majority of the improvement from RND. We also find that RND does not have a significant chance of improving over RPF (see Appendix G).

Performance of RPF on Other Games We also compare RPF to PPO and RND in games where PPO significantly improves on RND. This list of games is shown in Appendix C.2. We find that RPF performs slightly better than RND on these games, with detailed results described in Appendix I.

4.4 Which Parts of RPF Matter?

Network Reinitialization is Necessary for RPF We find that network reinitialization is an essential component of RPF, without which performance drops significantly. We consider "RPF–Resets," a variant of RPF which does not reinitialize the potential function periodically, on the same 13 games where RND significantly outperforms PPO. As shown in Table 2, RPF performs significantly better than RPF–Resets. Using network reinitialization improves performance on 10 out of 13 games. Network reinitialization leads to an aggregate improvement of 0.36 in RND-normalized score, clearly indicating that it is a necessary component of RPF.

Can Other Random Rewards Improve Exploration? We also consider Random State Functions (RSF), which periodically randomly initializes intrinsic reward functions of just the next state, and does not use potential-based reward shaping. We find that while it greatly improves over PPO as shown in Table 2, it is still very far behind RPF. Similar to RPF, RSF initializes a neural network Φ which takes in an observation s as input. For a transition of the form (s, a, s'), the intrinsic reward is $r_i(s, a, s') = \Phi(s')$. The neural network Φ is re-initialized every n iterations, where n is a hyperparameter. Thus, potential-based shaping might be necessary for random reward functions. One possible explanation could be that mitigating the intrinsic reward bias is necessary as the random reward functions do not decay throughout training, causing the intrinsic reward bias to harm performance. RPF mitigates the bias from intrinsic rewards, and we discuss this issue in depth in Appendix M. In addition, we test "RSF–Resets", a variant of RSF where the intrinsic reward function is not reinitialized periodically and is held fixed throughout training. We also test if state-dependent reward functions are necessary by testing "Normal Noise," where we simply add normal noise to the extrinsic rewards. As we show in Appendix H, both these methods significantly underperform RPF.

4.5 Is the Improvement in Exploration Because of Potential Shaping?

In addition to testing the performance of RPF compared to RND on the games where RND shows the highest improvement over PPO, we also aim to understand why RPF is able to improve exploration despite not measuring novelty. In this section, we test whether potential-based shaping is the cause of the improvement, and find that combining it with novelty-based intrinsic motivation hurts exploration.

Potential-based RND (PRND) We test the performance of potential-based shaping applied to RND on the same set of thirteen ATARI games and find that it hurts performance. Similar to RND, in PRND, a neural network Φ is randomly initialized, and a predictor neural network $\widehat{\Phi}$ parameterized by θ is trained to minimize the loss $L(\theta) = \left\|\widehat{\Phi}(s';\theta) - \Phi(s')\right\|_2^2$, where s' is the next observation. However, in PRND, the intrinsic reward of a transition (s, a, s') is: $r_i(s, a, s') = \gamma \left\|\widehat{\Phi}(s';\theta) - \Phi(s')\right\|_2^2 - \left\|\widehat{\Phi}(s;\theta) - \Phi(s)\right\|_2^2$, where s is the current observation, a is the action, s' is the next observation, and γ is the discount rate. Observation, intrinsic, and extrinsic reward normalization are done in the same way as for RND.

Performance of PRND Where RND is Helpful PRND performs worse than RND on these games, as shown in Table 2, where PRND has an overall RND-normalized score of 0.65. This suggests that RND performs better than PRND, while RPF and PRND are comparable in performance. This implies that combining potential-based shaping with RND can hurt exploration. RND also has a significant chance of improving over PRND, as we show in Appendix G.

Performance of PRND on Other Games We also test the performance of PRND on the games listed in Appendix C.2 where PPO significantly improves on RND. Interestingly, PRND performs better than PPO on these games. We refer the reader to Appendix I for more detailed results.

5 Related Work

Our findings reveals that randomness, if set properly, can show competitive performance with novelty-based exploration. Randomness plays a vital role in exploration in RL. An example of this is ϵ-greedy exploration [16]. Entropy regularization is another instance of this [36, 15], which encourages a policy to explore more by adding the entropy to the objective function. A related approach for encouraging more random actions by an agent is MaxEnt RL [11, 6, 9], which instead adds the entropy of the agent's policy to the reward function to encourage the agent to act as randomly as possible while still solving the task. Another alternative approach, currently restricted to off-policy methods, is to inject noise into the policy parameters, rather than the actions [8]. Our method, RPF, can be viewed through this lens of injecting noise into the agent by instead injecting a transition-dependent perturbation from a random function other than the policy into the reward function.

Randomness is also involved in novelty-based exploration but in a different form. A variety of different novelty-based exploration bonuses have been proposed which generate prediction errors as intrinsic rewards. While many prediction problems used to generate the intrinsic rewards are based

on important factors in the environment such as transition dynamics [22, 23, 24], some approaches randomly generate the prediction problem. A central example of this is RND [3] itself, which randomly initializes a fixed target network and the prediction problem is to predict the outputs of the target network. In contrast, RPF simply uses the random reward function for intrinsic motivation.

In addition to randomness, exploration in RL can also benefit from a constant shift in the reward function. Recent work by [30] suggests that shifting the reward function by a negative constant can lead to more exploration, while a positive constant leads to more exploitation. However, applying novelty-based intrinsic rewards and the random reward functions tested in Section 4 to RL involves shifting the reward function based on states, rather than a constant value or sign. Therefore, constant shifts alone may not fully explain the exploration gains achieved by these types of reward functions.

6 Discussion

Implications. Since RND only slightly improves over RPF, this suggests that either more accurate measures of novelty should be created, better learning algorithms which can benefit from better novelty estimates are required, or estimating novelty may not be an essential piece of exploration. We do not argue that random reward functions should be used over novelty-based intrinsic motivation. However, we think this work shows that current approaches to novelty-based intrinsic motivation may not be the way forward.

Limitations. We find that RPF performs worse than RND in Montezuma's Revenge, the most popular ATARI game for benchmarking exploration methods [3, 5], despite performing much better than PPO. We believe this is due to the non-stationarity of the intrinsic reward from RPF, which does not decay over time. This is unlike the RND reward function, which tends to decay over training. We discuss this in detail in Appendix M.

RND Does Not Work the Way You Think It Does. We analyzed the behavior of an agent trained using RND in Section 3. While it explores far better than PPO, the behavior of the RND reward function during training violates the intuition that it represents novelty accurately, and the agent's exploration behavior violates the intuition that it successfully optimizes the RND reward function. Some ways to make the behavior of RND match this intuition are to improve novelty estimation and design learning algorithms which can quickly optimize the RND reward function. However, we consider designing intrinsic rewards which do not measure novelty as an alternate direction.

Significantly Improving Exploration is Possible with Random Reward Functions.

We design and study RPF, an intrinsic motivation algorithm which generates intrinsic rewards by periodically randomly initializing an intrinsic reward function. Despite RPF not estimating novelty to compute intrinsic rewards, we show that RPF captures most of the improvement which RND [3] has over PPO [27]. We also show that RPF performs comparably to RND when PPO improves on RND.

Explanations for the Performance of RPF. One possible reason why RPF could be improving performance is that it provides an inductive bias which could either reward the agent for novel experiences or align with the extrinsic reward. However, this cannot be the case because of two reasons: First, RPF frequently re-initializes the reward functions, so any inductive bias present in one iteration of RPF will be reset. Second, the architecture of RPF has a linear layer as its output layer, which is symmetric about the origin. This means that the intrinsic reward given by RPF is approximately 0 on average, as we show in Appendix L. An alternative hypothesis for the improved performance of RPF might be that it improves on RND in games where the RND agent is unable to effectively optimize the RND reward function, even when compared to a random policy.

Necessary Components of RPF. We find that using a transition-dependent reward function, periodically reinitializing the reward function, and potential shaping are all crucial to the performance of RPF. While it is not known whether there exist other ways of randomly generating reward functions which perform as well or better than RPF in the context of exploration, other random reward functions have also been proposed such as Random Network Indicators [7]. Since RPF is only one way to randomly generate reward functions, it may be possible to further improve these methods and go beyond the performance of RND.

References

[1] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 34, 2021.

[2] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *NIPS*, 2016.

[3] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019.

[4] Eric Chen, Zhang-Wei Hong, Joni Pajarinen, and Pulkit Agrawal. Redeeming intrinsic rewards via constrained optimization. *arXiv preprint arXiv:2211.07627*, 2022.

[5] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

[6] Benjamin Eysenbach and Sergey Levine. Maximum entropy rl (provably) solves some robust rl problems. *arXiv preprint arXiv:2103.06257*, 2021.

[7] Jesse Farebrother, Joshua Greaves, Rishabh Agarwal, Charline Le Lan, Ross Goroshin, Pablo Samuel Castro, and Marc G Bellemare. Proto-value networks: Scaling representation learning with auxiliary tasks. *arXiv preprint arXiv:2304.12567*, 2023.

[8] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017.

[9] Divyansh Garg, Joey Hejna, Matthieu Geist, and Stefano Ermon. Extreme q-learning: Maxent rl without entropy. *arXiv preprint arXiv:2301.02328*, 2023.

[10] Zhaohan Daniel Guo, Shantanu Thakoor, Miruna Pîslar, Bernardo Avila Pires, Florent Altché, Corentin Tallec, Alaa Saade, Daniele Calandriello, Jean-Bastien Grill, Yunhao Tang, et al. Byol-explore: Exploration by bootstrapped prediction. *arXiv preprint arXiv:2206.08332*, 2022.

[11] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[12] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *NIPS*, 2016.

[13] 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.

[14] J Zico Kolter and Andrew Y Ng. Near-bayesian exploration in polynomial time. In *Proceedings of the 26th annual international conference on machine learning*, pages 513–520, 2009.

[15] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016.

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2015.

[18] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[19] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287, 1999.

[20] Georg Ostrovski, Marc G Bellemare, Aäron Oord, and Rémi Munos. Count-based exploration with neural density models. In *International conference on machine learning*, pages 2721–2730. PMLR, 2017.

[21] Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V Hafner. Intrinsic motivation systems for autonomous mental development. *Evolutionary Computation*, 2007.

[22] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2778–2787, 2017.

[23] Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In *International Conference on Machine Learning*, pages 5062–5071. PMLR, 2019.

[24] Aditya Ramesh, Louis Kirsch, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Exploring through random curiosity with general value functions. *arXiv preprint arXiv:2211.10282*, 2022.

[25] Jurgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *From animals to animats: Proceedings of the first international conference on simulation of adaptive behavior*, 1991.

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

[27] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[28] Adam Stooke and Pieter Abbeel. rlpyt: A research code base for deep reinforcement learning in pytorch. *arXiv preprint arXiv:1909.01500*, 2019.

[29] Alexander L Strehl and Michael L Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

[30] Hao Sun, Lei Han, Rui Yang, Xiaoteng Ma, Jian Guo, and Bolei Zhou. Optimistic curiosity exploration and conservative exploitation with linear reward shaping. In *Advances in Neural Information Processing Systems*, 2022.

[31] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. 2018.

[32] Adrien Ali Taïga, William Fedus, Marlos C Machado, Aaron Courville, and Marc G Bellemare. Benchmarking bonus-based exploration methods on the arcade learning environment. *arXiv preprint arXiv:1908.02388*, 2019.

[33] Adrien Ali Taiga, William Fedus, Marlos C. Machado, Aaron Courville, and Marc G. Bellemare. On bonus based exploration methods in the arcade learning environment. In *International Conference on Learning Representations*, 2020.

[34] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *31st Conference on Neural Information Processing Systems (NIPS)*, volume 30, pages 1–18, 2017.

[35] William F Whitney, Michael Bloesch, Jost Tobias Springenberg, Abbas Abdolmaleki, Kyunghyun Cho, and Martin Riedmiller. Decoupled exploration and exploitation policies for sample-efficient reinforcement learning. *arXiv preprint arXiv:2101.09458*, 2021.

[36] Ronald J Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

A Grid World Experiment Details

Hyperparameters We list the hyperparameters and configurations for the experiments described in Section 3 in Table 3. However, we present further experiments with a different set of architectures for the RND predictor and target networks in Appendix B. We also present further experiments with a wider range of entropy loss coefficients in Appendix B, to show that the RND return achieved by the policy is not because of a low entropy coefficient.

Compute Usage Each individual run takes approximately 5 minutes on a MacBook Pro. Across all experiments in our grid world environment, we tested approximately 30 runs in total.

Table 3: Hyperparameters and network architectures for grid world experiments.

Parameter	Value
PPO	
Total Timesteps	$1,000,000$
Optimizer	Adam
Learning Rate	0.001
Adam Epsilon	0.00001
Number of Parallel Environments	32
Discount Rate	0.99
Generalized Advantage Estimation λ	0.99
Minibatches per Epoch	4
Epochs per Training Step	4
Clipping Coefficient	0.2
Entropy Loss Weight	0.01
Discount Rate	0.99
Value Loss Weight	0.5
Gradient Norm Bound	0.5
Use Advantage Normalization	True
Use Clipped Value Loss	True
Policy Network Architecture	MLP (64,64,4)
Value Network Architectures	MLP (64,64,1)
RND	
Intrinsic Reward Coefficient	1.0
Drop Probability	0.25
Learning Rate	0.0001
Predictor Network Architecture	MLP (64,64,4,4,4)
Target Network Architecture	MLP (64,64,4)
RPF	
Intrinsic Reward Coefficient	1.0
Epochs Before Resetting	10
Potential Function Network Architecture	MLP (64,64,4)
Value Iteration	
Maximum Number of Iterations	$10,000$
Tolerance Before Stopping	0.0001

B Further Grid World Experiments

Increasing the RND Predictor Network Capacity In Section 3.1, we showed that it is possible for the RND reward function to not be monotonically decreasing at every state, indicating that it is an imperfect estimate of novelty. Here, we show that even when the capacity of the RND reward function is increased, it is still possible for the rewards to not be monotonically decreasing. Specifically, we change the architecture of the RND predictor network to be MLP (64,64,256,256,256,256,256), and the architecture of the RND target network to be MLP (64,64,256). Note that there are only

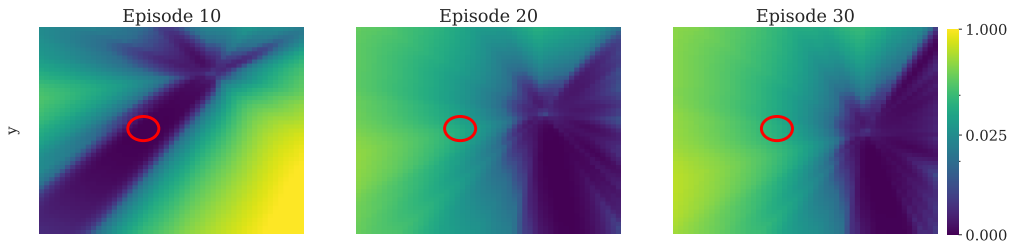

Figure 4: Heatmap visualization of the RND reward function (in log scale) taken at different points throughout training. Even when the predictor network has increased capacity, it is possible for the rewards to not be monotonically decreasing, as shown in the red circle.

Table 4: Returns of a random policy and an RND agent with respect to the RND reward function in our grid world environment, with different entropy coefficients.

Entropy Loss Coefficient	Episode	Random Policy	RND Agent
0.001	10	**112.33**	4.62
	15	17.97	**20.05**
	20	**36.44**	3.50
	25	**16.49**	1.77
	30	**13.98**	1.24
	35	**8.52**	1.89
0.01	10	41.01	**218.77**
	15	**27.34**	14.71
	20	**21.50**	0.62
	25	**12.03**	1.36
	30	**10.64**	0.19
	35	**5.95**	0.25
0.1	10	**68.73**	24.20
	15	**21.10**	2.65
	20	11.05	**18.56**
	25	**26.96**	0.97
	30	36.37	**53.22**
	35	**7.18**	0.32
1.0	10	**17.90**	4.12
	15	7.09	**8.56**
	20	**4.55**	0.85
	25	**4.17**	1.97
	30	**3.03**	2.32
	35	**3.75**	0.81

$50 \cdot 50 = 2500$ states, implying that in this case, the predictor network is overparameterized. We show the RND reward function at different points during training in Figure 4, and shown in the red circle is a region where the reward increases throughout training. Thus, since a good estimate of novelty should monotonically decreasing at all states, RND is an imperfect estimate of novelty even with increased network capacity.

Varying the Entropy Coefficient In Section 3.2, we showed that it is possible to achieve higher RND returns with a uniform random policy, compared to the RND agent which was actually trained with the RND rewards. One possible explanation for this is that the agent's policy is not random enough, and therefore the entropy loss coefficient used to train the RND agent is too low. We refute this possibility in Table 4, where we keep the original architecture of the RND predictor and target networks the same and vary the entropy loss coefficient. Over a variety of entropy loss coefficients, we find that the random policy usually achieves higher RND return than the RND agent itself.

C Lists of ATARI Games

C.1 ATARI Games Where RND Significantly Outperforms PPO

- assault
- berzerk
- bowling
- demon_attack
- frostbite
- ice_hockey
- kangaroo
- montezuma_revenge
- phoenix
- road_runner
- seaquest
- tennis
- venture

C.2 ATARI Games Where PPO Significantly Outperforms RND

- amidar
- carnival
- elevator_action
- gopher
- hero
- enduro
- star_gunner
- zaxxon

D Does RND Reward Monotonically Decrease in ATARI Games?

As we claimed in Section 3.1, the expected RND return achieved by a random policy does not monotonically decrease in the majority of games where RND outperforms PPO (Appendix C.1). We observed that at initialization, the agent's policy is very close to a uniform random policy, which means the RND predictor network is initially trained on a state distribution similar to that of a random policy. However, as we show in Figure 5, the RND reward function is rarely monotonically decreasing, implying that it is an imperfect measure of novelty.

E Pseudocode for RPF

The pseudocode for RPF is shown in Algorithm 1.

F Further Details of ATARI Experiments

Hyperparameters We display the hyperparameters for the different intrinsic rewards we considered in Table 5. Our base algorithm for all experiments is PPO [27], and we use the hyperparameters directly from [4]. We do not tune the hyperparameters of RND and obtain them directly from [4]. For the different strategies of generating random intrinsic rewards that we considered (RPF, RPF–Resets, RSF, RSF–Resets, Normal Noise), we first tuned the hyperparameters displayed in Table 5 on the hard-exploration ATARI game Venture. Specifically, we considered intrinsic reward coefficients in

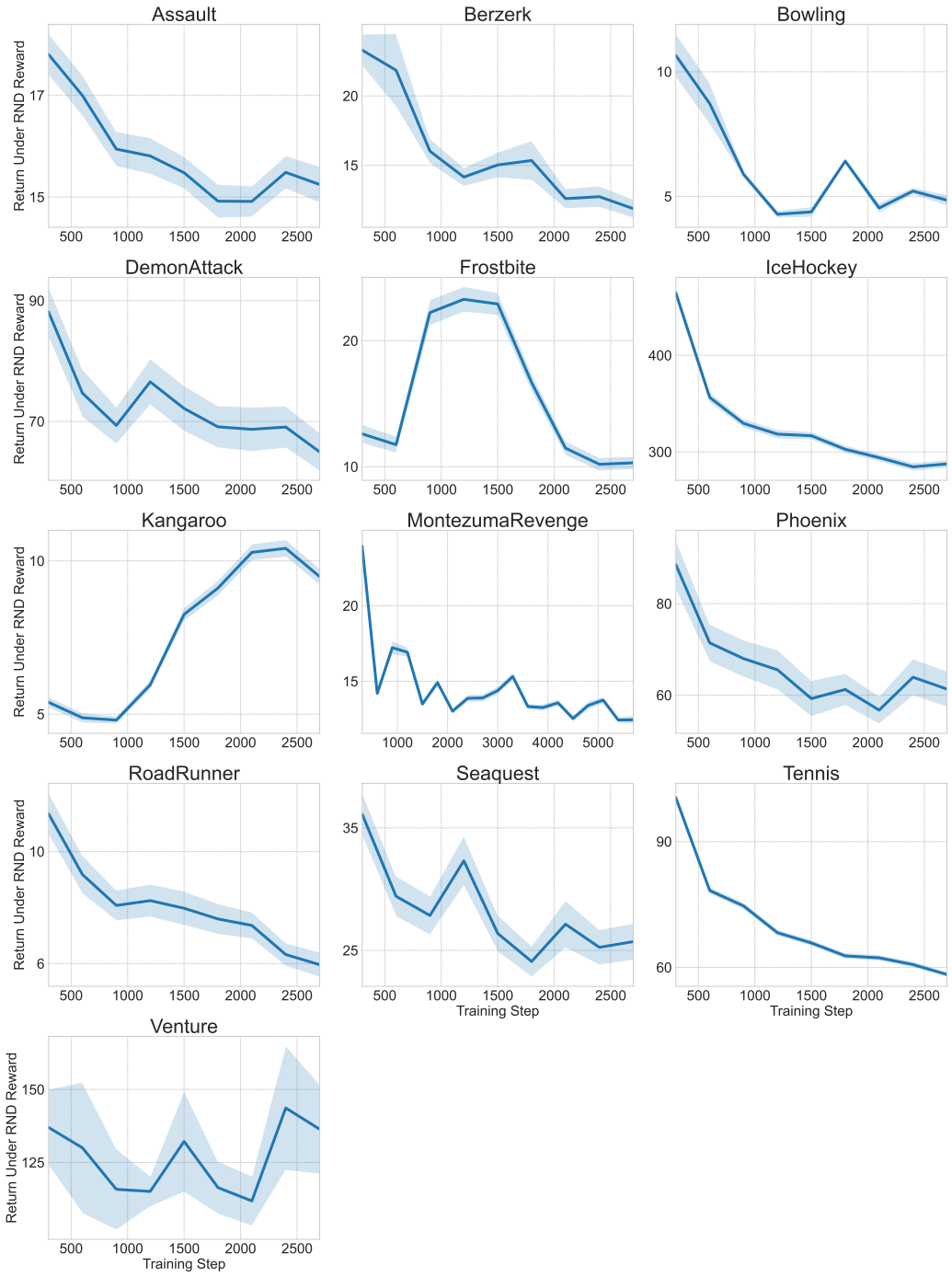

Figure 5: Expected return of a random policy under the RND reward function in 13 ATARI games where RND outperforms PPO. The expected return of the random policy is calculated over 100 episodes.

Algorithm 1 Intrinsic Rewards with RPF

$N_{iter} \leftarrow$ total number of iterations
$P \leftarrow$ number of parallel environments
$T \leftarrow$ length of rollout in each environment
$n \leftarrow$ max iterations per reward function
$\gamma \leftarrow$ discount rate
  Initialize neural network Φ
  Sample initial observation $s_0 \sim p(s_0)$
  for $i = 1$ **to** N_{iter} **do**
    for $j = 1$ **to** P **do**
      for $k = 1$ **to** T **do**
        sample action $a \sim \pi(a \mid s)$
        sample next observation $s' \sim \pi(s' \mid s, a)$
        Calculate intrinsic reward $\gamma\Phi(s') - \Phi(s)$
      end for
    end for
    if $i \equiv 0 \pmod{n}$ **then**
      Re-initialize Φ
    end if
  end for

the set $\{0.01, 0.1, 1.0\}$ and number of epochs before resetting (if applicable) in the set $\{1, 5, 10, 20\}$ and ran each combination on Venture for three seeds. We chose the combination which had highest performance on average. For PRND, we tuned the intrinsic reward coefficient according to the same procedure among the options of $\{0.01, 0.1, 1.0\}$.

Compute Usage Each individual seed of our strategies for randomly generating intrinsic rewards took approximately 3 hours on an NVIDIA RTX A6000 for 50 million frames. In Montezuma's Revenge, each agent is trained for 100 million frames, so each seed takes approximately 6 hours. Across all experiments in ATARI games, we tested approximately 900 runs in total.

G Detailed Results on Games Where RND Outperforms PPO

We list the mean scores of all the methods in games where RND outperforms PPO in Table 6. We also compute the probability that RND improves over PPO, RPF, and PRND in these games in Figure 6.

Probability of Improvement of RPF We find that RND does not have a significant chance of improving over RPF, suggesting that the benefit to exploration from the novelty estimation done by RND is small. In these ATARI games, RPF has a probability of 0.41 ± 0.05 of improving over RND as shown in Figure 6, where the 95% confidence interval is estimated using the bootstrap method.

Probability of Improvement of PRND PRND performs worse than RND on these games, as shown in Figure 6, where PRND has a probability of 0.38 ± 0.05 of improving over RND. This suggests that RND performs better than PRND, while RPF and PRND are comparable in performance.

H Results of RSF–Resets and Normal Noise

We also test two more strategies of randomly generating intrinsic rewards.

RSF–Resets In "RSF–Resets," the intrinsic reward is $r_i(s, a, s') = \Phi(s')$, but Φ is never re-initialized. To compensate for the consistent bias introduced by the random intrinsic reward function, we decrease the coefficient η of the intrinsic reward for this method.

Normal Noise We also test whether it is possible to improve over PPO in games where RND significantly outperforms PPO without using a state-dependent reward function. To test this, we sample intrinsic rewards from the normal distribution $\mathcal{N}(0, 1)$ and multiply by a scalar constant η.

Table 5: Hyperparameters for different ways of generating intrinsic rewards.

Parameter	Value
RND	
Intrinsic Reward Coefficient	1.0
Drop Probability	0.25
Learning Rate	0.0001
RPF	
Intrinsic Reward Coefficient	0.1
Epochs Before Resetting	5
RPF–Resets	
Intrinsic Reward Coefficient	0.1
RSF	
Intrinsic Reward Coefficient	0.1
Epochs Before Resetting	5
RSF–Resets	
Intrinsic Reward Coefficient	0.01
Normal Noise	
Intrinsic Reward Coefficient	0.1
PRND	
Intrinsic Reward Coefficient	1.0
Drop Probability	0.25
Learning Rate	0.0001

Table 6: Mean scores of various methods on the games where RND significantly outperforms PPO.

GAME	PPO	RND	RPF	PRND	RSF	RSF–RESETS	NOISE
ASSAULT	8426.3	11231.0	**12042.6**	10321.7	10576.9	8396.9	9011.2
BERZERK	1132.4	1448.7	**1504.8**	1429.7	1116.6	1329.1	1139.7
BOWLING	18.0	24.2	32.2	21.6	21.7	**35.1**	14.7
DEMONATTACK	5621.9	**9123.9**	6681.1	5842.4	7289.4	6332.3	5260.4
FROSTBITE	1080.0	1740.3	2340.9	3900.8	2669.4	**4454.6**	955.8
ICEHOCKEY	5.6	8.8	**10.6**	9.8	9.8	9.5	6.9
KANGAROO	6125.3	8272.9	8463.2	8108.1	8794.3	**10826.1**	9055.1
MONTEZUMAREVENGE	0.1	**2496.0**	546.4	1389.6	0.0	0.0	2.6
PHOENIX	8134.1	**10905.8**	8058.9	6963.7	10456.2	8315.8	4411.7
ROAD RUNNER	46687.2	**59545.0**	36377.3	53168.6	45358.2	58217.2	31462.7
SEAQUEST	1463.8	**1989.3**	1752.1	1831.9	1671.5	1877.0	1326.1
TENNIS	-0.1	**4.6**	-0.49	-0.16	-0.2	-0.3	9.0
VENTURE	104.5	1370.1	**1392.9**	1343.6	879.2	808.1	387.7

We present the RND-normalized score of the two methods in Table 7.

I Performance On Games where PPO Outperforms RND

As we showed in Section 4.3, it is possible to obtain a large majority of the gain in performance due to RND by using RPF, i.e., without measuring any form of novelty. However, one might wonder if this comes at a cost: in games where PPO significantly outperforms RND, does RPF get outperformed by RND? In the analysis conducted below, we find that RPF and RND perform similarly on such games, indicating that RPF and RND generally perform similarly.

We consider a set of eight ATARI games, listed in Appendix C.2, where PPO significantly outperforms RND. On these games, PPO has a probability of improvement of 0.65 ± 0.06 over RND, as shown

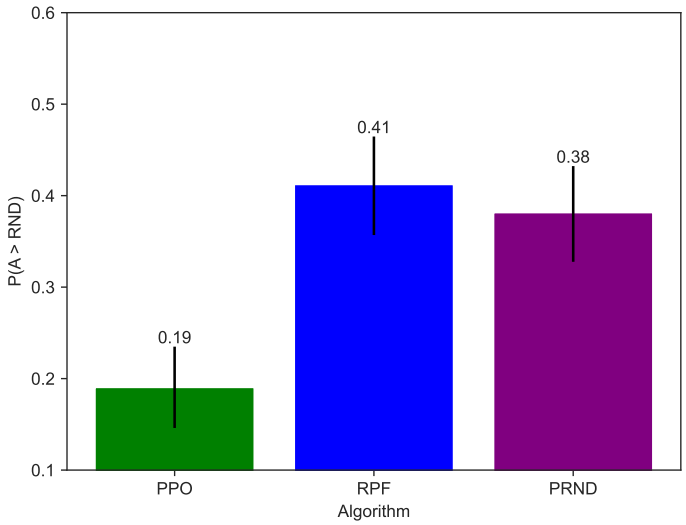

Figure 6: We compare RND to RPF on the set of thirteen ATARI games, listed in Appendix C.1, where RND significantly outperforms PPO. RPF, shown in blue, has a probability of improvement [1] of 0.41 ± 0.05 over RND on this set of games, indicating that it has performance close to that of RND. Meanwhile, PPO, shown in green, only has a probability of improvement of 0.19 ± 0.04 over RND on these games, showing that this is indeed a set of games where novelty-based intrinsic motivation leads to large improvements over pure extrinsic reward maximization. This improvement in performance over PPO is not orthogonal to RND and not solely due to potential-based reward shaping, since potential-based RND (PRND) has a probability of improvement over RND of 0.38 ± 0.05, which is slightly lower than that of RPF.

in Figure 7, showing that PPO indeed significantly outperforms RND on this set of games. We also find that RPF slightly outperforms RND with a probability of improvement of 0.53 ± 0.07. Thus, the probability of improvement of RPF over RND is less than that of PPO over RND, showing that RPF is also outperformed by PPO on these games. However, as the probability of improvement of RPF over RND is over 0.5, this means RPF and RND perform comparably. We also present the game breakdown of RPF in Table 8, where we see that RPF improves on RND in a majority of games in terms of the IQM of game score, further validating our claim that RPF performs comparably to RND on these games.

We also test the performance of potential shaping applied to RND on these games. Despite PRND being a novelty-based intrinsic motivation technique, it performs the best out of all methods, having the highest probability of improvement over RND of 0.72 ± 0.06 as shown by the purple bar in Figure 7. As shown in Table 8, PRND outperforms PPO in six out of eight games. This suggests that even when PPO outperforms RND, increased exploration can improve performance, provided the intrinsic reward bias is alleviated through methods such as potential shaping.

J Learning Curves on All ATARI Games

We plot the learning curves of PPO, RND, RPF, and PRND in 60 ATARI games in Figure 8. For PRND, we only plot learning curves on the 21 ATARI games listed in Appendix C as we only ran experiments on those games.

Table 7: RND-normalized scores of RSF–Resets and Normal Noise, two further methods of randomly generating rewards, on ATARI games where RND significantly outperforms PPO (higher is better). RSF–Resets improves significantly over PPO but does not reach the performance of RPF, indicating that the combination of potential shaping and network reinitialization is important for performance of our random rewards. Sampling the intrinsic rewards from a normal distribution that is independent of state does not improve performance at all, showing that it is necessary for intrinsic rewards to take in at least the next state as input. Each method was tested on each game for five random seeds.

Game	RSF–Resets	Normal Noise
Assault	-0.01	0.21
Berzerk	0.62	0.02
Bowling	2.77	-0.53
DemonAttack	0.20	-0.10
Frostbite	5.11	-0.18
IceHockey	1.21	0.41
Kangaroo	2.19	1.36
MontezumaRevenge	0.0	0.0
Phoenix	0.06	-1.34
Road Runner	0.90	-1.18
Seaquest	0.79	-0.26
Tennis	-0.06	1.95
Venture	0.56	0.22
Overall Score	0.45	-0.15

Table 8: Mean game score over five random seeds for various methods on ATARI games where PPO outperforms RND (higher is better). Our method, RPF, performs similarly to RND, showing that using random reward functions may not hurt performance any more than novelty-based exploration hurts performance. Interestingly, PRND achieves the best performance on six out of these eight games, even though this set of games is chosen so that PPO outperforms RND.

Game	PPO	RND	RPF (Ours)	PRND
Amidar	1075.3	774.7	851.9	**1104.5**
Carnival	5040.3	4524.7	4941.7	**5366.6**
ElevatorAction	**47699.7**	11288.8	22721.8	43513.5
Gopher	6047.3	2902.2	3418.2	**13653.5**
Hero	29724.3	26787.1	29467.8	**33514.9**
Enduro	1040.3	834.0	889.4	**1150.4**
StarGunner	21175.7	17409.3	10907.0	**23223.6**
Zaxxon	14614.4	11556.6	**14839.9**	11637.1

K Scores on All ATARI Games

We show the mean performance of RPF on all 61 ATARI games in comparison to PPO and RND in Table 9. In total, RPF matches PPO in 34 games and matches RND in 30 games, indicating that it performs similarly overall to both methods. Finally, RPF performs better than both PPO and RND in 21 games, or approximately one-third of the games.

L Magnitudes of Intrinsic Rewards Across Games

One potential hypothesis for the observed performance improvement of RPF in games where RND improves on PPO is that the reward provided by RPF is an inductive bias that could be aligned with either novelty or certain games. We offer evidence to the contrary here, by showing in Table 10 that the average value of the intrinsic rewards of RPF is close to 0. Therefore, there is no strong correlation between whether performance is improved and the average intrinsic reward of RPF. This is to be expected: the reward function given by RPF resets periodically, without allowing the policy to update enough in time to maximize the RPF rewards.

Table 9: Mean game score (higher is better) over five random seeds of PPO, RND, and RPF on 61 ATARI games.

GAME	PPO	RND	RPF (OURS)
ADVENTURE	**0.0**	**0.0**	**0.0**
AIRRAID	36081.2	35409.2	**37809.1**
ALIEN	1913.0	**2219.7**	1978.1
AMIDAR	**1075.3**	774.7	851.9
ASSAULT	8426.3	11231.0	**12042.6**
ASTERIX	15093.2	**17486.5**	16858.2
ASTEROIDS	1351.7	**1455.7**	1301.7
BANKHEIST	1336.8	**1341.7**	1336.1
BATTLEZONE	**83508.0**	78586.0	55331.0
BEAMRIDER	7385.3	7605.1	**7922.9**
BERZERK	1132.4	1448.7	**1504.8**
BOWLING	18.0	24.2	**32.2**
BOXING	79.1	**80.0**	**79.9**
BREAKOUT	574.5	**578.9**	564.7
CARNIVAL	**5040.2**	4524.7	4941.7
CENTIPEDE	5953.2	**6755.0**	5517.2
CHOPPERCOMMAND	8713.6	8633.5	**9601.5**
CRAZYCLIMBER	**149496.8**	137229.1	136929.8
DEMONATTACK	5621.9	**9123.9**	6681.1
DOUBLEDUNK	-1.3	-1.3	**3.5**
ELEVATORACTION	**47699.7**	11288.8	22721.8
ENDURO	**1040.3**	834.0	889.4
FISHINGDERBY	36.5	26.5	**37.3**
FREEWAY	31.1	**33.4**	33.1
FROSTBITE	1080.0	1740.3	**2340.9**
GOPHER	**6047.3**	2902.2	3418.2
GRAVITAR	1700.8	**1895.0**	1789.1
HERO	**29724.3**	26787.1	29467.8
ICEHOCKEY	5.6	8.8	**10.6**
JAMESBOND	13344.4	**13706.1**	11771.9
JOURNEYESCAPE	-505.6	-659.8	**-489.6**
KABOOM	1875.0	**1879.1**	1864.8
KANGAROO	6125.3	8272.9	**8463.2**
KRULL	**9875.1**	9857.2	9564.0
KUNGFUMASTER	46230.8	**48777.3**	42361.3
MONTEZUMAREVENGE	0.1	**2496.0**	546.4
MSPACMAN	5007.7	**5371.9**	4491.0
NAMETHISGAME	11093.4	10313.7	**11273.0**
PHOENIX	8134.1	**10905.8**	8058.9
PITFALL	**0.0**	-4.8	-1.2
PONG	20.8	**20.9**	20.6
POOYAN	**5744.2**	5665.1	4628.0
PRIVATEEYE	96.7	**112.6**	99.6
QBERT	21943.2	22715.3	**24205.1**
RIVERRAID	10323.9	11837.7	**13476.5**
ROADRUNNER	46687.2	**59545.0**	36377.3
ROBOTANK	37.8	38.7	**46.2**
SEAQUEST	1463.8	**1989.0**	1752.1
SKIING	-12144.4	-11598.9	**-9918.0**
SOLARIS	**2473.5**	2255.2	2218.3
SPACEINVADERS	**1624.1**	1504.5	1586.0
STARGUNNER	**21175.7**	17409.3	10907.0
TENNIS	-0.1	**4.6**	-0.5
TIMEPILOT	19941.0	**21414.2**	20824.3
TUTANKHAM	198.4	**231.7**	221.1
UPNDOWN	276967.0	**311471.2**	304187.1
VENTURE	104.5	1370.1	**1392.9**
VIDEOPINBALL	368909.2	364614.4	**428003.5**
WIZARDOFWOR	11163.7	**11635.2**	10858.3
YARSREVENGE	92387.3	85691.1	**113623.3**
ZAXXON	14614.4	11556.6	**14839.9**

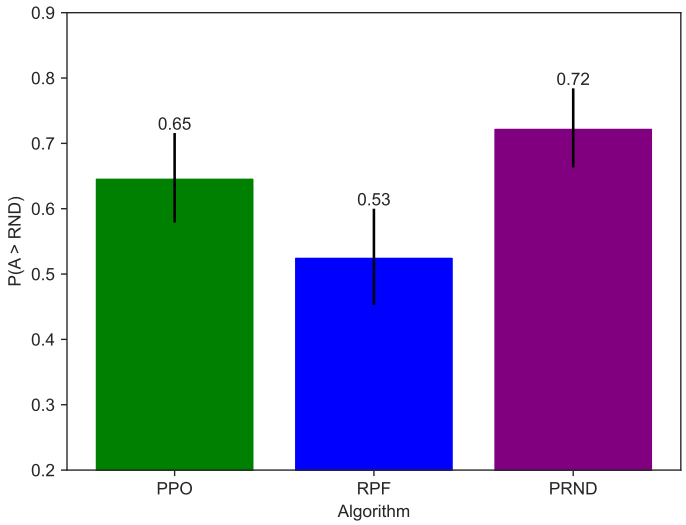

Figure 7: On the set of eight ATARI games (Appendix C.2) where we test how much RPF drops in performance compared to how much RND drops in performance, PPO significantly outperforms RND. PPO, shown in green, has a probability of improvement of 0.65 ± 0.06 over RND on these games, showing that PPO significantly improves over RND on these games. RPF, shown in blue, has a probability of improvement of 0.53 ± 0.07 over RND on this same set of games, showing that RPF and RND perform comparably on these games. However, applying potential shaping to RND, shown in purple, gives a probability of improvement over RND of 0.72 ± 0.06, performing even better than PPO. This suggests that exploration can still be beneficial in these games, and that RND is not exploring optimally.

M Limitations

As we show in Table 2, the performance of random reward functions greatly improves when using potential-based reward shaping. We hypothesize that this is because too much reliance on random reward functions can lead to the agent forgetting how to obtain previous rewards or to return to states it has visited in the past. Using potential shaping alleviates the issue of relying too much on the random intrinsic reward function to a large extent.

However, potential shaping does not completely fix this issue. As shown in Figure 9, RND significantly outperforms RPF in Montezuma's Revenge, the most popular ATARI game for benchmarking exploration methods [3, 5]. While RPF significantly improves over PPO in Montezuma's Revenge, we can see that the learning curve is unstable. We believe this is because of the nonstationary intrinsic reward function which does not decay over time. As the intrinsic reward function is constantly changing, the agent may be prevented from memorizing promising regions of its environment to explore. However, since the reward function for RND updates smoothly throughout training, we hypothesize that RND does not suffer from this issue.

Still, using only Montezuma's Revenge to evaluate the exploration ability of an agent can lead to unreliable results which do not hold in different environments [33]. Despite the fact that RND significantly outperforms RPF in Montezuma's Revenge, RND is only slightly better than RPF in aggregate, as we show in Section 4.3.

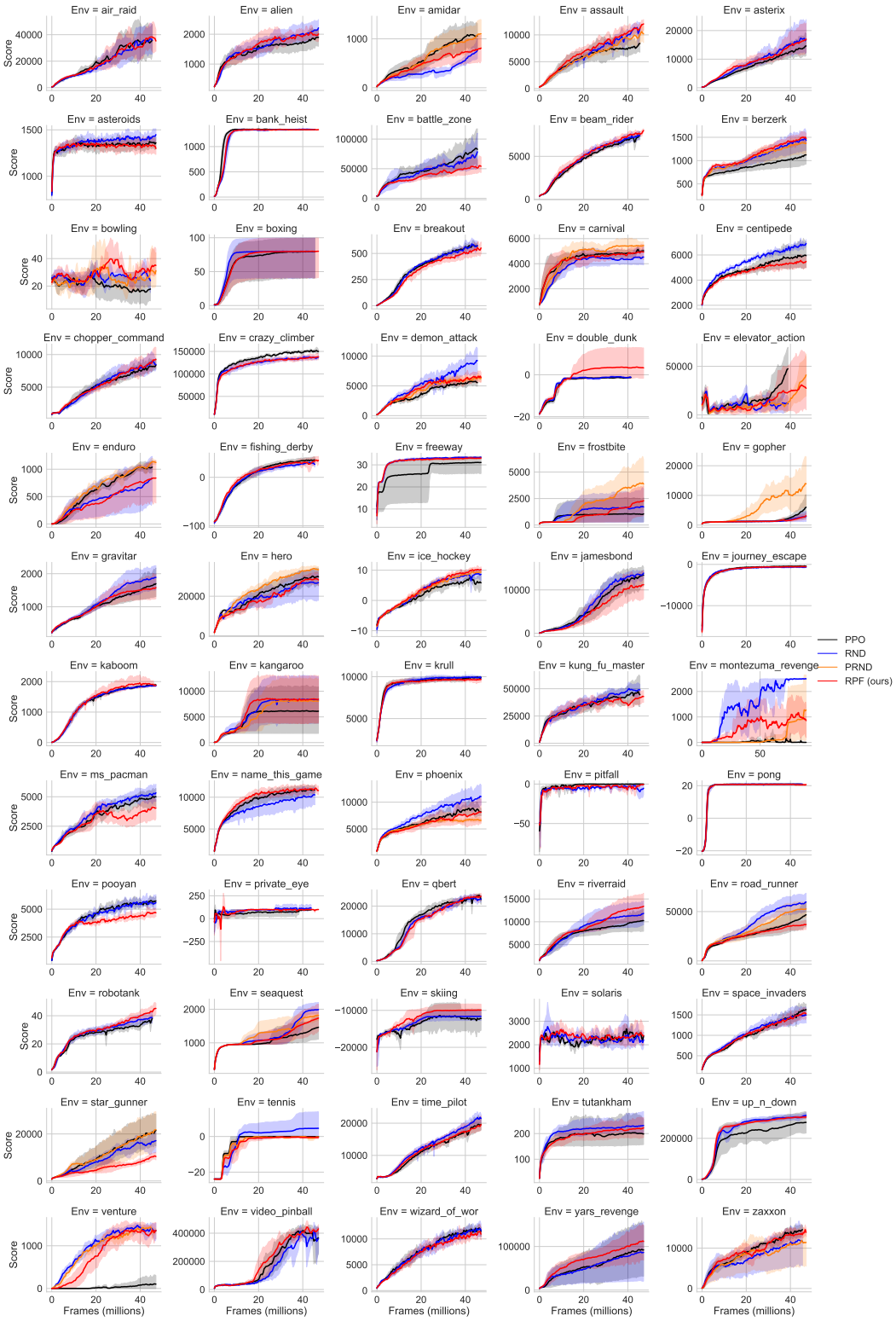

Figure 8: Learning curves of PPO, RND, RPF, and PRND on 60 ATARI games. Each curve displays the mean score of a method over 5 random seeds.

Table 10: The average intrinsic reward of RPF over each run is close to 0. There is no clear correlation between the sign of the intrinsic reward and whether RPF outperforms RND in a game.

GAME	AVERAGE INTRINSIC REWARD	RPF OUTPERFORMS RND
ASSAULT	$-3.58 \cdot 10^{-6}$	TRUE
BERZERK	$-1.82 \cdot 10^{-6}$	TRUE
BOWLING	$-5.67 \cdot 10^{-6}$	TRUE
DEMONATTACK	$1.81 \cdot 10^{-6}$	FALSE
FROSTBITE	$-8.39 \cdot 10^{-6}$	TRUE
ICEHOCKEY	$2.12 \cdot 10^{-6}$	TRUE
KANGAROO	$4.18 \cdot 10^{-6}$	TRUE
MONTEZUMAREVENGE	$-1.88 \cdot 10^{-6}$	FALSE
PHOENIX	$-4.26 \cdot 10^{-6}$	FALSE
ROADRUNNER	$2.89 \cdot 10^{-6}$	FALSE
SEAQUEST	$-9.34 \cdot 10^{-6}$	FALSE
TENNIS	$-2.91 \cdot 10^{-5}$	FALSE
VENTURE	$4.49 \cdot 10^{-6}$	TRUE

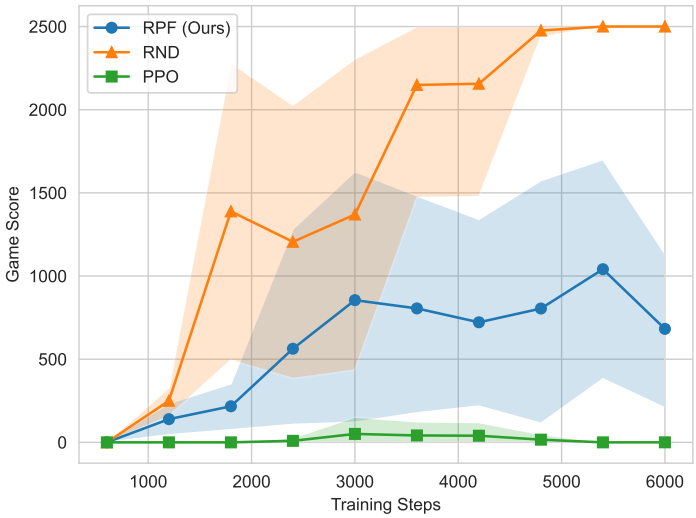

Figure 9: Learning curves of PPO, RPF, and RND on Montezuma's Revenge. While RPF clearly outperforms PPO, RPF is significantly outperformed by RND. The curves show the mean scores across all five runs per method. Shaded area is the 95% confidence interval, estimated using bootstrapping.