NovelD: A Simple yet Effective Exploration Criterion

Tianjun Zhang1 Huazhe Xu1 Xiaolong Wang2 Yi Wu3,5 Kurt Keutzer1

Joseph E. Gonzalez1 Yuandong Tian4

1University of California, Berkeley 2University of California, San Diego 3Tsinghua University 4Facebook AI Research 5Shanghai Qi Zhi Institute

Abstract

Efficient exploration under sparse rewards remains a key challenge in deep reinforcement learning. Previous exploration methods (e.g., RND) have achieved strong results in multiple hard tasks. However, if there are multiple novel areas to explore, these methods often focus quickly on one without sufficiently trying others (like a depth-wise first search manner). In some scenarios (e.g., four corridor environment in Sec. 4.2), we observe they explore in one corridor for long and fail to cover all the states. On the other hand, in theoretical RL, with optimistic initialization and the inverse square root of visitation count as a bonus, it won’t suffer from this and explores different novel regions alternatively (like a breadth-first search manner). In this paper, inspired by this, we propose a simple but effective criterion called NovelD by weighting every novel area approximately equally. Our algorithm is very simple but yet shows comparable performance or even outperforms multiple SOTA exploration methods in many hard exploration tasks. Specifically, NovelD solves all the static procedurally-generated tasks in Mini-Grid with just 120M environment steps, without any curriculum learning. In comparison, the previous SOTA only solves 50% of them. NovelD also achieves SOTA on multiple tasks in NetHack, a rogue-like game that contains more challenging procedurally-generated environments. In multiple Atari games (e.g., MonteZuma’s Revenge, Venture, Gravitar), NovelD outperforms RND. We analyze NovelD thoroughly in Mini-Grid and found that empirically it helps the agent explore the environment more uniformly with a focus on exploring beyond the boundary.

1 Introduction

Deep reinforcement learning (RL) has experienced significant progress over the last several years, with an impressive performance in games like Atari [41, 42], StarCraft [61], Go and Chess [57, 59]. However, its success often requires massive computational resources or manually designed dense rewards. The dense rewards are often impractical for real-world settings as they require a significant amount of task-specific domain knowledge.

An alternative approach is to allow the agent to explore the environment freely until it reaches the goal. While basic RL exploration criteria (e.g., ε-greedy) are quite simple, it fails to explore sufficiently in hard tasks. Modern works adopt various Intrinsic Reward (IR) designs to guide exploration in hard-exploration settings. However, we observe (Sec. 4.2) in our experiments if there are multiple...
We evaluate visitation count bonus explores the environment much more uniformly than the exploration criterion, and given by

\[ \text{transition function} \]

In a Markov Decision Process (MDP), we define state space \( S \), action space \( A \), and (non-deterministic) transition function \( T : S \times A \to P(S) \) where \( P(S) \) is the probability of next state given the current state and action. The goal is to maximize the expected reward

\[ R = \mathbb{E} \left[ \sum_{k=0}^{T} \gamma^k r_{t+k+1} \right] \]

where \( r_t \) is the reward and \( \gamma \) is the discount factor. In this paper, the total reward received at time step \( t \) is given by

\[ r_t = r^e_t + \alpha r^i_t \]

where \( r^e_t \) is the extrinsic reward given by the environment, \( r^i_t \) is the intrinsic reward from the exploration criterion, and \( \alpha \) is a scaling hyperparameter.

### 2 Background

In a Markov Decision Process (MDP), we define state space \( S \), action space \( A \), and (non-deterministic) transition function \( T : S \times A \to P(S) \) where \( P(S) \) is the probability of next state given the current state and action. The goal is to maximize the expected reward

\[ R = \mathbb{E} \left[ \sum_{k=0}^{T} \gamma^k r_{t+k+1} \right] \]

where \( r_t \) is the reward and \( \gamma \) is the discount factor. In this paper, the total reward received at time step \( t \) is given by

\[ r_t = r^e_t + \alpha r^i_t \]

where \( r^e_t \) is the extrinsic reward given by the environment, \( r^i_t \) is the intrinsic reward from the exploration criterion, and \( \alpha \) is a scaling hyperparameter.
We show the distinct preference of exploration directions between RND and NovelD. We also see that (Trajectory B), which typically show a profile of stable novelty measure until a sudden boost when it ventures into an unknown region. On the other hand, count-based approaches can be trapped within explored regions where the trajectories have low but consistent novelty rewards (Trajectory A).

3 Exploration via Novelty Difference

3.1 NovelD Intrinsic Reward Criterion

NovelD is a meta-criterion that can be applied on top of any novelty measure \( \text{novelty}(s) \) that tells how familiar a state \( s \) is for the agent. Formally, along a trajectory, we give agent IR if the previous state \( s_t \) has been sufficiently explored but \( s_{t+1} \) is not:

\[
    r^i(s_t, a_t, s_{t+1}) = \max \{ \text{novelty}(s_{t+1}) - \alpha \cdot \text{novelty}(s_t), 0 \},
\]

(1)

Here \( \alpha \) is a scaling factor. Intuitively, if we define \( \{ s : \text{novelty}(s) \leq m \} \) to be an explored region, then NovelD gives intrinsic reward (IR) to the agent when it explores beyond the boundary of explored regions. Note that IR is clipped to avoid negative IR if the agent transits back from a novel state to a familiar state. From the equation, only crossing the frontier matters to the intrinsic reward; if both \( s_{t+1} \) and \( s_t \) are novel or familiar, their difference would be small. For each trajectory going towards the frontier/boundary, NovelD assigns an approximately equal IR, regardless of the length (see Sec. 3.2). Like RIDE [11], in our actual implementation, partial observation \( o_t \) is used instead of the true state \( s_t \), when \( s_t \) is not available.

Episodic Restriction on Intrinsic Reward (ERIR). Simply using Eqn. 1 to guide exploration can result in the agent going back and forth between novel states \( s_{t+1} \) and their previous states \( s_t \). RIDE [11] avoids this by scaling the intrinsic reward \( r^i(s) \) by the inverse of the episodic state visitation counts. NovelD puts a more aggressive restriction: the agent is only rewarded when it visits the state \( s \) for the first time in an episode. Thus, the intrinsic reward of NovelD becomes:

\[
    r^e(s_t, a_t, s_{t+1}) = \max \{ \text{novelty}(s_{t+1}) - \alpha \cdot \text{novelty}(s_t), 0 \} \cdot \mathbb{I}\{N_e(s_{t+1}) = 1\}
\]

(2)

\( N_e \) here stands for episodic state count and is reset every episode. In contrast, the novelty measure \( \text{novelty}(\cdot) \) is a life-long counter throughout training.

Novelty Measure using Random Network Distillation. How to accurately measure the novelty of a state \( s \) in large-scale stochastic environments remains an open problem [2]. Similar to RND [11], we use the difference between a random fixed target network \( \phi \) and a trainable predictor network \( \phi'_w \) as a novelty measure:

\[
    \text{novelty}(s_t) = \text{novelty}(s_t; w) := ||\phi(s_t) - \phi'_w(s_t)||_2,
\]

(3)

Once the novelty score for state \( s_t \) is computed, we minimize \( \text{novelty}(s_t; w) \) with respect to \( w \) and perform a one-step weight update for \( w \), which is the parameter of the predictor network \( \phi'_w \). Therefore, when \( s_t \) is visited again, the quantity \( \text{novelty}(s_t; w) \) will be lower, showing that the state has been seen in the past.

3.2 Conceptual Advantages of NovelD over Existing Criteria

We show the distinct preference of exploration directions between RND and NovelD. We also see that NovelD is a consistent algorithm, the intrinsic reward converges to zero after sufficient exploration.

Explore via Novelty Difference. Fig 2 shows the conceptual comparison between NovelD (Eq 2) and exploration with novelty measure alone (e.g., RND). We could clearly see their distinctive preferences: our criterion tends to reward more the trajectories that move out of the explored regions (Trajectory B), which typically show a profile of stable novelty measure until a sudden boost when it ventures into an unknown region. On the other hand, count-based approaches can be trapped within explored regions where the trajectories have low but consistent novelty rewards (Trajectory A).
We mainly use three challenging environments from MiniGird:

We evaluate NovelD on challenging procedurally-generated environment MiniGrid \[ \text{Asymptotic Inconsistency.} \]

This is because when the embedding network \( \psi \) is a learned embedding network \( \psi \) converges after sufficient exploration, the agent can always obtain non-zero IR if a significant change occurs in state representation (e.g., opening a door or picking up a key in MiniGrid). Therefore, the learned policy does not maximize the extrinsic reward \( r^e \), deviating from the goal of RL. Automatic curriculum approaches \( \text{Eq. 4} \) have similar issues due to an ever-present IR. In contrast, our Eq. \( 4 \) is asymptotically consistent as \( r^e \to 0 \) when \( N \to \infty \).

## 4 Experiments

We evaluate NovelD on challenging procedurally-generated environment MiniGrid \( \text{[13]} \), several hard-exploration scenarios in Atari games, and NetHack \( \text{[52]} \) with sparse rewards (i.e., get reward only when reaching the final goal). For all the experiments and all exploration approaches (including NovelD), we use PPO \( \text{[55]} \) as the base RL algorithm and add intrinsic reward specified by various methods, to encourage exploration.

In MiniGrid, we compare NovelD with RND \( \text{[11]} \), ICM \( \text{[19]} \), RIDE \( \text{[52]} \) and AMIGo \( \text{[12]} \). We only evaluate AMIGo for 120M steps in our experiments. The algorithm obtains better results when trained for 500M steps as shown in \( \text{[12]} \). For all other baselines, we follow the same training paradigm from \( \text{[52]} \). Mean and standard deviation across four runs of different seeds are computed.

NovelD solves all the static tasks provided by MiniGrid. In contrast, all the baselines end up with zero rewards on half of the tasks we tested. In NetHack, NovelD achieves SOTA with a large margin over baselines (IMPALA \( \text{[19]} \) without exploration bonus and RND).

### 4.1 MiniGrid Environments

We mainly use three challenging environments from MiniGird: Multi-Room (MR), Key Corridor (KC) and Obstructed Maze (OM). We use these abbreviations for the rest of the paper (e.g., OM2Dlh stands for ObstructedMaze2Dlh). Fig. \( \text{[3]} \) shows one example of a rendering on OM2Dlh as well as all the environments we tested with their relative difficulty.

In MiniGrid, all environments are of size \( K \times K \) (\( K \) is environment-specific) where each tile contains an object: wall, door, key, ball, chest, etc. The action space is defined as turn left, turn right, move forward, pick up an object, drop an object, and toggle an object (e.g., open or close a door). MR consists of a series of rooms connected by doors and the agent must open the door to get to the next room. Success is achieved when the agent reaches the goal. In KC, the agent has to explore the environment to find the key and open the door along the way to achieve success. OM is the hardest: the doors are locked, the keys are hidden in boxes, and doors are obstructed by balls.

**Results.** NovelD manages to solve all the static environments in MiniGrid. In contrast, all baselines solve only up to medium-level tasks and fail to make any progress on more difficult ones. Note that some medium-level tasks we define here are categorized as hard tasks in RIDE and AMIGo (e.g., KC5R3 is labeled as “KCHard” and KCS5R3 is labeled as “KCHarder” in their works). Fig. \( \text{[4]} \) shows the results of our experiments. Half of the environments (e.g., KCS6R3, OM1Q) are extremely hard and all the baselines fail. In contrast, NovelD easily solves all listed above without any curriculum learning. We also provide the testing performance for NovelD and all baselines in Tab. \( \text{[5]} \). The results are averaged across four seeds and 32 random initialized environments.
Figure 4: Results for various hard exploration environments from MiniGrid. NovelD successfully solves all the environments while all other baselines only manage to solve two to three relatively easy ones.

Table 1: Visitation counts for the toy corridor environment after 3K episodes. NovelD explores corridors more uniformly than count-based approaches.

<table>
<thead>
<tr>
<th></th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>40</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>3.54/3.53</td>
</tr>
<tr>
<td>Count-Based</td>
<td>66K ± 28K</td>
<td>8K ± 4K</td>
<td>23K ± 15K</td>
<td>11K ± 15K</td>
<td>1.06 ± 0.99</td>
</tr>
<tr>
<td>NovelD</td>
<td>28K ± 2K</td>
<td>28K ± 6K</td>
<td>28K ± 4K</td>
<td>28K ± 8K</td>
<td>1.97 ± 0.62</td>
</tr>
<tr>
<td>RND</td>
<td>0.2K ± 0.2K</td>
<td>70K ± 10K</td>
<td>0.2K ± 0.10K</td>
<td>28K ± 40K</td>
<td>0.24 ± 0.28</td>
</tr>
<tr>
<td>NovelD 2D</td>
<td>27K ± 6K</td>
<td>27K ± 5K</td>
<td>31K ± 12K</td>
<td>28K ± 8K</td>
<td>1.96 ± 0.48</td>
</tr>
</tbody>
</table>

Table 2: Entropy of the visitation counts of each room. Such state distribution of NovelD is much more uniform than RND.

<table>
<thead>
<tr>
<th>Room</th>
<th>0.2M</th>
<th>0.5M</th>
<th>2.0M</th>
<th>5.0M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room1</td>
<td>3.48</td>
<td>3.54</td>
<td>3.41</td>
<td>3.53</td>
</tr>
<tr>
<td>Room2</td>
<td>2.87</td>
<td>3.33</td>
<td>3.51</td>
<td>3.56</td>
</tr>
<tr>
<td>Room3</td>
<td>/</td>
<td>/</td>
<td>3.35</td>
<td>3.56</td>
</tr>
<tr>
<td>Room4</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
</tbody>
</table>

* Results are presented in the order of “RND / NovelD”.

Figure 5: Normalized visitation counts \(N(s_t)/Z\) (\(Z\) is a normalization constant) for the locations of agents. NovelD successfully explores all rooms within 4.6M steps while RND gets stuck in the fifth room within 9.8M steps.

Multi-Room environments are relatively easy in MiniGrid. However, all the baselines except RIDE fail. As we increase the room size and number (e.g., MRN12S10), NovelD can achieve the goal quicker than RIDE. Our method quickly solves these environments within 20M environment steps.

On Key Corridor environments, RND, AMIGo, RIDE, IMPALA and NovelD successfully solves KCS3R3 while ICM makes reasonable progress. However, when we increase the room size (e.g., KCS5R3), none of the baseline methods work. NovelD manages to solve these environments in 40M environment steps. The agent demonstrates the ability to explore the room and finds the corresponding key to open the door in a randomized, procedurally generated environment.

Obstructed Maze environments are also tricky. As shown in Fig. 4, RIDE and RND manage to solve the easiest task OM2Dlh which doesn’t contain any obstructions. In contrast, NovelD not only rapidly solves OM2Dlh, but also solves four more challenging environments including OMFull. These environments have obstructions blocking the door (as shown in Fig. 3) and are much larger than OM2Dlh. Our agent learns to move the obstruction away from the door to open the door and enter the next room in these environments. This “skill” is hard to learn since no extrinsic reward is assigned to moving the obstruction. However, learning the skill is critical to achieving the goal.
Figure 6: IR heatmaps for three-room environment. We can see that the high-IR region of NovelD has a more clear focus (the frontier) comparing to RND.

4.2 Analysis of Intrinsic Reward in Pure Exploration

We observe that NovelD leads to a more focused exploration at the boundary and broader state coverage by only using IR in the environment (“pure exploration” setting).

**Broader State Coverage in Long-Corridor Environment.** We design a toy environment with four disconnected corridors with lengths 40, 10, 30, and 10 respectively. The agent starts from the common entry of the corridors and picks which corridor to enter. In this example, there is no extrinsic reward, and the agent’s exploration is guided by IR. We use deep Q-learning (i.e., $Q$ function is parameterized by a deep network) and try various IRs. This includes tabular IR (count-based and NovelD tabular) and neural-network-approximated IR (RND and NovelD) respectively for this experiment. We remove clipping from NovelD for a fair comparison. Tab. 1 shows the visitation counts across 4 runs w.r.t. each corridor after 600 episodes of training. NovelD tabular explores all four corridors uniformly. On the other hand, count-based approaches focus on only two out of four corridors. NovelD also shows much more stable performance across runs as the standard deviation is much lower than RND.

**Visitation Counts Analysis in MiniGrid.** To study whether NovelD yields a broader state coverage in MiniGrid, we test NovelD and RND in a fixed (instead of procedurally generated for simplicity) MR87S8 environment. It contains 7 rooms connected by doors. We define two metrics to measure the effectiveness of an exploration strategy: (1) visitation counts $N(s)$ at every state over training, and (2) entropy of visitation counts in each room: $H(\rho'(s))$ where $\rho'(s) = \frac{N(s)}{\sum_{s' \in S_r} N(s')}$. Fig. 5 shows the heatmap of normalized visitation counts $N(s_t)/Z$, where $Z$ is the normalization constant. At first, RND enters room 2 faster than NovelD. However, NovelD consistently makes progress exploring new states and discovers all the rooms in 5M steps, while RND gets stuck in room 5 even trained with 10M steps.

In Tab. 2, the entropy of distribution in each room $H(\rho'(s))$ for NovelD is larger than that of RND. This suggests that NovelD encourages a more uniform exploration of the states than RND.

**Intrinsic Reward Heatmap Analysis in MiniGrid.** We visualize the IR produced in our algorithm and study whether it has a higher focus on the boundary. We generate the plot by running the policy from different checkpoints for 2K steps and plot the IR associated with each state in the trajectory. States that do not receive IR from the sampled trajectories are left blank. From Fig. 6, we can see that before opening the door to the 2nd room, the NovelD IR is only high in the first one. As soon as the agent opens the door, the boundary of the explored region gets pushed to the next one. This can also be illustrated in the figure that regions with high IR are also changed correspondingly. In other words, this shows that when the door between two rooms becomes a bottleneck for exploration, IR of NovelD focuses on solving this. A similar phenomenon happens between the 2nd and 3rd rooms. In contrast, the IR in RND is more spread out (e.g., both 1st and 2nd room in the second figure).

4.3 Ablation Study

**Episodic Restriction on IR and Clipping in NovelD.** To illustrate the importance of exploring via novelty difference, we compare NovelD with a modified version of RND: RND+ERIR. ERIR only gives RND intrinsic reward when visiting a new state for the first time in an episode. As shown in Fig. 7, the ERIR also improves the performance of RND, but NovelD has much better results. We remove ERIR from NovelD and performance also drops.
Figure 7: Ablation Study on NovelD with different scaling factor $\alpha$ and clipping threshold $\beta$. Results show that there exist an optimal choice for $\alpha = 0.5$ and $\beta = 0$.

Figure 8: Ablation for NovelD under different representations. Successor representation sometimes works better than random feature, but not consistently.

Figure 9: Results for tasks on NetHack. NovelD achieves the SOTA results comparing to RND and IMPALA.

The optimal scaling ratio and clipping factor. We did an ablation on how different scaling ratio $\alpha$ and clipping factor $\beta$ is affecting the performance of NovelD. Specifically, with the hyperparameter $\alpha$ and $\beta$, the intrinsic reward now takes the form:

$$r^i(s_t, a_t, s_{t+1}) = \max [\text{novelty}(s_{t+1}) - \alpha \cdot \text{novelty}(s_t), \beta],$$

(4)

Results show that $\alpha = 0.5$ and $\beta = 0$ works the best. We also compare with AK18 [15] since they use a difference-based IR as well. AK18 achieves zero-reward so we don’t run them on all the tasks.

NovelD under Different Representations. To study how NovelD performs under different representations, we set both the target $\phi(v)$ and predictor network $\phi'(v)$ in RND to be MLP and uses the same CNN-based encoders $v = f(s)$ to project the image to a vector for both of them. We trains another CNN-based encoder $f'(s)$ according to different criteria and replace $f(s)$ with $f'(s)$ periodically. Note that once we changed the representation of states, the novelty measure changes. So we store all the samples and retrain the predictor and target from scratch. We tries four different representations of the encoder: ICM [49] tries to learn a compact representation that are informative about state transition, Random embedding, DBC [62] follows the criterion of aggregating model-irrelevant states and Successor Features [37] of random embedded states captures the expected future features ($f'(s_t) = \mathbb{E}[\sum_{t=0} \gamma^t \phi(s_t)]$ where $\phi$ is a random embedding network).

We see that from Fig 8, DBC and ICM got zero reward since their representations are very compact and every state $s$ look similar to the RND networks. Thus, the predictor network quickly approximates the target network well and induces zero IR. For successor representation of random embedding, it works better than random in KC series of maps but worse in MR maps.

4.4 Atari Games

Atari games are common benchmarks for RL exploration. We test NovelD on multiple hard exploration tasks (e.g., MonteZuma’s Revenge, Gravitar, Solaries). We use the same paradigm as RND [11]
and the same set of hyperparameters on 128 parallel environments. We use the code base from RND and 84x84 image of the game as input. In NovelD, same RNN network architecture is used for all the tasks (except for the fourth figure Fig. 11, we use simple CNN architecture). The baseline algorithm being used is PPO. Due to the computation limit, we didn’t run NovelD to 2 billion steps as originally reported in the RND paper. We mark the performance of RND at the similar steps as red dot. Note that we didn’t compare with SOTA method Go-Explore since it involves a lot of hand-tuned hyperparameters and highly incorporates domain knowledge. For another strong baseline NGU, since there is no open-sourced implementation online, it is hard for us to evaluate their code and do a fair comparison with them.

In Fig. 11, we can see that using CNN-based model, NovelD achieves approximately 10000 external rewards after two Billion frames while the performance reported in RND is around 6700. When using an RNN-based model, NovelD reached around 13000 external rewards in 100K updates while RND only achieves 4400. In Fig. 10, we also see that NovelD manage to explore 23 rooms in the given steps while RND only explores 17. For other games (e.g., Venter, Gravitor and Solaries), NovelD is consistently better than RND in the given steps, achieving 500 to 1000 more rewards.

4.5 The NetHack Learning Environment

We also evaluate NovelD in NetHack Learning Environment, a more challenging and realistic environment. The player needs to descend over 50 procedurally-generated levels to the bottom and find “Amulet of Yendor” in the dungeon. The procedure can be described as first retrieve the amulet, then escape the dungeon and finally unlock five challenging levels (the four Elemental Planes and the Astral Plane). We test NovelD on a set of tasks with tractable subgoals in the game: Staircase: navigating to a staircase to the next level, Pet: reaching a staircase and keeping the pet alive, Gold: collecting gold, Score: maximizing the score in the game, Scout: scouting to explore unseen areas in the environment, and Oracle: finding the oracle (an in-game character at level 5-9).

Results in Fig. 9 show that NovelD surpasses PPO+RND and IMPALA (without exploration bonus) on all tasks. Especially on Pet, NovelD outperforms the other two with a huge margin. This again illustrates the strong performance of NovelD in an environment with huge action spaces and long-horizon reward. Oracle is the hardest task and no approaches are able to find the oracle and obtain a reward of 1000. NovelD still manages to find a policy with less negative rewards (i.e., a penalty of taking actions that do not lead to game advancement, like moving towards a wall).

5 Discussion

On NetHack, we do not report RIDE performance. Our attempt fails and the authors verify it doesn’t work yet on NetHack.
Deterministic and Stochastic Environments. NovelD can both work with deterministic (MiniGrid) and stochastic (NetHack and Atari) environments. Although reaching the frontier of a stochastic environment is itself a hard problem, by putting a high IR on the boundary states and training the agent with RL algorithms, the policy will prefer to reach those frontier states more frequently. In such case, NovelD also manage to achieve its exploration purpose. In addition to that, we also test NovelD in stochastic environments like NetHack and Atari games, showing superior performance. Finally, the hashing-based ERIR do have some potential issue in stochastic environments. However, in all of the tasks we tested, it doesn’t show a clear problem of directly applying it in stochastic environments.

Noisy TV problem. As mentioned in [11], curiosity-based and count-based criterion will get stuck in nosity-TV problem. However, RND manages to mitigate this problem [11]. Since we also adopt RND as the novelty measure, NovelD also shows the resilient to the problem. We also test NovelD in a manually-made noisy-TV setting in MiniGrid introduced in RIDE [52], where there always some blocks changing color at every time step. NovelD still gain a strong performance even under this setting. In addition, empirically we don’t see the performance degrade of NovelD due to the noisy-TV problem in all our experiments including MiniGrid, Atari Games and NetHack.

Limitations. We don’t test NovelD in some continuous RL domains. Especially, directly applying ERIR to that for those tasks (e.g., some robotics tasks) might lead to some performance degrade. We leave a more general solution of this to future work.

6 Related Work

In addition to the two criteria (count-based and state-diff based) mentioned above, another stream of defining IRs is curiosity-based. The main idea is to encourage agents to explore areas where the prediction of the next state from the current learned dynamical model is wrong. Dynamic-AE [50] computes the distance between the predicted and the real state on the output of an autoencoder, ICM [49] learns the state representation through a forward and inverse model and EMI [52] computes the representation through maximizing mutual information $I(s,a; s')$ and $I(s,s';a)$.

Another line of research is using information gain to reward the agent. VIME [28] uses a Bayesian network to measure the uncertainty of the learned model. Later, to reduce computation, a deterministic approach has been adopted [11]. Other works also propose to use an ensemble of networks for measuring uncertainty [24]. We can also reward the agent by Empowerment [25], [26], [27], prioritizing the states that the agent can take control through its actions. It is different from state-diff: if $s_{t+1}$ differs from $s_t$ but not due to agent’s choice of actions, then the empowerment at $s_t$ is zero. Other criteria exist, e.g., diversity [20], feature control [20], [46], [16] or the KL divergence between current distribution over states and a target distribution of states [55].

Outside of intrinsic reward, researchers have proposed to use randomized value functions to encourage exploration [46], [27], [47]. Adding noise to the network is also shown to be effective [23], [51]. There has also been effort putting to either explicitly or implicitly separate exploration and exploitation [14], [22], [50]. Go-Explore series [17], [13] also fall in this category. We might also set up different goals for exploration [26], [43], [3]. Another stream of research studies exploration problem using model-based approaches [8], [44], [51]. They either assume a world model is given or attempt to learn it, which is often impractical when the environment is complex. However, NovelD is completely model-free and can be applied widely to complicated environments.

Curriculum learning [17] has also been used to solve hard exploration tasks. The curriculum can be explicitly generated by: searching the space [54], teacher-student setting [59], increasing distance between the starting point and goal [29] or using a density model to generate a task distribution for the meta learner [21]. NovelD never explicitly generates a curriculum.

Figure 12: Results for NovelD under the Noisy-TV setting introduced in RIDE [52]. NovelD still have strong performance under the noisy TV environment.
7 Conclusion

In this work, inspired by breadth-first search, we introduce a new criterion for intrinsic reward (IR) that encourages exploration using a regulated difference of novelty measure using RND. Based on the criterion, our algorithm NovelD successfully solves all static tasks in a procedurally generated MiniGrid environment, which is a significant improvement over the previous SOTA. In multiple environments, NovelD shows a broader state coverage with a focus of IR on the boundary states. We also evaluate NovelD on NetHack, a more challenging task. NovelD outperforms all the baselines by a large margin. In summary, this simple criterion and the ensuing algorithm demonstrate effectiveness in solving sparse reward problems in reinforcement learning, opening up new opportunities to many real-world applications.

8 Acknowledgements

This project occurred under the BAIR Commons at UC-Berkeley and we thanks Commons sponsors for their support. In addition to NSF CISE Expeditions Award CCF-1730628, UC Berkeley research is supported by gifts from Alibaba, Amazon Web Services, Ant Financial, CapitalOne, Ericsson, Facebook, Futurewei, Google, Intel, Microsoft, Nvidia, Scotiabank, Splunk and VMware.

References


Checklist

1. For all authors...

   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes] We claim to propose efficient exploration criterion that achieves strong results. The experiments are conducted in MiniGrid, NetHack and MonteZuma’s Revenge.

   (b) Did you describe the limitations of your work? [Yes] We discuss limitations in the Discussion section.
(c) Did you discuss any potential negative societal impacts of your work? [No] We do not foresee any obvious negative societal impacts from our work, which contributes a general-purpose exploration algorithm.

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A] We don’t have theorem part.
   (b) Did you include complete proofs of all theoretical results? [N/A] We don’t have theorem part.

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We upload the code in supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We include the training detail in supplemental material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We include the error bars for experiments.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you’re using/curating? [No] We use publicly available datasets/tasks.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] We don’t use data of this sort.

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]