DEEP EXPLORATION BY NOVELTY-PURSUIT WITH MAXIMUM STATE ENTROPY

Anonymous authors
Paper under double-blind review

ABSTRACT

Efficient exploration is essential to reinforcement learning in huge state space. Recent approaches to address this issue include the intrinsically motivated goal exploration process (IMGEP) and the maximum state entropy exploration (MSEE). In this paper, we disclose that goal-conditioned exploration behaviors in IMGEP can also maximize the state entropy, which bridges the IMGEP and the MSEE. From this connection, we propose a maximum entropy criterion for goal selection in goal-conditioned exploration, which results in the new exploration method *novelty-pursuit*. Novelty-pursuit performs the exploration in two stages: first, it selects a 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. 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.

1 INTRODUCTION

Efficient exploration is important to learn a (near-) optimal policy for reinforcement learning (RL) in huge state space (Sutton & Barto, 1998). Dithering strategies like $\epsilon-$greedy, Gaussian action noise, and Boltzmann exploration are inefficient and require exponential interactions to explore the whole state space. In contrast, deep exploration (Osband et al., 2016) overcomes this dilemma via temporally extended behaviors with a long-term vision. Recently, proposed methods include the intrinsically motivated goal exploration process (IMGEP) (Forestier et al., 2017), and maximum state entropy exploration (MSEE) (Hazan et al., 2019). In particular, IMGEP selects interesting states from the experience buffer as goals for a goal-conditioned exploration policy. In this way, exploration behaviors are naturally temporally-extended via accomplishing self-generated goals. On the other hand, maximum state entropy aims to search a policy such that it maximizes the entropy of state distribution.

In this paper, we show that goal-conditioned exploration behaviours can also maximize the entropy of state distribution. Informally, we show that the target of maximizing the support of empirical state distribution and the entropy of empirical state distribution can be both achieved by goal-conditioned policy with a specific design. In particular, the goal-conditioned policy performs in two stages: first, it selects a 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. The exploration policy leads to maximize the state entropy on the whole state distribution considering tabular MDP. Thus, the IMGEP and the MSEE is connected by goal-conditioned behaviors. From this connection, we proposed a practical method called *novelty-pursuit*. An illustration can be seen in Figure 1. Intuitively, this process is efficient since the agent avoids exploring within the explored region. Besides, the exploration boundary will be expanded further as more and more new states are discovered. Finally, the agent will probably explore the whole state space.

In practice, it is unknown where the exploration boundary is. To deal with this problem, we assume the state distribution density over the whole state space is continuously expanded, thus the visited states and non-visited states are geometrically close. So the exploration boundary can be approximated using states with the least density. In high-dimension space, we estimate the exploration boundary based on prediction errors given by Random Network Distillation (Burda et al., 2019b).

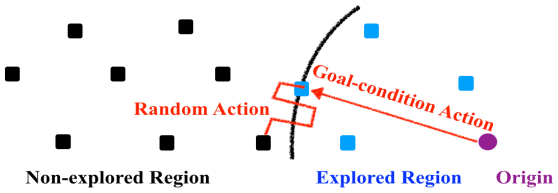


Figure 1: Illustration for the proposed method. A goal-conditioned policy firstly reaches the exploration boundary, then perform random actions to discover new states.

Ideally, a greater prediction error indicates fewer state distribution density. Besides, we observe that previous methods based on distance-like reward are inefficient to train the goal-conditioned exploration policy. We employ training techniques based on reward shaping (Ng et al., 1999) and HER (Andrychowicz et al., 2017) to accelerate training the goal-conditioned policy.

Our contributions are summarized as follows: (1) We disclose that goal-conditioned behaviors can also maximize the state entropy, which bridges the intrinsically motivated goal exploration process and the maximum state entropy explore. (2) We propose a method called novelty-pursuit from this connection and give practical implementations. (3) We demonstrate the exploration efficiency of the proposed method and achieve better performance on environments from the maze, Mujoco tasks, to long-horizon video games of SuperMarioBros.

2 Background

Reinforcement Learning. In the standard reinforcement learning framework (Sutton & Barto, 1998) a learning agent interacts with a Markov Decision Process (MDP). The sequential decision process is characterized as follows: at each time t, the agent receives a state s_t from the environment and selects an action a_t from its policy $\pi(s, a) = \Pr\{a_t = a | s_t = s\}$; that decision is sent back to the environment, and the environment gives a reward signal $r_t(s, a)$ and transits to the next state based on the (unknown) state transition probabilities $p^a_{ss'} = \Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$. This process repeats until the agent encounters a terminal state after which the process restarts. Each (stationary) policy $\pi(s, a)$ induces a state distribution $d_\pi(s) = (1 - \gamma)\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t \Pr\{s_t = s\}]$. The target of reinforcement learning is to maximize the expected discounted return $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t]$ in an unknown environment (e.g. $p^a_{ss'}$ is unknown), where $\gamma \in (0, 1]$ is a factor that balances the importance of future reward. Thus, the agent needs exploration to discover potential valuable states. Without sufficient exploration, the policy may be stuck into the local optimum.

Intrinsically Motivated Goal Exploration Process. Intrinsically motivated goal exploration process (IMGEP) (Baranes & Oudeyer, 2009; Forestier et al., 2017) relies on a goal-conditioned (or goal-parameterized) policy π_g for unsupervised exploration. It involves the following steps: 1) selecting an intrinsic or interesting state from the experience buffer as the desired goal; 2) exploring with a goal-conditioned policy $\pi_g(s, a, g) = \Pr\{a_t = a | s_t = s, g_t = g\}$; 3) reusing experience for an exploitaion policy $\pi_e(s, a) = \Pr\{a_t = a | s_t = s\}$ to maximize the external reward. Note that the first two steps don't need external reward.

Maximum State Entropy Exploration. Maximum state entropy exploration (Hazan et al., 2019) aims to search an exploration policy π^* such that it maximizes the entropy of induced state distribution (or minimizes the KL-divergence between the uniform distribution and induced state distribution) among the class of stationary policies (i.e., $\pi^* \in \arg\max_\Pi H[d_\pi]$). Without any information about tasks given by the environment, the principle of maximum state entropy exploration is safe for exploitation.

3 IMGEP with Maximum State Entropy Exploration

In this section, we bridge the intrinsically motivated goal exploration process and maximum state entropy exploration. We begin with practical considerations when maximizing state entropy, then

analyze the exploration characteristics of the proposed metric of visitation counts for IMGEP. We also discuss practical approaches based on the above analysis.

In practice, an exact density estimator for high-dimension state space is intractable, and the state space is unknown, which leads to an empirical state distribution over visited states. The differences are important. For example, directly optimizing the entropy of empirical state distribution over visited states is not what we want, because it ignores the non-visited states outside of the empirical state distribution (see the top row in Fig 2). Instead, we need to first maximize the support of induced state distribution (i.e., discovering new states), then we maximize the entropy of induced state distribution with full support (see the bottom row in Fig 2). In the following, we demonstrate that selecting the states with the least visitation counts among visited states as goals can achieve the above functions under some assumptions.

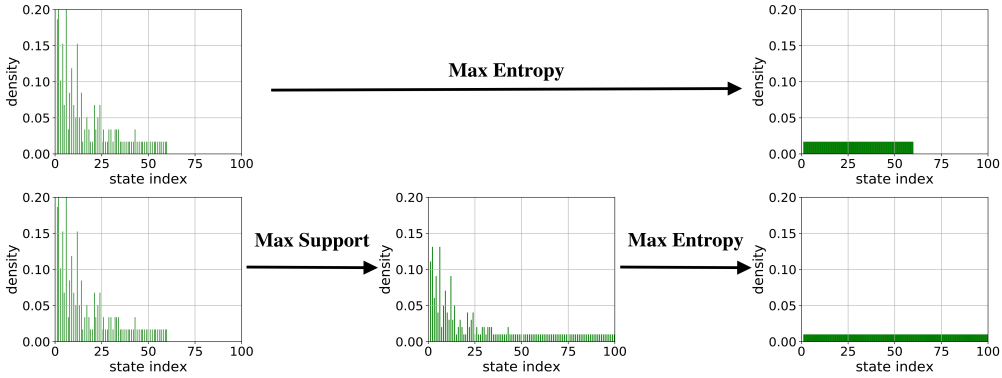


Figure 2: Histograms for normalized state visitation counts, where the x-axis represents the index of state. Top row: directly maximizing the entropy of empirical state distribution over visited states; Bottom row: firstly maximizing the counting measure of induced state distribution support, then maximizing the entropy of state distribution with full support.

Let the index set $\{1, 2, \cdots, |S|\}$ denotes the state space S, $\pi_{1:t}$ denotes the set of policies $\{\pi_1, \pi_2, \cdots, \pi_t\}$ over previous iterations, π_{t+1} denotes the policy of next iteration, $x_t(i)$ denotes the cumulative visitation counts of i-th state induced by history policies $\pi_{1:t}$, and $Z_t = \sum_{i=1}^{|S|} x_t(i)$ denotes the sum of all state visitation counts. Hence, the entropy of empirical state distribution induced by policies $\pi_{1:t}$ is defined as $H[d_{\pi_{1:t}}(s)] = \sum_{i=1}^{|S|} \frac{x_t(i)}{Z_t} \log \frac{x_t(i)}{Z_t}$ (H_t for short), and the counting measure of empirical state distribution support induced by policies $\pi_{1:t}$ is defined as $\mu[d_{\pi_{1:t}}(s)] = \sum_{i=1}^{|S|} \mathbf{1}_{x_t(i) \geq 1}$ (μ_t for short).

The theoretical analysis starts with the situation that each iteration the goal-conditioned exploration policy can only select a state to visit. Our question is which state to visit gives the most benefits in terms of maximum state entropy. This question is closely related to the goal generation in IMGEP. To facilitate the analysis, let the unit vector $\boldsymbol{e} = [0, \cdots, 1, \dots] \in \mathbb{R}^{|S|}$ denotes a choice (i.e., $\boldsymbol{e}(i) = 1$ indicates that the policy selects i-th state to visit). Note that $x_{t+1} = x_t + \boldsymbol{e}_t$ with this assumption.

Theorem 1 (Max Counting Measure of Support) *$\forall x_t(i) \geq 0, i \in \{1, \cdots, |S|\}$, $K = \{i | x_t(i) = 0\}$; unless $K = \emptyset$, for any unit vector $\boldsymbol{e}_t$ such that $\boldsymbol{e}_t(i) = 1$ with $i \in K$, we have $\mu_{t+1} = \mu_t + 1$.*

The proof is obvious and we omit here. Theorem 1 indicates visiting the non-visited states is to maximize the counting measure of induced state distribution support. The agent can obtain potential reward signal by discovering new states. However, we don't know what non-visited states are and where non-visited states locate in practice since we can't access to the whole state space in advance. In other words, we can't select these non-visited states as goals since they are not contained in the experience buffer. To deal with this problem, we assume that state density over the whole state space are continuous, thus visited states and non-visited states are close. Intuitively, there is an exploration boundary separating them. We can approximate the boundary using states with least visitation counts among all visited states. In conclusion, the goal-conditioned exploration policy

is asked to reach the exploration boundary after which it performs random actions to discover new states.

Theorem 2 (Max Entropy) $\forall x_t(i) \geq 1, i \in \{1, \cdots, |S|\}$*; for any unit vector* $\boldsymbol{e}_t^*$ *such that* $\boldsymbol{e}_t^*(i) = 1$ *with* $i \in \arg\min_j x_t(j)$*, we have* $\boldsymbol{e}_t^* \in \arg\max_{\boldsymbol{e}_t} H_{t+1}$.

We provide the proof in the appendix A.1. Theorem 2 characterizes the behavior of visiting the states with the least visitations when the whole state space has been explored (i.e., the stage after maximizing the counting measure of induced state distribution support). The same mechanism derived from Theorem 1 can also be applied to maximize the entropy of induced state distribution by ignoring the influence of random action sequences. To summarize, despite whether the whole state space has been traversed, the goal-condition exploration is asked to reach the exploration boundary and perform random action sequences.

It is easy to unify above theoretical analysis via a smoothed entropy $H_\sigma(d_\pi) = -\mathbb{E}_{d_\pi}[\log(d_\pi) + \sigma]$ (Hazan et al., 2019). For our problem, the entropy is proper by assigning the non-visited states with a dummy visitation counts between 0 and 1. In that case, Theorem 2 still holds and suggests firstly selecting these non-visited states and subsequently selecting the states with least visitation counts to maximize the smoothed state entropy.

We call the proposed method **novelty-pursuit**. For a complete reinforcement learning setting, the above analysis neglects the influence of trajectories to the exploration boundary and uncontrollable factors due to random actions after reaching the exploration boundary. But we think the practical approximation of planning oracle (i.e, a perfect goal-conditioned policy) and an exact exploration boundary with visitations counts are more important. In the following, we describe how to deal with these problems.

4 METHOD

In this section, we present practical implementations for the proposed method. How to approximate visitation counts in high-dimension space and how to estimate the state boundary is given in Section 4.1. We describe the training technique of goal-conditioned policy in Section 4.2. Finally, we introduce an exploitation policy to learn the experience collected by the goal-conditioned exploration policy in Section 4.3. We outline the novelty-pursuit exploration in Algorithm 1.

4.1 APPROXIMATING STATE BOUNDARY IN HIGH-DIMENSION SPACE

Generally, computing the visitation counts in high-dimension space is intractable. However, it is possible to build some variables related to the visitation counts. For example, Random Network Distillation (RND) (Burda et al., 2019b) shows that prediction errors given by two randomly initialized network have a strong relationship to the number of training samples. Thus, we can use the prediction errors to sort visited states. Other approaches like pseudo-counts (Bellemare et al., 2016; Ostrovski et al., 2017) can be also applied, but we find that RND is easy to scale up.

RND is consist of two randomly initialized neural networks: a fixed network called target network $f(x; \omega_t)$, and a trainable network called predictor network $\hat{f}(x; \omega_p)$. Both two networks take a state s as input and output a vector with the same dimension. Each time a batch of data feed into the predictor network to minimize the difference between the predictor network and the target network concerning the predictor network's parameters, shown in 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 states with the highest prediction errors. In particular, after a goal-conditioned policy collects a mini-batch of transitions, this data feed to train the predictor network. Also, a state with high prediction error 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 no historical data will be reused to train the predictor network. Besides, each iteration a state will be selected from the priority

Algorithm 1 Exploration by Novelty-Pursuit

Input: predictor network update interval K; goal-conditioned policy update interval M; mini-batch size of samples for goal-condtioned policy N;
Initialize parameter θ for goal-condtioned exploration policy $\pi_g(s, g, a; \theta)$.
Initialize parameter ω_t for target network $f(x; \omega_t)$, and ω_p for predictor network $\hat{f}(x; \omega_p)$.
Initialize a replay buffer D_g for π_g, and a priority queue Q to store novely state.
for each iteration **do**
  Reset the environment and get the observation o_0;
  Choose a goal g from priority queue Q, and set $goal\_success = False$;
  for each timestep t **do**
    if $goal\_success == True$ **then**
      Choose an random action a_t; **# Explore on the state boundary**
    else
      Choose an action a_t from $\pi_g(s_t, g, a_t; \theta)$; **# Go to the state boundary**
    end if
    Send a_t to the environment and get r^e_t, s_{t+1};
    Update $goal\_success(s_{t+1}, g)$;
    # Store new states and update the predictor network
    if $t \,\%\, K == 0$ **then**
      Store transitions $\{s_k, g, a_k, r^e_k\}^t_{k=t-K}$ into replay buffer D_g;
      Calculate prediction errors for $\{s_k\}^t_{k=t-K}$ and store them into priority queue Q;
      Update predictor network $\hat{f}(x; \omega_p)$ using $\{s_k\}^t_{k=t-K}$;
    end if
    # Update π_g with reward shaping
    if $t \,\%\, M == 0$ **then**
      Update π_g with $\{s_k, g_k, a_k, r^i_k\}^K_{k=1}$ sampled from D_g;
    end if
  end for
end for

queue as a goal for the goal-conditioned policy. After achieving the goal, the exploration policy will perform random actions to discover new states. Consider the bias due to approximation, we sample goals from a distribution based on their prediction errors (e.g., softmax distribution).

4.2 Training techniques for goal-conditioned policy

Before we describe the training techniques for the goal-conditioned policy, we emphasize that training this policy doesn't require the external reward signal from the environment. But we additionally use the external reward for the goal-conditioned policy (i.e., $r' = r_{ext} + \alpha r_g$) to reduce the mismatch behaviors between the goal-conditioned policy and the exploitation policy.

Following multi-goal reinforcement learning (Andrychowicz et al., 2017; Plappert et al., 2018a), we manually extract goal information from state space. Specifically, each state s is associated with an achieved goal of ag, and the desired goal is denoted as g. To avoid ambiguity, a goal-conditioned policy $\pi_g(s, a, g; \theta)$[1] is asked to accomplish a desired goal g. For our settings, the achieved goal is coordinate information, and achieved goal of a novel state is considered as the desired goal when feeding to a goal-conditioned policy.

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

A proper intrinsic reward function for the goal-conditioned policy is an indicator function with some tolerance, shown in Equation 2. If the "distance" between the achieved goal ag and the desired goal g is less than some threshold ϵ, the goal-conditioned policy receives a positive reward otherwise zero. Note that this function is also be used to judge whether agents reach the state boundary.

[1]With the respect of input to a goal-conditioned policy, s contains ag to keep notations simple.

However, the training of goal-conditioned policy is slow if this sparse reward function is used. Next, we introduce some techniques to deal with this problem.

$$r(ag_t, g_t) = d(ag_{t-1}, g_t) - d(ag_t, g_t) \tag{3}$$

Rewarding shaping introduces additional training rewards to guide the agent. Reward shaping is invariant to the optimal policy if shaping reward function is a potential function (Ng et al., 1999). Specifically, we define the difference of two consecutive distances (between the achieved goal and the desired goal) as shaping reward function, shown in Equation 7. Since shaping reward function is dense, it can lead to substantial reductions in learning time. Verification of the optimal goal-conditioned policy is invariant between this function and the indicator reward function is given in the appendix A.2. Alternatively, one can use also Hindsight Experience Replay (HER) (Andrychowicz et al., 2017) to train the goal-conditioned policy via replacing each episode with an achieved goal rather than one that the agent was trying to achieve. But one should be careful since HER changes the goal distribution for learning.

Each iteration the goal-conditioned policy is assigned with a goal selected from the priority queue. The goal is unchanged during an episode until it succeeds. However, random actions generated after goal success is unconditioned, and these samples are helpful when new states are selected as goals. Inspired by HER, we specify the last state as the goal for these action sequences, which is beneficial when new states are selected as goals.

4.3 Exploiting experience from exploration policy

For evaluation, we additionally train an unconditioned exploitation policy, which only takes the state as input. This policy learns experience collected by the exploration policy to overcome hard exploration in an off-policy learning fashion. At the same time, the exploitation policy also interacts with the environment to mitigate the side effect of exploration error (Fujimoto et al., 2019), a phenomenon that off-policy learning degenerates when data from the exploration policy is not correlated to the experience generated by the exploitation policy. Note that exploitation policy is trained with an RL objective to maximize expected discounted external return.

5 Experiment

In this section, we aim to answer the following research questions: (1) Does novelty-pursuit effectively maximize the state entropy? (2) Do the proposed goal-selection criterion and training techniques improve performance for IMGEP? (3) How does the performance of novelty-pursuit compare with the state-of-the-art approaches in complex environments? We conduct experiments from the 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 the appendix A.4 and A.5, respectively.

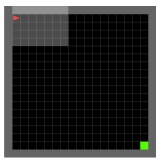
5.1 Environment settings

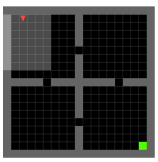
Here we briefly describe the environment settings (see Figure 3 for illustrations). Detailed settings are given in the appendix A.3.

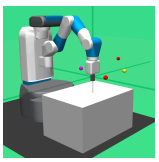
Empty Room & Four Rooms. An agent navigates in the maze of 19×19 to find the exit (Chevalier-Boisvert et al., 2018). The agent receives a time penalty until it finds the exit and receives a positive reward. The maximum reward for both two environments is $+1$, and the minimum reward is -1. Note that the observation for RND is a local-view image with shape of $(7, 7, 3)$.

FetchReach. A 7-DOF Fetch Robotics arm (simulated in the Mujoco (Todorov et al., 2012)) is asked to grip spheres above a table. There are 4 spheres on the table, 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 maximum reward for is $+4$, and the minimum reward is -1.

SuperMarioBros. A Mario agent with raw image observation explores to discover the flag. The reward is based on the score given by the NES simulator (Kauten, 2018) and is clipped into -1 and


(a) Empty Room


(b) Four Rooms


(c) FetchReach


(d) SuperMarioBros

Figure 3: Illustration of four environments.

$+1$ except $+50$ when getting a flag. There are 24 stages in the game, but we only focus on the 1-1, 1-2, and 1-3.

5.2 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. Note that we don't use any external reward. We consider the following baselines: 1) random: select actions uniformly; 2) bonus: a policy based on the exploration bonus using the prediction error of RND; 3) novelty-pursuit: the proposed method; 4) novelty-pursuit-planning: the proposed method with a perfect goal-conditioned policy; 5) novelty-pursuit-count: the proposed method with selecting goals based on visitation counts rather than the prediction errors; 6) novelty-pursuit-oracle: combination of two oracles used in 4) and 5); 7) maximum: the maximum state entropy over the whole state space. The results are summarized in Table 1.

Table 1: Entropy of state distribution at timesteps $20k$.

random	bonus	novelty-pursuit	novelty-pursuit-planning	novelty-pursuit-counts	novelty-pursuit-oracle	maximum
5.12	5.12	5.35	5.47	5.43	5.63	5.67

First, we can see that novelty-pursuit achieves a higher entropy than random and bonus strategy. Second, when the planning oracle and visitation counts are available, the novelty-pursuit improves by 0.12 and 0.08, respectively. Third, the combination of two oracles gives a near-perfect performance (the gap between the maximum state entropy is only 0.04). This result demonstrates that goal-condition exploration behaviors presented by novelty-pursuit can maximize the state entropy and validates the analysis in Section 3.

5.3 Ablation study of goal-selection and training techniques

In this section, we study the factors that contribute to our method by ablation experiments. Firstly, we focus on the criterion of goal-section in IMGEP. We compare novelty-pursuit with two other goal-selection methods: 1) random-selection: selecting states randomly from the experience buffer; 2) learning-progress: selecting a feasible state (goal success rate is between 0.3 and 0.7) with probability of 0.8 and an arbitrary visited state with the probability of 0.2, which is adopted from (Florensa et al., 2018). Results on the Empty Room are shown in Figure 4. Secondly, we study how goal-conditioned policy learning affect performance. We compare HER and the reward-shaping with distance reward (i.e., reward based on L_1 norm in our problem) used in (Forestier et al., 2017). Results on the Empty Room are shown in Figure 5.

From Figure 4, we see that IMGEP doesn't work when randomly selecting goals, but novelty-pursuit gives a greater boost compared to the learning-progress. We think the reason is that this heuristic method is brittle to the estimation of goal success rate and lack an explicit exploration objective. From Figure 5, we find that the IMGEP with HER or reward shaping outperforms than the IMGEP with distance reward. As discussed in Ng et al. (1999), reward based on distance may change the optimal behavior of goal-condition exploration policy, thus hurts the performance for IMGEP.

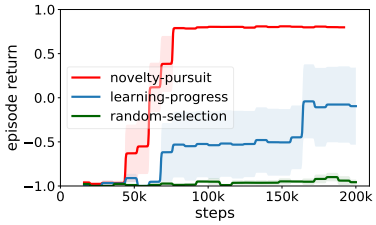


Figure 4: Comparison of goal-selection

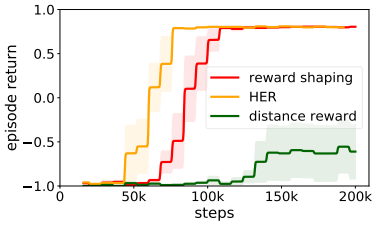


Figure 5: Comparison of training mechanisms

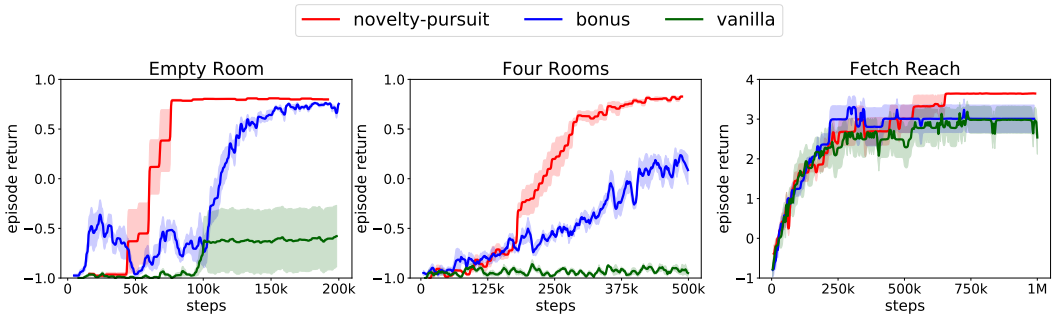


Figure 6: Average results over 5 seeds on the Empty Room, Four Rooms and FetchReach environments.

5.4 Evaluation on complex environments

In this section, we compare different methods in terms of external reward. We will see that without sufficient and efficient exploration, the policy may be stuck into the local optimum. Two baseline methods using reinforcement learning are considered: 1) vanilla: a vanilla policy; 2) bonus: the state-of-the-art method that combines the external reward and intrinsic reward (i.e., the prediction error given by RND).

First, we consider the previously used Empty Room and the Four Room environments. The results are shown in Figure 6. We see that the vanilla policy hardly finds the exit. Novelty-pursuit is comparative to bonus and outperforms bonus on the Four Rooms environment, where we observe that bonus is somewhat misled by the intrinsic reward though we have tried many weights to balance the external reward and intrinsic reward.

Secondly, we consider the FetchReach environment and results are shown in Figure 6. 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.

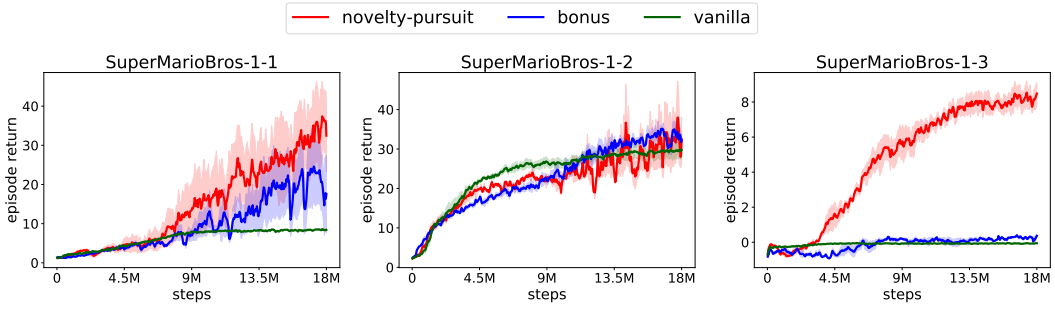


Figure 7: Average results over 3 seeds on SuperMarioBros.

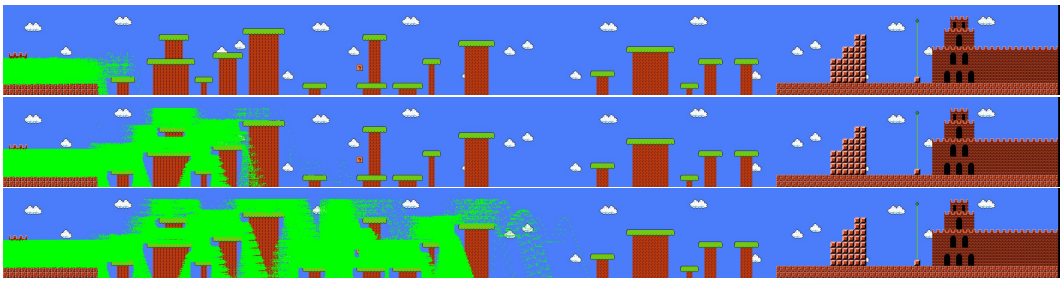


Figure 8: Trajectory visualization on SuperMarioBros-1-3. Trajectories are plotted in green cycles with the same number of samples. Top row: vanilla; Middle row: bonus; Bottom row: novelty-pursuit.

Finally, we consider the SuperMarioBros environments, which is very hard to discover the flag due to long horizon. Learning curves are plotted in Figure 7 and the final performance is listed in Table 2. On SuperMarioBros-1-1 and SuperMarioBros-1-3, novelty-pursuit outperforms other methods. There are dense rewards on SuperMarioBros-1-2 (i.e., the agent can easily get scores), thus all methods perform well. We observe that novelty-pursuit can easily get the flag on SuperMarioBros-1-1 and both of bonus and novelty-pursuit can occasionally get the flag on SuperMarioBros-1-2. It is very hard to explore SuperMarioBros-1-3 since the agent easily dies and receives a penalty. We plot trajectories of different methods on SuperMarioBros-1-3 in Figure 8. It is evident that novelty-pursuit achieves a deeper exploration than other methods. We attribute it to the disentanglement of exploration and exploitation.

Table 2: Final Performance over 3 seeds on SuperMarioBros.

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

6 Related work

Exploration. Traditionally, the exploration strategy is based on the exploitation policy that receives an external reward from the environment. Traditional exploration methods include injecting noise on action space (Mnih et al., 2015; Lillicrap et al., 2016) or parameter space (Plappert et al., 2018b; Fortunato et al., 2018), and adding the policy's entropy regularization (Schulman et al., 2017; Mnih et al., 2016).

For tabular Markov Decision Process, there are lots of work utilizing confidence based reward to balance exploration and exploitation (Kearns & Singh, 2002; Strehl & Littman, 2008; Kolter & Ng, 2009; Lattimore & Hutter, 2014). Several exploration strategies for deep RL based approximation visitation counts have been proposed in high-dimension space (Bellemare et al., 2016; Ostrovski et al., 2017). Another type of exploration is curiosity-driven exploration. These methods track the uncertainty of dynamic (Stadie et al., 2015; Pathak et al., 2017; Burda et al., 2019a;b) to explore intrinsic states. Deep (temporally extended) exploration via tracking the uncertainty of value function is studied in (Osband et al., 2016). Besides, maximum (policy) entropy reinforcement learning encourages exploration by maximizing the cumulative sum of external reward and policy entropy (Ziebart et al., 2008; Haarnoja et al., 2017; O'Donoghue et al., 2016; Haarnoja et al., 2018).

Recently, Hazan et al. (2019) introduce a new exploration objective: maximum state entropy. They provide an efficient algorithm when restricted to a known tabular MDP (a density estimator oracle is required for an unknown tabular MDP) and gives the theoretical analysis. We derive the criterion of goal generation based on the principle of maximum state entropy.

Our method is based on the framework of intrinsically motivated goal exploration processes (IMGEP) (Baranes & Oudeyer, 2009; Forestier et al., 2017; Péré et al., 2018). Go-Explore (Ecoffet et al., 2019) is reminiscent of IMGEP and achieves dramatic improvement on the hard exploration problem of Montezumas Revenge. But with the assumption that the environments are resettable or deterministic and many hand-engineering designs, Go-explore is restricted to specific environments.

Goal-conditioned Policy. By taking environment observation and desired goal as inputs, the goal-conditioned policy is expected to accomplish a series of tasks. Schaul et al. (2015) propose the universal value function approximator (UVFA) and train it by bootstrapping from the Bellman equation. However, training goal-condtioned policy is also still a challenging problem due to goal-condition reward is sparse (e.g. 1 for success, 0 for failure). Andrychowicz et al. (2017) propose hindsight experience replay (HER) by replacing each episode with an achieved goal rather than one that the agent was trying to achieve. This operation introduces more reward signals and serves as an implicit curriculum. Florensa et al. (2018) use a generator network to adaptively produce artificial feasible goals. We also use a goal-conditioned policy, but goals are selected from the experience buffer rather than being specified in advance. What's more, we utilize the technique of reward shaping (Ng et al., 1999) to accelerate training.

Learning from experience. Off-policy reinforcement learning algorithms such as DQN(Mnih et al., 2015), DDPG (Lillicrap et al., 2016), and ACER (Wang et al., 2017), reuse experience to improve data efficiency. Besides, how to additionally utilize (good) experience to overcome exploration dilemma is studied in (Oh et al., 2018; Goyal et al., 2019). These works are perpendicular to ours since we focus on how to discover these valuable states.

7 Conclusion

This paper bridges intrinsically motivated goal exploration process (IMGEP) and maximum state entropy exploration. We propose a method called novelty-pursuit from the connection. We demonstrate that exploration efficient of the proposed method called novelty-pursuit, and show that the chance of finding the (near-) optimal policy is high.

We notice that for high-level abstraction environments, it may not easy to extract goal information by hand. In that case, an unsupervised goal representation learning may be helpful. We also note that current training techniques based on an RL objective may not efficient for utilizing experience collected by the exploration policy. We leave these for future works.

References

Marcin Andrychowicz, Dwight Crow, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Proceedings of the 30th Advances in Neural Information Processing Systems*, pp. 5048–5058, 2017.

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

Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos. Unifying count-based exploration and intrinsic motivation. In *Proceedings of the 29th Advances in Neural Information Processing Systems 29*, pp. 1471–1479, 2016.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.

Yuri Burda, Harrison Edwards, Deepak Pathak, Amos J. Storkey, Trevor Darrell, and Alexei A. Efros. Large-scale study of curiosity-driven learning. In *Proceedings of the 7th International Conference on Learning Representations*, 2019a.

Yuri Burda, Harrison Edwards, Amos J. Storkey, and Oleg Klimov. Exploration by random network distillation. In *Proceedings of 7th International Conference on Learning Representations*, 2019b.

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. `https://github.com/maximecb/gym-minigrid`, 2018.

Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. `https://github.com/openai/baselines`, 2017.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *CoRR*, abs/1901.10995, 2019.

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1514–1523, 2018.

Sébastien Forestier, Yoan Mollard, and Pierre-Yves Oudeyer. Intrinsically motivated goal exploration processes with automatic curriculum learning. *CoRR*, abs/1708.02190, 2017.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Matteo Hessel, Ian Osband, Alex Graves, Volodymyr Mnih, Rémi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy networks for exploration. In *Proceedings of the 6th International Conference on Learning Representations*, 2018.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 2052–2062, 2019.

Anirudh Goyal, Philemon Brakel, William Fedus, Soumye Singhal, Timothy P. Lillicrap, Sergey Levine, Hugo Larochelle, and Yoshua Bengio. Recall traces: Backtracking models for efficient reinforcement learning. In *Proceedings of the 7th International Conference on Learning Representations*, 2019.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1352–1361, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 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.

Elad Hazan, Sham M. Kakade, Karan Singh, and Abby Van Soest. Provably efficient maximum entropy exploration. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 2681–2691, 2019.

Christian Kauten. Super Mario Bros for OpenAI Gym. GitHub, 2018. URL `https://github.com/Kautenja/gym-super-mario-bros`.

Michael J. Kearns and Satinder P. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232, 2002.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

J. Zico Kolter and Andrew Y. Ng. Near-bayesian exploration in polynomial time. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 513–520, 2009.

Tor Lattimore and Marcus Hutter. Near-optimal PAC bounds for discounted mdps. *Theoretical Computer Science*, 558:125–143, 2014.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *Proceedings of the 4th International Conference on Learning Representations*, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas 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*, 518(7540):529–533, 2015.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33nd International Conference on Machine Learning*, pp. 1928–1937, 2016.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning*, 1999.

Brendan O'Donoghue, Rémi Munos, Koray Kavukcuoglu, and Volodymyr Mnih. PGQ: combining policy gradient and q-learning. *CoRR*, abs/1611.01626, 2016.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 3875–3884, 2018.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. In *Proceedings of the 29th Advances in Neural Information Processing Systems*, pp. 4026–4034, 2016.

Georg Ostrovski, Marc G. Bellemare, Aäron van den Oord, and Rémi Munos. Count-based exploration with neural density models. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 2721–2730, 2017.

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*, pp. 2778–2787, 2017.

Alexandre Péré, Sébastien Forestier, Olivier Sigaud, and Pierre-Yves Oudeyer. Unsupervised learning of goal spaces for intrinsically motivated goal exploration. In *Proceedings of the 6th International Conference on Learning Representations*, 2018.

Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, Vikash Kumar, and Wojciech Zaremba. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *CoRR*, abs/1802.09464, 2018a.

Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y. Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. In *Proceedings of the 6th International Conference on Learning Representations*, 2018b.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 1312–1320, 2015.

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

Bradly C. Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *CoRR*, abs/1507.00814, 2015.

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.

Richard S. Sutton and Andrew G. Barto. *Introduction to reinforcement learning*. MIT Press, 1998.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude, 2012.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Rémi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. In *Proceedings of the 5th International Conference on Learning Representations*, 2017.

Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 33nd AAAI Conference on Artificial Intelligence*, pp. 1433–1438, 2008.

A APPENDIX

A.1 PROOF OF THEOREM 2

Suppose we have two choices x_i and x_j, we want to compare the difference between the entropy $H_{x_i}[d_{\pi 1:t+1}]$ visiting x_i and the entropy $H_{x_j}[d_{\pi 1:t+1}]$ by visiting x_j. Let $Z = \sum_i x_t(i)$ denotes visitation counts over all states. To simplify notation, we omit the subscript t for visitation counts. By definition, the difference is:

$$\begin{aligned} g(x_i, x_j) &= H_{x_i}[d_{\pi 1:t+1}] - H_{x_j}[d_{\pi 1:t+1}] \\ &= -\frac{x_i+1}{N+1}\log\frac{x_i+1}{N+1} - \frac{x_j}{N+1}\log\frac{x_j}{N+1} + \frac{x_j+1}{N+1}\log\frac{x_j+1}{N+1} + \frac{x_i}{N+1}\log\frac{x_i}{N+1} \\ &= (\frac{x_j+1}{N+1}\log\frac{x_j+1}{N+1} - \frac{x_j}{N+1}\log\frac{x_j}{N+1}) - (\frac{x_i+1}{N+1}\log\frac{x_i+1}{N+1} - \frac{x_i}{N+1}\log\frac{x_i}{N+1}) \end{aligned} \tag{4}$$

Let $f(x) = \frac{x+1}{N+1}\log\frac{x+1}{N+1} - \frac{x}{N+1}\log\frac{x}{N+1}$, which yields

$$g(x_i, x_j) = f(x_j) - f(x_i) \tag{5}$$

By looking at the derivative of $f(x)$, we know that $f(x)$ is a monotonically increasing function. Thus, $\forall x_i < x_j$, we have that $g(x_i, x_j) > 0$.

$$f'(x) = \frac{1}{N+1}\log(1+\frac{1}{x}) > 0 \tag{6}$$

In conclusion, unless x_i is minimal, we can always choose $x_j < x_i$ such that that $g(x_i, x_j) < 0$. Hence, visiting the states with the smallest visitation counts is optimal.

A.2 REWARD SHAPING FOR MULTI-GOAL REINFORCEMENT POLICY

Reward shaping is invariant to the optimal policy under some conditions (Ng et al., 1999). Here we verify that reward shaping introduced by us doesn't change the optimal policy for goal-conditioned policy. Adding up shaping rewards gives:

$$\begin{aligned} &\sum_{t=1}^{T} - d(ag_t, g) + d(ag_{t+1}, g) \\ &\quad = -d(ag_1, g) + d(ag_2, g) - d(ag_2, g) + d(ag_3, g) + \cdots - d(ag_T, g) + d(ag_{T+1}, g) \\ &\quad = -d(ag_1, g) + d(ag_{T+1}, g) \end{aligned} \tag{7}$$

For the optimal policy, $d(ag_{T+1}, g) = 0$, while $d(ag_1, g)$ is a constant. Thus optimal policy induced by sparse goal-conditioned reward is invariant to this reward shaping.

A.3 ENVIRONMENT PREPOSSESSING

Maze. Different from (Chevalier-Boisvert et al., 2018), we only use the image and coordination information as inputs. We only consider four actions: turn left, turn right, move forward and move backward. The maximum episode length is 190 for Empty Room, and 500 for Four Rooms. Each

time the agent receives a time penalty of $1/\mathrm{max\_episode\_length}$ and receives +1 when finding the exit.

FetchReach. We implement this environment based on *FetchReach-v0* in Gym (Brockman et al., 2016). The maximum episode length is 50. The locations 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 remove spheres outside of the table i.e., the valid x range: (1.0, 1.5), the valid y range is (0.45, 1.05), and valid z range is (0.45, 0.65).

SuperMarioBros. We implement this environment based on (Kauten, 2018) with 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. The maximum episode length is 800. The environment restarts 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 baselines (Dhariwal et al., 2017). For each environment, RND uses a similar network architecture. The predictor network has additional MLP layers than the predictor network.

A.5 HYPERPARAMETERS

Table 3 gives hyperparameters for ACER (Wang et al., 2017) on the maze and SuperMarioBros (the learning algorithm is RMSProp (Tieleman & Hinton, 2012)). DDPG (Lillicrap et al., 2016) used in Fetch Reach environments is based on the HER algorithm implemented in baselines (Dhariwal et al., 2017) 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 predictor network is 0.0005 and the optimization algorithm is Adam (Kingma & Ba, 2015) for all experiments, and the batch size of training data is equal to the product of rollout length and the number of parallel environments.

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

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
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	20M

For goal-conditioned policy, the weight α to balance the goal-conditioned reward and the external reward is 1 for all environments except 2 for SuperMarioBros. For bonus method used in Section 5, the weight β to balance the intrinsic reward and the external reward (i.e., $r' = r_{ext} + \beta r_{int}$) 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. We also do a normalization for the intrinsic reward by dividing the intrinsic rewards via a running estimate of the standard deviation of the sum of discounted intrinsic rewards.