Efficient Exploration by Novelty-Pursuit

Ziniu Li[1,2] and Xiong-Hui Chen[3]*

[1] Polixir, Nanjing, China
ziniu.li@polixir.ai
[2] The Chinese University of Hong Kong, Shenzhen, Shenzhen, China
ziniuli@link.cuhk.edu.cn
[3] National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
chenxh@lamda.nju.edu.cn

Abstract. Efficient exploration is essential to reinforcement learning in tasks with huge state space and long planning horizon. Recent approaches to address this issue include the intrinsically motivated goal exploration processes (IMGEP) and the maximum state entropy exploration (MSEE). In this paper, we propose a goal-selection criterion in IMGEP based on the principle of MSEE, which results in the new exploration method *novelty-pursuit*. Novelty-pursuit performs the exploration in two stages: first, it selects a seldom visited state as the target for the goal-conditioned exploration policy to reach the boundary of the explored region; then, it takes random actions to explore the non-explored region. We demonstrate the effectiveness of the proposed method in environments from simple maze environments, MuJoCo tasks, to the long-horizon video game of SuperMarioBros. Experiment results show that the proposed method outperforms the state-of-the-art approaches that use curiosity-driven exploration.

Keywords: Reinforcement learning · Markov decision process · Efficient exploration.

1 Introduction

Reinforcement learning (RL) [39, 40] is a learning paradigm that an agent interacts with an unknown environment to improve its performance. Since the environment transition is unknown in advance, the agent must explore (e.g., take new actions) to discover states with positive rewards. Hence, efficient exploration is important to learn a (near-) optimal policy in environments with huge state space and sparse rewards [44], where deep-sights planning behaviors are required [29]. In those cases, simple exploration strategies like ϵ-greedy are inefficient due to time-uncorrelated and uncertainty-unaware behaviors [2].

To avoid insufficient exploration, advanced curiosity-driven approaches encourage diverse actions by adding the uncertainty-based exploration bonus on

* The two authors contributed equally to this work.

the environment reward [4, 7, 30, 31, 37]. In addition, recently proposed methods to tackle this issue include the intrinsically motivated goal exploration processes (IMGEP) [14], and the maximum state entropy exploration (MSEE) [18]. In particular, IMGEP was biologically-inspired to select intrinsically interesting states from the experience buffer as goals and to train a goal-conditioned (goal-input) exploration policy to accomplish the desired goals. On the other hand, MSEE aimed to search for a policy such that it maximizes the entropy of state distribution.

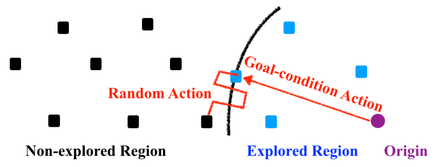


Fig. 1. Illustration for the proposed method. First, a goal-conditioned policy plans to reach the exploration boundary; then, it performs random actions to discover new states.

In this paper, we propose a goal-selection criterion for IMGEP based on the principle of MSEE, which results in the new exploration method *novelty-pursuit*. Abstractly, our method performs in two stages: first, it selects a novel state that was seldom visited as the target for the goal-conditioned exploration policy to reach the boundary of the explored region; subsequently, it takes random actions to discover new states. An illustration is given in Figure 1. Intuitively, this process is efficient since the agent avoids meaningless exploration within the explored region. Besides, the exploration boundary will be expanded further as more and more new states are discovered. To leverage good experience explored by the goal-conditioned policy, we also train an unconditioned policy that exploits collected experience in the way of off-policy learning.

We conduct experiments on environments from simple maze environments, MuJoCo tasks, to long-horizon video games of SuperMarioBros to validate the exploration efficiency of the proposed method. In particular, we demonstrate that our method can achieve a large state distribution entropy, which implies our exploration strategy prefers to uniformly visit all states. Also, experiment results show that our method outperforms the state-of-the-art approaches that use curiosity-driven exploration.

2 Related work

In addition to the ϵ-greedy strategy, simple strategies to remedy the issue of insufficient exploration include injecting noise on action space [23, 26] or parameter space [8, 10, 15, 24, 34], and adding the policy's entropy regularization

[25, 36]. For the tabular Markov Decision Process, there are lots of theoretical studies utilizing upper confidence bounds to perform efficient exploration [21, 22, 38]. Inspired by this, many deep RL methods use curiosity-driven exploration strategies [4, 7, 30, 31, 37]. In particular, these methods additionally add the uncertainty-based exploration bonus on the environment reward to encourage diverse behaviors. Moreover, deep (temporally extended) exploration via tracking the uncertainty of value function was studied in [29]. Maximum (policy) entropy reinforcement learning, on the other hand, modifies the original objective function and encourages exploration by incorporating the policy entropy into the environment reward [17, 28].

Our method is based on the framework of intrinsically motivated goal exploration processes (IMGEP) [3, 14, 32]. Biologically inspired, IMGEP involves the following steps: 1) selecting an intrinsically interesting state from the experience buffer as the goal; 2) exploring with a goal-conditioned policy to accomplish the target; 3) reusing experience by an exploitation policy that maximizes environment rewards. Obviously, the performance of exploitation policy heavily relies on samples collected by the goal-conditioned exploration policy so that the criterion of intrinsic interest is crucial for IMGEP. As reminiscent of IMGEP, Go-Explore [12] used the heuristics based on visitation counts and other domain knowledge to select goals. However, different from the basic framework of IMGEP, Go-Explore directly reset environments to target states, upon which it took random actions to explore. As a result, it achieved dramatic improvement in the challenging exploration task of Montezuma's Revenge. However, the requirement that the environment is resettable (or the environment transition is deterministic), together with many hand-engineering designs, clearly restricts Go-Explore's applications. The exploration scheme of our method in Figure 1 is similar to Go-Explore, but our method does not require environments to be resettable or deterministic.

Recently, [18] introduced a new exploration objective: maximum state entropy. Since each policy induces a (stationary) state distribution, [18] provided a provably efficient reinforcement learning algorithm under the tabular MDP to search for an optimal policy such that it maximizes the state (distribution) entropy. The such-defined optimal policy is conservative to visit uniformly all states as possible in unknown environments. Inspired by this principle, we propose to select novel states as goals for the goal-conditioned exploration policy in IMGEP. Note that [18] mainly focused on the pure exploration problem while we additionally train an exploitation policy that leverages collected experience in the way of off-policy learning to maximize environment rewards.

Finally, we briefly review recent studies about how to quickly train a goal-conditioned policy since it is a core component of our method. Particularly, [35] proposed the universal value function approximator (UVFA) and trained it by bootstrapping from the Bellman equation. However, this training procedure is still inefficient because goal-conditioned rewards are often sparse (e.g. 1 for success and 0 for failure). To remedy this issue, [1] developed the hindsight experience replay (HER) that replaced the original goal with an achieved goal. As a result, the agent can receive positive rewards even though it does not accomplish the original goal. In this way, learning on hindsight goals may help generalize for unaccomplished goals. Moreover, [13] used a generator neural network to adaptively produce artificial feasible goals to accommodate different learning levels, which servers as an implicit curriculum learning.

3 Background

In the standard reinforcement learning (RL) framework [39, 40], a learning agent interacts with an Markov Decision Process (MDP) to improve its performance via maximizing cumulative rewards. The sequential decision process is characterized as follows: at each timestep t, the agent receives a state s_t from the environment and selects an action a_t from its policy $\pi(s, a) = \Pr\{a = a_t | s = s_t\}$; this decision is sent back to the environment, and the environment gives a reward signal $r(s_t, a_t)$ and transits to the next state s_{t+1} based on the state transition probability $p_{ss'}^{a} = \Pr\{s' = s_{t+1} | s = s_t, a = a_t\}$. This process repeats until the agent encounters a terminal state after which the process restarts.

The main target of reinforcement learning is to maximize the (expected) episode return $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$, where $\gamma \in (0, 1)$ is a discount factor that balances the importance of future rewards and the expectation is taken over the stochastic process induced by the environment transition and the action selection. Since the environment transition (and possibly the reward function) is unknown in advance, the agent needs exploration to discover valuable states with positive rewards. Without sufficient exploration, the agent will be stuck into the local optimum by only learning from sub-optimal experience [44].

4 Method

As demonstrated in Figure 1, our exploration method called *novelty-pursuit* runs in two stages: first, it selects a novel state as the goal for the goal-conditioned exploration policy to reach the boundary of the explored region; then, it takes random actions to explore the non-explored region. Again, newly observed states will be set as the desired goals in the next round. As this process repeats, the exploration boundary will be expanded further and the whole state space will probably be explored. To leverage good experience explored by the goal-conditioned policy, we also train an unconditioned policy that exploits collected experience in the way of off-policy learning. We outline the proposed approach in Algorithm 1 (for simplicity, the procedure of training the exploitation policy is omitted).

In the following parts, we focus on the detailed implementation of the proposed method. Firstly, goal-selection in complicated tasks (e.g., tasks with high-dimensional visual inputs) is given in Section 4.1. Then, we introduce the accelerating training techniques for goal-conditioned exploration policy in Section 4.2. Finally, we discuss how to effectively distill a good exploitation policy from collected experience in Section 4.3.

Algorithm 1 Exploration by novelty-pursuit

```
Input: predictor network update interval $K$; goal-conditioned policy update interval $M$; mini-batch size of samples for updating goal-conditioned policy $N$.
Initialize parameter $\theta$ for goal-conditioned exploration policy $\pi_g(s, g, a; \theta)$.
Initialize parameter $\omega_t$ for target network $f(\cdot; \omega_t)$, and $\omega_p$ for predictor network $\hat{f}(\cdot; \omega_p)$.
Initialize an empty experience replay buffer $\mathcal{B}$, and a priority queue $Q$ with randomly collected states.
for each iteration do
    Reset the environment and get the initial state $s_0$;
    Choose a goal $g$ from priority queue $Q$, and set $goal\_success = False$;
    for each timestep $t$ do
        # Interact with the environment
        if $goal\_success == True$ then
            Choose an random action $a_t$;
        else
            Choose an action $a_t$ according to $\pi_g(s_t, g, a_t; \theta)$;
        end if
        Send $a_t$ to the environment, get reward $r_t$ and the next state $s_{t+1}$, and update $goal\_success$;
        # Store new states and update the predictor network
        if $t \% K == 0$ then
            Store samples $\{s_k, g, a_k, r_k\}_{k=t-K}^{t}$ into replay buffer $\mathcal{B}$;
            Calculate prediction errors for $\{s_k\}_{k=t-K}^{t}$ and put these states into priority queue $Q$;
            Update predictor network $\hat{f}(\cdot; \omega_p)$ using $\{s_k\}_{k=t-K}^{t}$;
        end if
        # Update the goal-conditioned policy
        if $t \% M == 0$ then
            Update $\pi_g$ with $\{s_k, g_k, a_k, r'_k\}_{k=1}^{N}$ sampled from $\mathcal{B}$;
        end if
    end for
end for
```

4.1 Selecting goals from the experience buffer

As mentioned previously, our method is inspired by the principle of maximum state entropy exploration [18] to select novel states with the least visitation counts from the experience buffer as the targets. In this way, the goal-conditioned policy will spend more steps to visit the seldom visited states. As a result, the exploration policy prefers to uniformly visit all states and therefore leads to an increase in the state distribution entropy.

Unfortunately, computing visitation counts for tasks with high-dimensional visual inputs is intractable due to the curse of dimensionality. Therefore, we cannot directly apply the above goal-selection. However, it is still possible to build some variables that are related to visitation counts and are easy to compute. For example, [7] showed that prediction errors on a batch of data have a strong relationship with the number of training iterations. That is, if some samples are selected multiple times, the prediction errors of such samples will be small. Thus, we can use the prediction errors to reflect the visitation counts of observed states. Other approaches like pseudo-counts [4, 30] can be also applied, but we find that the mentioned method called random network distillation (RND) by [7] is easy to scale up.

Concretely, RND is consist of two randomly initialized neural networks: a fixed (non-trainable) network called target network $f(\cdot;\omega_t)$ that is parameterized by ω_t, and a trainable network called predictor network $\hat{f}(\cdot;\omega_p)$ that is parameterized by ω_p. Both two networks take a state s as input and output a vector with the same dimension. Each time a batch of data of size K feeds into the predictor network to minimize the difference between the predictor network and the target network with respect to the predictor network's parameters (see Equation 1).

$$\min_{\omega_p} \frac{1}{K} \sum_{i=1}^{K} ||f(s_i;\omega_t) - \hat{f}(s_i;\omega_p)||^2 \tag{1}$$

In practice, we employ an online learning setting to train RND and maintain a priority queue to store novel states based on the prediction errors of RND. In particular, after the goal-conditioned policy collects a mini-batch of states, these states will be used to train the predictor network. In this way, frequently visited states will have small prediction errors while the prediction errors for seldom visited states will be large. Also, states with large prediction errors will be stored into the priority queue and the state with the least prediction error will be removed out of the priority queue if full. This process repeats and for simplicity, no past data will be reused to train the predictor network. Based on this scheme, each iteration a novel state will be selected[4] from the priority queue as the desired goal for the goal-conditioned policy. After achieving the goal, the exploration policy will perform random actions to discover new states. Intuitively, such defined exploration behaviors will try to uniformly visit all states, which will even the state distribution and lead to an increase in state distribution entropy. This will be empirically verified in Section 5.

4.2 Training goal-conditioned policy efficiently

Ideally, each time we sample a novel state from the experience buffer and directly set it as the input for the goal-conditioned policy π_g. However, this processing is not friendly since the size of policy inputs is doubled and the representation of inputs may be redundant. Following prior studies about multi-goal reinforcement learning [1, 33], we manually extract useful information from the state space as the input of π_g. For instance, we extract the agent position information (i.e., coordinates) from raw states, which provides a good representation of the desired goal.

[4] We sample goals from a distribution (e.g., softmax distribution) based on their prediction errors rather than in a greedy way.

To assign rewards for goal-conditioned policy π_g, we need to judge whether it achieves the desired goal. Again, we use the same technique to extract a representation called achieved goal ag from the observed state [1, 33]. Let us denote the desired goal as g, we conclude the desired goal is accomplished if $d(ag_t, g)$ is small than a certain threshold ϵ, where ag_t is the achieved goal at timestep t and d is some distance measure (e.g., ℓ_2-norm). As a result, an ordinary method to compute rewards for the goal-conditioned policy is (note that the desired goal g does not change during an episode):

$$r'(ag_t, g) = \begin{cases} 1 & \text{if } d(ag_t, g) < \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

However, the training of goal-conditioned policy is slow with this sparse reward function. An alternative method is to use negative distance as the reward, i.e., $r' = -d(ag_t, g_t)$. However, the distance reward function may lead to unexpected behaviors [1, 27]. Next, we introduce some techniques to deal with the above problems.

$$r'(ag_t, g) = d(ag_{t-1}, g) - d(ag_t, g) \tag{3}$$

Firstly, let us consider the technique of reward shaping [27], which introduces additional training rewards to guide the agent. Clearly, this operation will modify the original objective and change the optimal policy if we don't pose any restrictions. Interestingly, reward shaping is invariant to the optimal policy if the reward shaping function is a potential function [27]. Specifically, we can define the difference of two consecutive distances (between the achieved goal and the desired goal) as a reward shaping function, shown in Equation 3. Since this function gives dense rewards, it leads to substantial acceleration in learning a near-optimal goal-conditioned policy. Consequently, the policy π_g can avoid meaningless actions within the explored region and quickly reach the exploration boundary. Verification of the optimal goal-conditioned policy is invariant under this reward shaping function is given in Appendix A.1.

Alternatively, one can use Hindsight Experience Replay (HER) [1] to train the goal-conditioned policy via replacing each episode with an achieved goal rather than one that the agent was trying to achieve. Concretely, for state s_t, we may randomly select a future state s_{t+k} from the same trajectory and replace its original target with s_{t+k}, where k is a positive index. In this way, the agent can still get positive rewards though it may not achieve the originally defined goal. But one should be careful when applying this technique since HER clearly changes the goal distribution for learning and may lead to undesired results for our setting.

4.3 Exploiting experience collected by exploration policy

In the above, we have discussed how to efficiently train a goal-conditioned policy to explore. To better leverage experience collected by the goal-conditioned policy, we need to additionally train an exploitation policy π_e that learns from goal-conditioned policy's experience buffer with environment rewards in an off-policy learning fashion.

Interestingly, it was shown that off-policy learning without interactions does not perform well on MuJoCo tasks [16]. The authors conjectured off-policy learning degenerates when the experience collected by the exploration policy is not correlated to the trajectories generated by the exploitation policy (they called this phenomenon *extrapolation error*). To remedy this issue, we add environment rewards on the goal rewards computed in the previous part. The environment rewards are scaled properly so that they do not change the original objective too much. In this way, π_e and π_g will not be too distinct. In addition, we parallelly train π_e as well as π_g and allow π_e to periodically interact with the environment to further mitigate the extrapolation error.

5 Experiment

In this section, we conduct experiments to answer the following research questions: 1) does novelty-pursuit lead to an increase in the state entropy compared with other baselines? 2) does the training technique for goal-conditioned policy improve the performance? 3) how does the performance of novelty-pursuit compare with the state-of-the-art approaches in complicated environments? We conduct experiments from simple maze environments, MuJoCo tasks, to long-horizon video games of SuperMarioBros to evaluate the proposed method. Detailed policy network architecture and hyperparameters are given in Appendix A.4 and A.5, respectively.

Here we briefly describe the environment settings (see Figure 2 for illustrations). Details are given in Appendix A.3.

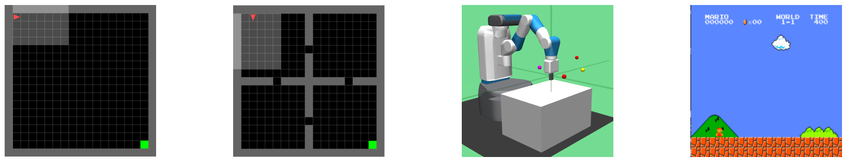
Empty Room & Four Rooms. An agent navigates in the maze of 17×17 to find the exit (the green square in Figure 2 (a) and (b)) [9]. The agent receives a time penalty until it finds the exit and receives a positive reward. The maximal episode return for both two environments is $+1$, and the minimal episode return is -1. Note that the observation is a partially observed image of shape $(7, 7, 3)$.

FetchReach. A 7-DOF Fetch Robotics arm (simulated in the MuJoCo simulator [42]) is asked to grip spheres above a table. There are a total of 4 spheres and the robot receives a positive reward of $+1$ when its gripper catches a sphere (the sphere will disappear after being caught) otherwise it receives a time penalty. The maximal episode return is $+4$, and the minimal episode return is -1.

SuperMarioBros. A Mario agent with raw image inputs explores to discover the flag. The reward is based on the score given by the NES simulator [19] and is clipped into -1 and $+1$ except $+50$ for a flag. There are 24 stages in the SuperMarioBros game, but we only focus on the stages of 1-1, 1-2, and 1-3.

5.1 Comparison of exploration efficiency

In this section, we study the exploration efficiency in terms of the state distribution entropy. We focus on the Empty Room environment because it is tractable to calculate the state distribution entropy for this environment.



(a) Empty Room (b) Four Rooms (c) FetchReach (d) SuperMarioBros

Fig. 2. Four environments considered in this paper.

Table 1. Average entropy of visited state distribution over 5 random seeds on the Empty Room environment. Here we use ± to denote the standard deviation.

	Entropy
random	5.129 ± 0.021
bonus	5.138 ± 0.085
novelty-pursuit	**5.285 ± 0.073**
novelty-pursuit-planning-oracle	5.513 ± 0.077
novelty-pursuit-counts-oracle	5.409 ± 0.059
novelty-pursuit-oracles	5.627 ± 0.001
maximum	5.666

We consider the following baselines: 1) random: uniformly selecting actions; 2) bonus: a curiosity-driven exploration method that uses the exploration bonus [7]; 3) novelty-pursuit: the proposed method. We also consider three variants of our method: 4) novelty-pursuit-planning-oracle: the proposed method with a perfect goal-conditioned planning policy; 5) novelty-pursuit-counts-oracle: the proposed method with goal-selection based on true visitation counts; 6) novelty-pursuit-oracles: the proposed method with the above two oracles. The results are summarized in Table 1. Here we measure the entropy over all visited states by the learned policy. Note that the maximum state distribution entropy for this environment is 5.666.

Firstly, we can see that novelty-pursuit achieves a higher entropy than the random and bonus method. Though the bonus method outperforms the random method, it is inefficient to a maximum state entropy exploration. We attribute this to delayed and imperfect feedbacks of the exploration bonus. Secondly, when the planning oracle and visitation counts oracle are available, the entropy of our method roughly improves by 0.228 and 0.124, respectively. We notice that the planning-oracle variant avoids random actions within the exploration boundary and spends more meaningful steps to explore around the exploration boundary, thus greatly improves the entropy. Based on this observation, we think accelerating goal-conditioned policy training is more important for our method. Thirdly, the combination of two oracles gives a near-perfect performance (the gap between the maximum state entropy is only 0.039). This result demonstrates that goal-condition exploration behaviors presented by novelty-pursuit can increase the state entropy.

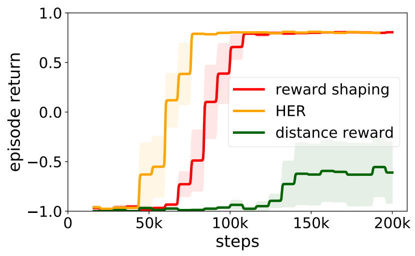


Fig. 3. Training curves of learned policies with different goal-conditioned policy training techniques on the Empty Room environment. Solid lines correspond to the mean of episode returns while shadow regions indicate the standard deviation.

5.2 Ablation study of training techniques

In this section, we study the core component of our method regarding quickly learning a goal-conditioned policy.

In particular, we study the effects of qualities of goal-conditioned policies when using different training techniques. We compare HER and the reward-shaping with the distance reward function. Results on the Empty Room are shown in Figure 3.

From Figure 3, we find that both HER and reward shaping can accelerate training goal-conditioned policies. Similar to the planning-oracle in the previous section, such-trained goal-conditioned policy avoids random exploration within the explored region, hence it can quickly find the exit. The distance reward function may change the optimal behaviors of goal-conditioned policy and therefore does not perform well.

5.3 Evaluation on complicated environments

In this section, we compare different methods in terms of environment rewards. We will see that without sufficient and efficient exploration, the learned policy may be stuck into the local optimum. Two baseline methods are considered: 1) vanilla: DDPG [23] with Gaussian action noise on Fetch Reach with the continuous action space and ACER [43] with policy entropy regularization on other environments with the discrete action space; 2) bonus: a modified vanilla method that combines the environments reward and the exploration bonus [7]. Note reported results of novelty-pursuit are based on the performance of the exploitation policy π_e rather than the goal-conditioned exploration policy π_g. And we keep the same number of samples and policy optimization iterations for all methods to ensure fairness.

Firstly, we consider the Empty Room and the Four Rooms environments. The results are shown in the first two parts of Figure 4. We see that the vanilla method hardly finds the exit on the maze environments. Novelty-pursuit outperforms the bonus method on both environments. And we also observe that the behaviors

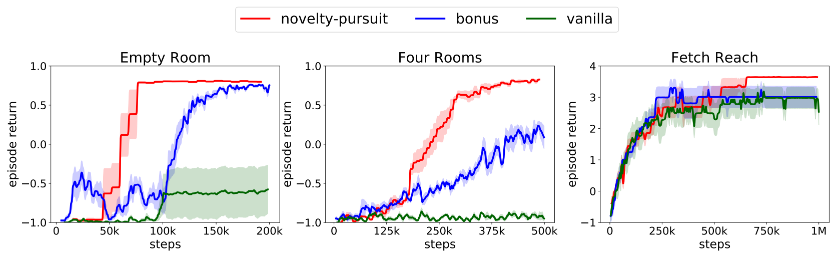


Fig. 4. Training curves of learned policies over 5 random seeds on the Empty Room, Four Rooms, and FetchReach environments. Solid lines correspond to the mean of episode returns while shadow regions indicate the standard deviation.

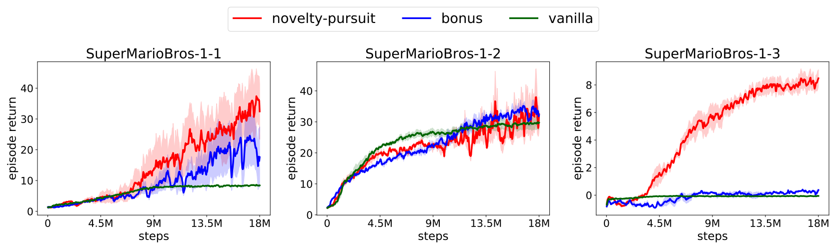


Fig. 5. Training curves of learned policies over 3 random seeds on the game of Super-MarioBros. Solid lines correspond to the mean of episode returns while shadow regions indicate the standard deviation.

of the bonus method are somewhat misled by the imperfect exploration bonus though we have tried many weights to balance the environment reward and exploration bonus.

Secondly, we consider the FetchReach environment, and results are shown in the most right part of Figure 4. We see that novelty-pursuit can consistently grip 4 spheres while other methods sometimes fail to efficiently explore the whole state space to grip 4 spheres.

Finally, we consider the SuperMarioBros environments, in which it is very hard to discover the flag due to the huge state space and the long horizon. Learning curves are plotted in Figure 5 and the final performance is listed in Table 2. We find the vanilla method gets stuck into the local optimum on SuperMarioBros-1-1 while the bonus method and ours can find a near-optimal policy. All methods perform well on SuperMarioBros-1-2 because of the dense rewards of this task. On SuperMarioBros-1-3, this task is very challenging because rewards are very sparse and all methods fail to discover the flag on this environment. To better understand the learned policies, we plot trajectories of different methods on SuperMarioBros-1-3 in Figure 6, and more results on other environments can be found in Appendix A.2. It turns out only our method can

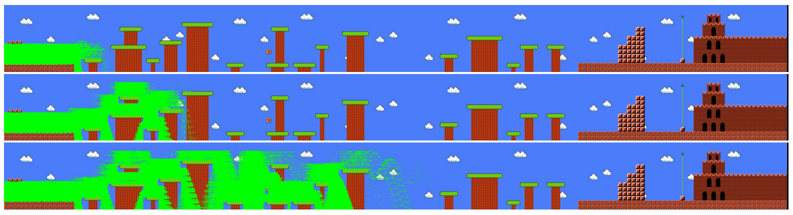


Fig. 6. Trajectory visualization on SuperMarioBros-1-3. Trajectories are plotted in green cycles with the same training samples (18 million). The agent starts from the most left part and needs to fetch the flag on the most right part. Top row: vanilla ACER; middle row: ACER + exploration bonus; bottom row: novelty-pursuit (ours).

get positive rewards and make certain progress via deep exploration behaviors presented by the goal-conditioned policy on SuperMarioBros-1-3.

Table 2. Final Performance of learned policies over 3 random seeds on SuperMarioBros. We use ± to denote the standard deviation.

	novelty-pursuit	bonus	vanilla
SuperMarioBros-1-1	**36.02 ± 8.19**	17.74 ± 7.84	8.43 ± 0.14
SuperMarioBros-1-2	**33.30 ± 6.13**	**33.19 ± 1.53**	29.64 ± 2.02
SuperMarioBros-1-3	**8.14 ± 0.55**	0.20 ± 0.14	-0.07 ± 0.01

6 Conclusion

We focus on the efficient exploration aspect of RL in this paper. We propose a goal-section criterion based on the principle of maximum state entropy exploration and demonstrate the proposed method is efficient towards exploring the whole state space. Therefore, the proposed method could escape from the local optimum and heads the (near-) optimal policy. Goal representation is somewhat manually extracted in our method and we believe an automatic representation learning for goal-conditioned learning can lead to more general applications. In addition to the way of off-policy learning, methods based on imitation learning to leverage good experience is another promising direction.

Acknowledgements

The authors will thank Fabio Pardo for sharing ideas to visualize trajectories for SuperMarioBros. In addition, the authors appreciate the helpful instruction from Dr. Yang Yu and the insightful discussion with Tian Xu as well as Xianghan Kong.

References

[1] Andrychowicz, M., Crow, D., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., Zaremba, W.: Hindsight experience replay. In: Proceedings of the 30th Annual Conference on Neural Information Processing Systems. pp. 5048–5058 (2017)

[2] Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multi-armed bandit problem. Machine Learning **47**(2-3), 235–256 (2002)

[3] Baranes, A., Oudeyer, P.: R-IAC: robust intrinsically motivated exploration and active learning. IEEE Transactions on Autonomous Mental Development **1**(3), 155–169 (2009)

[4] Bellemare, M.G., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., Munos, R.: Unifying count-based exploration and intrinsic motivation. In: Proceedings of the 29th Annual Conference on Neural Information Processing Systems. pp. 1471–1479 (2016)

[5] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: OpenAI gym. CoRR **1606.01540** (2016)

[6] Burda, Y., Edwards, H., Pathak, D., Storkey, A.J., Darrell, T., Efros, A.A.: Large-scale study of curiosity-driven learning. In: Proceedings of the 7th International Conference on Learning Representations (2019)

[7] Burda, Y., Edwards, H., Storkey, A.J., Klimov, O.: Exploration by random network distillation. In: Proceedings of the 7th International Conference on Learning Representations (2019)

[8] Chen, X., Yu, Y.: Reinforcement learning with derivative-free exploration. In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems. pp. 1880–1882 (2019)

[9] Chevalier-Boisvert, M., Willems, L., Pal, S.: Minimalistic gridworld environment for openai gym. `https://github.com/maximecb/gym-minigrid` (2018)

[10] Choromanski, K., Pacchiano, A., Parker-Holder, J., Tang, Y., Sindhwani, V.: From complexity to simplicity: Adaptive es-active subspaces for blackbox optimization. In: Proceedings of the 32nd Annual Conference on Neural Information Processing Systems. pp. 10299–10309 (2019)

[11] Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y., Zhokhov, P.: OpenAI baselines. `https://github.com/openai/baselines` (2017)

[12] Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K.O., Clune, J.: Go-explore: a new approach for hard-exploration problems. CoRR **1901.10995** (2019)

[13] Florensa, C., Held, D., Geng, X., Abbeel, P.: Automatic goal generation for reinforcement learning agents. In: Proceedings of the 35th International Conference on Machine Learning. pp. 1514–1523 (2018)

[14] Forestier, S., Mollard, Y., Oudeyer, P.: Intrinsically motivated goal exploration processes with automatic curriculum learning. CoRR **1708.02190** (2017)

[15] Fortunato, M., Azar, M.G., Piot, B., Menick, J., Hessel, M., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C., Legg, S.: Noisy networks for exploration. In: Proceedings of the 6th International Conference on Learning Representations (2018)

[16] Fujimoto, S., Meger, D., Precup, D.: Off-policy deep reinforcement learning without exploration. In: Proceedings of the 36th International Conference on Machine Learning. pp. 2052–2062 (2019)

[17] Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: Proceedings of the 35th International Conference on Machine Learning. pp. 1856–1865 (2018)

[18] Hazan, E., Kakade, S.M., Singh, K., Soest, A.V.: Provably efficient maximum entropy exploration. In: Proceedings of the 36th International Conference on Machine Learning. pp. 2681–2691 (2019)

[19] Kauten, C.: Super Mario Bros for OpenAI Gym. `https://github.com/Kautenja/gym-super-mario-bros` (2018)

[20] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Proceedings of the 3rd International Conference on Learning Representations (2015)

[21] Kolter, J.Z., Ng, A.Y.: Near-bayesian exploration in polynomial time. In: Proceedings of the 26th International Conference on Machine Learning. pp. 513–520 (2009)

[22] Lattimore, T., Hutter, M.: Near-optimal PAC bounds for discounted mdps. Theoretical Computer Science **558**, 125–143 (2014)

[23] Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. In: Proceedings of the 4th International Conference on Learning Representations (2016)

[24] Liu, F., Li, Z., Qian, C.: Self-guided evolution strategies with historical estimated gradients. In: Proceedings of the 29th International Joint Conference on Artificial Intelligence. pp. 1474–1480 (2020)

[25] Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T.P., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: Proceedings of the 33rd International Conference on Machine Learning. pp. 1928–1937 (2016)

[26] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M.A., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)

[27] Ng, A.Y., Harada, D., Russell, S.J.: Policy invariance under reward transformations: Theory and application to reward shaping. In: Proceedings of the 16th International Conference on Machine Learning (1999)

[28] O'Donoghue, B., Munos, R., Kavukcuoglu, K., Mnih, V.: PGQ: combining policy gradient and q-learning. CoRR **1611.01626** (2016)

[29] Osband, I., Blundell, C., Pritzel, A., Roy, B.V.: Deep exploration via bootstrapped DQN. In: Proceedings of the 29th Annual Conference on Neural Information Processing Systems. pp. 4026–4034 (2016)

[30] Ostrovski, G., Bellemare, M.G., van den Oord, A., Munos, R.: Count-based exploration with neural density models. In: Proceedings of the 34th International Conference on Machine Learning. pp. 2721–2730 (2017)

[31] Pathak, D., Agrawal, P., Efros, A.A., Darrell, T.: Curiosity-driven exploration by self-supervised prediction. In: Proceedings of the 34th International Conference on Machine Learning. pp. 2778–2787 (2017)

[32] Péré, A., Forestier, S., Sigaud, O., Oudeyer, P.: Unsupervised learning of goal spaces for intrinsically motivated goal exploration. In: Proceedings of the 6th International Conference on Learning Representations (2018)

[33] Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., Kumar, V., Zaremba, W.: Multi-goal reinforcement learning: Challenging robotics environments and request for research. CoRR **1802.09464** (2018)

[34] Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R.Y., Chen, X., Asfour, T., Abbeel, P., Andrychowicz, M.: Parameter space noise for exploration. In: Proceedings of the 6th International Conference on Learning Representations (2018)

[35] Schaul, T., Horgan, D., Gregor, K., Silver, D.: Universal value function approximators. In: Proceedings of the 32nd International Conference on Machine Learning. pp. 1312–1320 (2015)

[36] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. CoRR **1707.06347** (2017)

[37] Stadie, B.C., Levine, S., Abbeel, P.: Incentivizing exploration in reinforcement learning with deep predictive models. CoRR **1507.00814** (2015)

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

[39] Sutton, R.S., Barto, A.G.: Introduction to reinforcement learning. MIT Press (1998)

[40] Szepesvári, C.: Algorithms for Reinforcement Learning. Morgan & Claypool Publishers (2010)

[41] Tieleman, T., Hinton, G.: Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude (2012)

[42] Todorov, E., Erez, T., Tassa, Y.: MuJoCo: A physics engine for model-based control. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 5026–5033 (2012)

[43] Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., de Freitas, N.: Sample efficient actor-critic with experience replay. In: Proceedings of the 5th International Conference on Learning Representations (2017)

[44] Yu, Y.: Towards sample efficient reinforcement learning. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence. pp. 5739–5743 (2018)

A Appendix

A.1 Reward shaping for training goal-conditioned policy

Reward shaping is invariant to the optimal policy under some conditions [27]. Here we verify that the reward shaping function introduced by our method doesn't change the optimal behaviors for goal-conditioned policy. Lets' consider the total shaping rewards during an episode of length T:

$$\begin{aligned}
&\sum_{t=1}^{T} -d(ag_t, g) + d(ag_{t+1}, g) \\
&= -d(ag_1, g) + d(ag_2, g) - d(ag_2, g) + d(ag_3, g) \cdots \\
&= -d(ag_1, g) + d(ag_{T+1}, g)
\end{aligned}$$

For the optimal policy π_g^*, $d(ag_{T+1}, g) = 0$ while $d(ag_1, g)$ is a constant. Therefore, the optimal policy π_g induced by the reward shaping is invariant to the one induced by the sparse reward function in Equation 2.

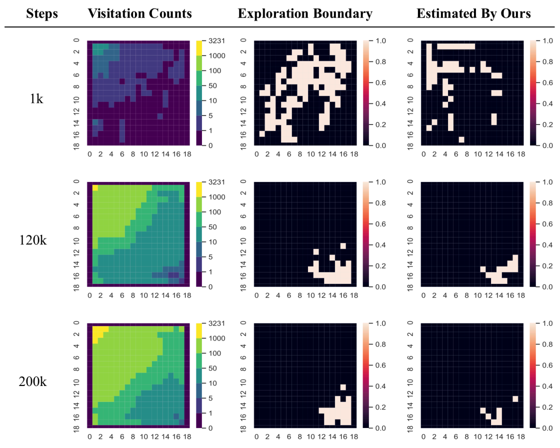


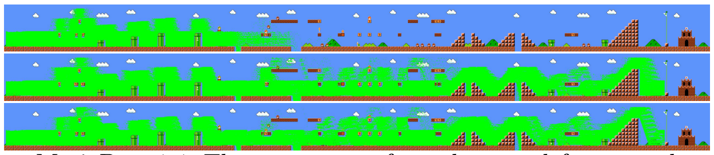
Fig. 7. Visualization for the true visitation counts and the corresponding exploration boundary.

A.2 Additional results

In this part, we provide additional experiment results to better understand our method.

Empty Room. We visualize the true visitation counts and the corresponding exploration boundary in Figure 7. Note the agent starts from the left top corner and the exit is on the most bottom right corner. The data used for visualization is collect by a random policy. Hence, the visitation counts are large on the left top part. We define the true exploration boundary as the top 10% states with least visitation counts and the estimated exploration boundary given by our method are states with the largest prediction errors in the priority queue. From this figure, we can see that our method can make a good approximation to the true exploration boundary given by visitation counts.

SuperMarioBros. In Figure 8, we make additional trajectory visualization on SuperMarioBros-1-1 and SuperMarioBros-1-2. Trajectories are plotted with the same number of samples (18M). We can observe that the vanilla method gets into the local optimum on SuperMarioBros-1-1 even though it has used the policy entropy regularization to encourage exploration. In addition, only our method can get the flag on SuperMarioBros-1-2.


(a) SuperMarioBros-1-1. The agent starts from the most left part and needs to find the flag on the most right part.

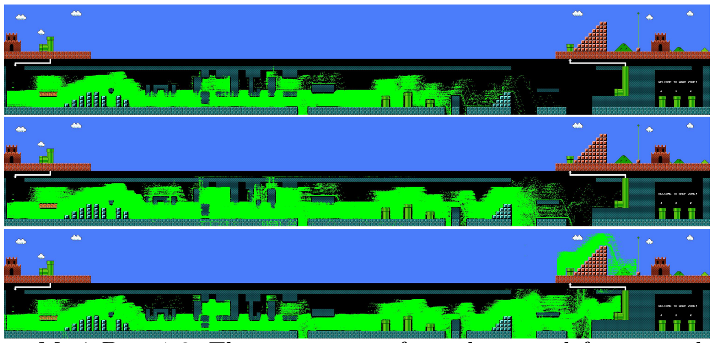

(b) SuperMarioBros-1-2. The agent starts from the most left part and needs to get the flag through the water pipe on the right part (see arrows).

Fig. 8. Trajectory visualization on SuperMarioBros-1-1 and SuperMarioBros-1-2. For each figure, top row: vanilla ACER; middle row: ACER + exploration bonus; bottom row: novelty-pursuit (ours). The vanilla method gets stuck into the local optimum on SuperMarioBros-1-1. Only our method can get the flag on SuperMarioBros-1-2.

A.3 Environment prepossessing

In this part, we present the used environment preprocessing.

Maze. Different from [9], we only use the image and coordination information as inputs. Also, we only consider four actions: turn left, turn right, move forward and move backward. The maximal episode length is 190 for Empty Room, and 500 for Four Rooms. Each time the agent receives a time penalty of 1/max_episode_length and receives a reward of +1 when it finds the exit.

FetchReach. We implement this environment based on *FetchReach-v0* in Gym [5]. The maximal episode length is 50. The xyz coordinates of four spheres are $(1.20, 0.90, 0.65)$, $(1.10, 0.72, 0.45)$, $(1.20, 0.50, 0.60)$, and $(1.45, 0.50, 0.55)$. When sampling goals, we resample goals if the target position is outside of the table i.e., the valid x range: $(1.0, 1.5)$, the valid y range is $(0.45, 1.05)$, and the valid z range is $(0.45, 0.65)$.

SuperMarioBros. We implement this environment based on [19] with OpenAI Gym wrappers. Prepossessing includes grey-scaling, observation downsampling, external reward clipping (except that 50 for getting flag), stacked frames of 4, and sticky actions with a probability of 0.25 [26]. The maximal episode length is 800. The environment restarts to the origin when the agent dies.

A.4 Network architecture

We use the convolutional neural network (CNN) for Empty Room, Four Rooms, and video games of SuperMarioBros, and multi-layer perceptron (MLP) for FetchReach environment. Network architecture design and parameters are based on the default implementation in OpenAI baselines [11]. For each environment, RND uses a similar network architecture. However, the predictor network has additional MLP layers than the predictor network to strengthen its representation power [7].

A.5 Hyperparameters

Table 3 gives hyperparameters for ACER [43] on the maze and SuperMarioBros (the learning algorithm is RMSProp [41]). DDPG [23] used in Fetch Reach environments is based on the HER algorithm implemented in OpenAI baselines [11] expect that the actor learning rate is 0.0005. We run 4 parallel environments for DDPG and the size of the priority queue is also 100. As for the predictor network, the learning rate of the predictor network is 0.0005 and the optimization algorithm is Adam [20] for all experiments, and the batch size of training data is equal to the product of rollout length and the number of parallel environments.

The goal-conditioned exploration policy of our method is trained by combing the shaping rewards defined in Equation 3 and environment rewards, which helps reduce the discrepancy with the exploitation policy. The weight for environment rewards is 1 for all environments except 2 for SuperMarioBros. For the bonus method used in Section 5, the weight β to balance the exploration bonus is 0.1 for Empty Room and Four Rooms, 0.01 for FetchReach, 1.0 for

SuperMarioBros-1-1 and SuperMarioBros-1-3, and 0.1 for SuperMarioBros-1-2. Following [6, 7] we also do a normalization for the exploration bonus by dividing them via a running estimate of the standard deviation of the sum of discounted exploration bonus. In addition, we find sometimes applying the imitation learning technique for the goal-conditioned policy can improve performance. We will examine this in detail in future works. Though we empirically find HER is useful in simple environments like the maze and the MuJoCo robotics tasks, we find it is less powerful than the technique of reward shaping on complicated tasks like SuperMarioBros. Hence, reported episode returns of learned policies are based on the technique of reward shaping on SuperMarioBros and HER for others.

For the exploitation policy, we periodically allow it to interact with the environment to mitigate the exploration error [16]. For all experiments, we split the half interactions for the exploitation method. For example, if the number of maximal samples is $200k$, the exploration and the exploitation policy will use the same $100k$ interactions.

Table 3. Hyperparameters of our method based on ACER on the maze and SuperMarioBros environments.

Hyperparameters	Empty Room	Four Rooms	SuperMarioBros
Rollout length	20	20	20
Number of parallel environments	4	4	8
Learning rate	0.0007	0.0007	0.00025
Learning rate schedule	linear	linear	constant
Discount factor γ	0.95	0.95	0.95
Entropy coefficient	0.10	0.10	0.10
Size of priority queue	100	100	20
Total training steps	200K	500K	18M