Potential Driven Reinforcement Learning for Hard Exploration Tasks

Enmin Zhao[1,2], **Shihong Deng**[1], **Yifan Zang**[1,2], **Yongxin Kang**[2,1], **Kai Li**[1] and **Junliang Xing**[1,2*]
[1]Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
{zhaoenmin2018, shihong.deng, zangyifan2019, kangyongxin2018, kai.li, junliang.xing}@ia.ac.cn

Abstract

Experience replay plays a crucial role in Reinforcement Learning (RL), enabling the agent to remember and reuse experience from the past. Most previous methods sample experience transitions using simple heuristics like uniformly sampling or prioritizing those good ones. Since humans can learn from both good and bad experiences, more sophisticated experience replay algorithms need to be developed. Inspired by the potential energy in physics, this work introduces the artificial potential field into experience replay and develops Potentialized Experience Replay (PotER) as a new and effective sampling algorithm for RL in hard exploration tasks with sparse rewards. PotER defines a potential energy function for each state in experience replay and helps the agent to learn from both good and bad experiences using intrinsic state supervision. PotER can be combined with different RL algorithms as well as the self-imitation learning algorithm. Experimental analyses and comparisons on multiple challenging hard exploration environments have verified its effectiveness and efficiency.

1 Introduction

A fundamental challenge in Reinforcement Learning (RL) is how to trade-off between exploration and exploitation. The agent has to decide whether to greedily exploit what it already knows to maximize the expected cumulative reward or explore the unknown environment to gather more information and find a potentially better policy. Though simple exploration strategies such as ϵ-greedy action selection [Mnih *et al.*, 2015] or Gaussian control noise [Mnih *et al.*, 2016] using experience replay [Mnih *et al.*, 2015] work well on a wide range of tasks, they are inefficient in hard exploration tasks with sparse rewards such as the Atari games *Montezuma's Revenge* and *Freeway*. On these hard exploration tasks, standard RL algorithms perform poorly without even finding a single positive reward.

For the hard exploration tasks in RL, many previous methods try to leverage past good experiences to help the agent learn more effectively and efficiently. Prioritized Experience Replay (PriER) [Schaul *et al.*, 2016] introduces prioritizing experiences to replay import transitions more frequently. Imitation learning methods [Aytar *et al.*, 2018; Pohlen *et al.*, 2018] develop algorithms that utilize the good experiences from the expert demonstrations. However, expert demonstrations are often not available in practical applications. In this case, curiosity-based methods [Bellemare *et al.*, 2016; Pathak *et al.*, 2017] introduce some notions of curiosity or uncertainty as an exploration bonus to guide the learning process. However, such exploration methods still struggle with hard exploration games such as *Montezuma's Revenge*.

In another way, Hindsight Experience Replay (HER) [Andrychowicz *et al.*, 2017] and its extension methods [Zhao and Tresp, 2018; Ren *et al.*, 2019] encourage the agent to learn from the states it has encountered. They set random, high trajectory energy or valuable states as goals and perform well in robotic manipulation tasks. However, unlike robotic manipulation tasks, agents lose lives quickly in hard exploration tasks like *Montezuma's Revenge*, existing experience replay methods do not consider the episodes or states that might cause agents' death. When the desired goal-states may lead to agents' death, learning such strategies will not help achieve the task goal.

To deal with the above problems, we argue that not only the past good experiences but also the bad ones are beneficial for the learning of the agent's policy. To realize the above argument, one thing the agent must know is how to select appropriate and safe goals. Inspired by the Artificial Potential Field (APF) model used in the robot navigation task for obstacle avoidance, we define a potential exploration function for the hard exploration tasks. Then we introduce this potential exploration function into the experience replay methods and propose Potentialized Experience Replay (PotER). It is a new experience replay algorithm that automatically generates safe learning goals based on existing models and random exploration regardless of the reward existence. When no reward is obtained, the goals generated by our PotER can help the current RL model find a reasonable policy. Meanwhile, when new rewards are obtained, the goals generated by PotER can effectively help the current Self-Imitation Learning (SIL) [Oh *et al.*, 2018] model imitate both some good and bad experiences. To summarize, the main contributions of this work are

*Corresponding Author

in threefold:

- We introduce the ideas from the APF into the RL experience replay sampling procedure and potentialize all the states in the experience replay, which provides a general mechanism to enable the agent to learn effective policies from both the good experiences and the bad ones.
- We develop a new RL method by incorporating the proposed Potentialized Experience Replay with different RL algorithms, which naturally exhibits behavior in a curriculum learning manner to accomplish sub-task goals from easy to hard.
- We propose to further enhance the developed RL algorithms equipped with Potentialized Experience Replay using Self-Imitation Learning, which further improves the learning efficiency when doing hard exploration tasks with sparse rewards.

We evaluate the proposed PotER method with different RL algorithms on hard exploration tasks like *Montezuma's Revenge*. Extensive experimental analyses and comparisons demonstrate the effectiveness of the instantiated learning algorithms. The source code of this work is available at https://github.com/ZhaoEnMin/PotER.

2 Related Work

This section introduces some related works on dealing with the hard exploration tasks, including experience replay, effective exploration, and learning from demonstrations. The APF that inspired this work is also introduced in the last.

Experience replay. The success of DQN [Mnih *et al.*, 2015] and its variants owe much to the usage of experience replay buffer. Actor-critic RL algorithms such as A2C/A3C [Mnih *et al.*, 2016] and PPO [Schulman *et al.*, 2017] are known for their sampling inefficiency. Prioritized Experience Replay [Schaul *et al.*, 2016] improves the sample efficiency by prioritizing experiences based on TD-errors. It is a stochastic sampling method that interpolates between the pure greedy prioritization and the uniform random sampling, which gives different weights to transition tuples with TD-errors. Many existing methods either require importance sampling [Wang *et al.*, 2017; Gruslys *et al.*, 2018] or are limited to continuous control [Lillicrap *et al.*, 2016]. In another way, the Hindsight methods [Andrychowicz *et al.*, 2017; Zhao and Tresp, 2018; Ren *et al.*, 2019] encourage the agent to learn from the states it encountered. For hard exploration Atari games, Hindsight methods often fail in achieving the final goal, since many internal goals lead the agent to lose lives, and it is not appropriate to learn strategies for such goals.

Effective exploration. Various approaches have been proposed to improve the exploration effectiveness in RL tasks with sparse rewards, including count-based exploration [Bellemare *et al.*, 2016; Tang *et al.*, 2017] and prediction-based exploration [Pathak *et al.*, 2017; Savinov *et al.*, 2019; Burda *et al.*, 2019]. State visitation counts have been investigated to reduce the agent's uncertainty by visiting states or state-action pairs with low visit-counts. For RL tasks with large state space, a *pseudo count* can be constructed using a density model over the state space [Bellemare *et al.*, 2016], or cluster occurrence counts with locality-sensitive hashing to cluster states [Tang *et al.*, 2017]. Also, deep successor representation[Machado *et al.*, 2018] explicitly tries to lead the agent to states in the past, which are rare. Another class of exploration methods is based on the prediction error for a problem related to the agent's transitions. Directly predicting the next observations or their embeddings [Pathak *et al.*, 2017; Stadie *et al.*, 2015] suffers from the *noisy TV* problem in stochastic or partially observable environments. Recently, some works addressing this noisy TV problem in prediction-based exploration have been proposed, and the notable ones are episodic curiosity through reachability [Savinov *et al.*, 2019] and random network distillation [Burda *et al.*, 2019].

Learning from demonstrations. DQfD [Hester *et al.*, 2018] and Q-filter [Nair *et al.*, 2018] attempt to learn from expert demonstrations. SIL [Oh *et al.*, 2018] uses a replay buffer filled with past good experiences and learns to imitate what the agent has experienced but has not yet learned without using expert demonstrations. However, it is too hard for the agent to find some good experiences to imitate when the environment is complex, and the rewards are sparse. We propose a potential function to evaluate the experience replay buffer to help the agent set goals. When there is no reward, the agent can achieve this goal better through RL and SIL. When rewards are obtained, potential field function marks the *important* states, which lead to rewards as goals. In this way, agents can learn successful policy by imitating the experiences reaching the important states even the environment does not give a reward, which is commonly ignored by previous SIL methods.

Artificial potential field. APF [Khatib, 1986; Rimon and Koditschek, 1992] is a classical obstacle avoidance approach used for manipulators and mobile robots. The robot can form a planned path according to the designed gravitational field and repulsive force field, and avoid obstacles through the designed gravity and repulsive force to achieve the ultimate goal. APF has been mainly used in studies related to robots and path planning. To the best of our knowledge, there is no previous work that introduces the idea of APF into the hard exploration RL tasks.

3 Preliminaries

3.1 Reinforcement Learning

For the standard RL formalism, an environment is described by a set of states $\mathcal{S} = \{\mathbf{s}_t\}$, the initial state $\mathbf{s}_0$, a set of actions $\mathcal{A} = \{a_t\}$, a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$, a transition probability model $p(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t)$, and a discount factor $\lambda \in [0, 1]$. After executing an action $a_k = \pi(\mathbf{s}_t)$ at each task state $\mathbf{s}_t$ using a deterministic policy π, the agent will get a reward $r(\mathbf{s}_t, a_k)$ and a new state $\mathbf{s}_{t+1}$. State $\mathbf{s}_{t+1}$ is sampled from the distribution $p(\cdot|\mathbf{s}_t, a_t)$. The return is the sum of future rewards $R_t = \sum_{t=1}^{\infty} \lambda^{t-1} r(\mathbf{s}_{t+1}|\mathbf{s}_t, a_k)$. Agents need to maximize its excepted return $\mathcal{E}_{\mathbf{s}_i}[R_0|\mathbf{s}_i]$. The Q-function is $Q^\pi = \mathcal{E}[R_t|\mathbf{s}_t, a_t]$. Denote the optimal Q-function as Q^*, which can be obtained by the Bellman equation:

$$Q^*(\mathbf{s}, a) = \mathcal{E}_{\mathbf{s}' \sim p(\cdot|\mathbf{s}_t, a_t)}[r(\mathbf{s}, a) + \lambda \max_{a' \in \mathcal{A}} Q^*(\mathbf{s}', a')]. \quad (1)$$

3.2 Deep Q Networks and Experience Replay

Deep Q-Networks (DQN) [Mnih *et al.*, 2015] is an off-policy RL algorithm for discrete action spaces by learning to approximate the Q-function using a deep neural network. The Q-function is defined as $\pi_Q(\mathbf{s}) = \arg\max_{a\in A} Q(\mathbf{s}, a)$. Experience Replay is firstly proposed for DQN by buffering a certain size of the generated episodes into the computer memory. During training the agent, the stored transition tuples $(\mathbf{s}_t, a_t, r_t, \mathbf{s}_{t+1})$ are uniformly sampled to form a mini-batch to update the network parameters, with the loss function defined as $\mathcal{L} = \mathcal{E}(Q(\mathbf{s}_t, a_t) - y_t)^2$, where $y_t = r_t + \lambda \max_{a'\in A} Q(\mathbf{s}_{t+1}, a')$.

3.3 Artificial Potential Field

Artificial potential field (APF) [Rimon and Koditschek, 1992] is a common approach in the robot path planning task, which uses the potential energy function of the planned object through the artificial construction of each state in the space. Denoting the agent position as $\mathbf{x}$, the artificial potential energy $U(\mathbf{x})$ consists of two parts, the attractive potential energy $U_{att}(\mathbf{x})$ and the repulsive potential energy $U_{rep}(\mathbf{x})$:

$$U(\mathbf{x}) = U_{att}(\mathbf{x}) + U_{rep}(\mathbf{x}), \tag{2}$$

where $U_{att}(\mathbf{x})$ and $U_{rep}(\mathbf{x})$ are respectively denoted as:

$$U_{att}(\mathbf{x}) = \tfrac{1}{2} k_a \rho^2_{goal}(\mathbf{x}), \tag{3}$$

$$U_{rep}(\mathbf{x}) = \begin{cases} \frac{1}{2} k_r (\frac{1}{\rho_{obst}(\mathbf{x})} - \frac{1}{\rho_o})^2, & \rho(\mathbf{x}) \le \rho_o, \\ 0, & \rho(\mathbf{x}) > \rho_o. \end{cases} \tag{4}$$

The parameters of APF include the attractive parameter $k_a > 0$ and the repulsive parameter $k_r < 0$. The terms $\rho_{goal}(\mathbf{x})$ and $\rho_{obst}(\mathbf{x})$ are usually modeled using the Euclidean distance from the current position $\mathbf{x}$ to the goal position and the nearest obstacle position, respectively. The scalar ρ_o is the influence distance of the repulsive potential field.

4 Potential Driven Reinforcement Learning

In many hard exploration tasks, the sparse task rewards are not sufficient for the agent to learn effective strategies to accomplish the task objectives. The agent needs to explore different task states in the environment to see if they can lead to the final objective. Inspired by the studies in robot navigation, we introduce the ARF [Rimon and Koditschek, 1992] into RL for hard exploration tasks and propose a potential exploration function to model the feasibility of different task states in accomplishing the final task objective. We then potentialize the experience replay using this potential exploration function to make the sampling process in RL more efficient. The PotER can be incorporated into different RL algorithms. We exemplify this using an on-policy RL algorithm with self-imitation learning in the last part of this section.

4.1 Potential Exploration Function

In hard exploration tasks like the Atari 2600 games, we denote the task state as $\mathbf{s}$, which can be the visual input of the game. When the agent collects some episodes through the interactions with the environment. We sample the states of a trajectory in the experience replay buffer and denote a state trajectory with m states as $\mathbf{T}=(\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_m)$. A trajectory set with n trajectories is denoted as $\mathcal{T} = \{\mathbf{T}_0, \mathbf{T}_1, \ldots, \mathbf{T}_n\}$. In the studies of the artificial potential field for robot navigation, the distance to the goal state is often used to model the potential of one task state. For hard exploration tasks, however, the goal state is not knowable a priori, and the distance between two states often cannot be computed directly. Thus, we propose to use the initial task state and the obstacle (death) states as some anchor points to measure the potential of one state. A state with a longer distance to the initial state and all the obstacle (death) states is likely to have good potential towards the task goal.

To calculate the distance between two states, we record the trajectories during exploration and update the distance between two states as the shortest distance along different trajectories. Denoting $d_{\mathbf{T}}(\mathbf{s}_i, \mathbf{s}_j)$ as trajectory-level distance between $\mathbf{s}_i$ and $\mathbf{s}_j$, the task-level distance between these two states is estimated as:

$$d(\mathbf{s}_i, \mathbf{s}_j) = \min_{\mathbf{T}_k \in \mathcal{T}} d_{\mathbf{T}_k}(\mathbf{s}_i, \mathbf{s}_j). \tag{5}$$

With the above estimation of the distance function, we model the potential exploration energy $U_{pe}(\mathbf{s})$ of each state. The attractive potential exploration energy of one state $\mathbf{s}$ is modeled using the distance to the initial state, *i.e.*, $U_{att}(\mathbf{s}) = d(\mathbf{s}, \mathbf{s}_{init})$. The repulsive potential exploration energy of one state $\mathbf{s}$ is modeled using the distance to the nearest obstacle (death) state, *i.e.*, $U_{rep}(\mathbf{s}) = d(\mathbf{s}, \mathbf{s}_{obst})$. The scalar d_o is the influence distance of the repulsive potential field. Unlike the APF in robot navigation, the states in hard exploration tasks cannot be treated equally since the states do not appear with the same probability in the experience replay buffer. Inspired by Ant Colony Optimization [Dorigo and Birattari, 2010] that ants go to places with more pheromones, we let the potential energies be proportional to the state occurrence probability $p(\mathbf{s})$, which can be estimated using the frequency in the experience replay buffer. The potential exploration function of one state is finally model as:

$$U_{pe}(\mathbf{s}) = U_{att}(\mathbf{s}) + U_{rep}(\mathbf{s}), \tag{6}$$

where

$$U_{att}(\mathbf{s}) = \tfrac{1}{2} p(\mathbf{s}) k_a d^2(\mathbf{s}, \mathbf{s}_{init}), \tag{7}$$

$$U_{rep}(x) = \begin{cases} \frac{1}{2} p(\mathbf{s}) k_r (\frac{1}{d(\mathbf{s}, \mathbf{s}_{obst})} - \frac{1}{d_o})^2, & d(\mathbf{s}, \mathbf{s}_{obst}) \le d_o, \\ 0, & d(\mathbf{s}, \mathbf{s}_{obst}) > d_o. \end{cases} \tag{8}$$

4.2 Potentialized Experience Replay

The potential exploration function can be naturally incorporated into the experience replay to give each state in the experience replay a measure of importance. The potentialized experience replay (PotER) provides more flexibility for an RL algorithm in sampling from the experience replay. Unlike the prioritized experience replay [Schaul *et al.*, 2016], which gives higher TD error transitions higher importance, PotER provides a mechanism for a RL algorithm to learn from both the good experiences (*i.e.* the attractive potential

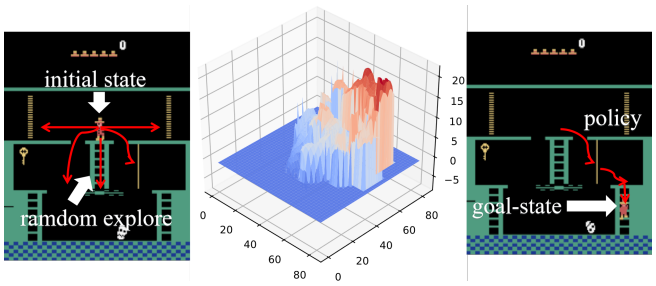

Figure 1: PotER works in *Montezuma's Revenge*. The agent uses random explorations from the initial state (left). By setting the internal goals (middle) using PotER, the agent learns the optimal policy more easily (right).

exploration energy $U_{att}(\mathbf{s})$) and the bad ones (*i.e.* the repulsive potential exploration energy $U_{rep}(\mathbf{s})$). PotER can also help RL algorithms to set internal goals for exploration when there is no external reward. Unlike the hindsight experience replay [Andrychowicz *et al.*, 2017], multiple internal goals can be set by PotER in a more meaningful order. Next, we will give it more explanations. It is well accepted that human learns much better when the training examples are not randomly presented but organized in a meaningful order, *e.g.*, gradually illustrating more concepts from easy to hard. When incorporating this principle into machine learning, the curriculum learning strategy [Bengio *et al.*, 2009] demonstrates promising results in supervised learning of deep neural networks. The PotER can be used to help the agent to form valuable internal goals by itself, which can be regarded as a form of curriculum learning.

At the beginning of the exploration, the agent uses random explorations from the initial state to sample transitions to construct the experience reply. Then PotER calculates the potential exploration function value for each collected state. The state with better potential and external rewards will get more attention in the next exploration. During exploration, for the explored trajectory set in current timestamp $\mathcal{T}^{(t)}$, the agent selects the state with the maximal potential as the internal goal for next exploration, *i.e.*,

$$\mathbf{s}_{goal}^{(t)} = \underset{\mathbf{s}\in\mathbf{T},\mathbf{T}\in\mathcal{T}^{(t)}}{\arg\max}\ (U_{pe}(\mathbf{s})), \tag{9}$$

When new sample transitions are collected into the experience replay buffer, if some old sample transitions need to be removed from the experience replay buffer, the states with the largest potential are guaranteed to be kept in the current replay buffer. With this kind of replay buffer management, the internal goals set in different timestamps by PotER satisfy the following relationships:

$$U_{pe}(\mathbf{s}_{goal}^{(0)}) \leq U_{pe}(\mathbf{s}_{goal}^{(1)}) \leq \cdots \leq U_{pe}(\mathbf{s}_{goal}^{(t)}) \leq \cdots . \tag{10}$$

where the internal goal sequence $U_{pe}(\mathbf{s}_{goal}^{(0)}), U_{pe}(\mathbf{s}_{goal}^{(1)}), \cdots$ now forms a curriculum [Bengio *et al.*, 2009].

Figure 1 shows an example that how PotER works in *Montezuma's Revenge* when the agent does not learn any policy. The goal-states PotER set are basically the same as the goals set by humans players and hierarchical imitation learning method using expert demonstrations [Le *et al.*, 2018]. If the RL algorithms learn how to get all the previous goal-states, PotER sets the last goal-state as the initial state and find the state with the maximum potential relative to the initial state as the next goal-state.

Algorithm 1: PotER based RL with SIL.

Initialize RL parameter θ, number of iterations used to set goal N, potentialized replay buffer ε, goals $\mathcal{G}$, timestamp $t_{goal} = 0$.
while *not complete* **do**
  for *iteration*=$1, 2, \ldots N$ **do**
    for *each step* **do**
      Execute an action $(s_t, a_t, r_t, s_{t+1}) \sim \pi_\theta(a_t|s_t)$
      Store transition $\varepsilon \leftarrow \varepsilon \cup (s_t, a_t, r_t, s_{t+1})$
  Set $\mathbf{s}_{goal}^{(t_{goal})}$ based on Eq. (9)
  Update goals $\mathcal{G} \leftarrow \mathcal{G} \cup \mathbf{s}_{goal}^{(t_{goal})}$
  $t_{goal} = t_{goal} + 1$
  Clear potentialized replay buffer $\varepsilon \leftarrow \emptyset$
  while π_θ *can not get* s_{goal} *in most cases* **do**
    Perform RL and SIL algorithms

4.3 PotER in RL with Self-imitation Learning

The PotER can be incorporated into different RL algorithms to improve its sampling efficiency and effectiveness. In the experiments, we will take the on-policy baseline algorithm A2C [Mnih *et al.*, 2016] as an example. To further speed up learning efficiency, more sophisticated learning strategies can also be incorporated. We adopt self-imitation learning (SIL) [Oh *et al.*, 2018] as another example combined with PotER to further verify the generalization ability of PotER. With self-imitation learning, the agent can quickly imitate both the past good episodes and some internal goals, which can gradually reach the final task goals. Figure 2 shows an example of how agents imitate experiences when there is no reward obtained. An overall description of PotER based RL with SIL is shown in Algorithm 1.

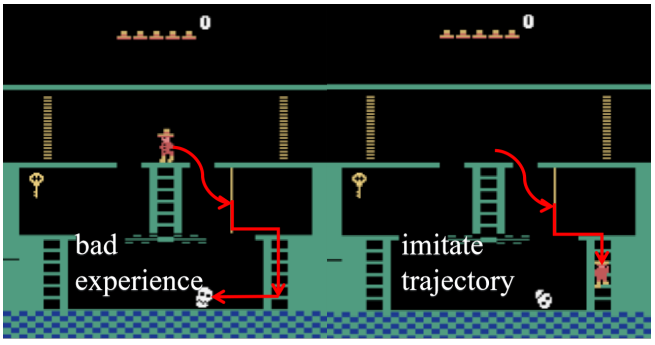

Figure 2: An example shows how agents imitate experiences when there is no reward obtained.

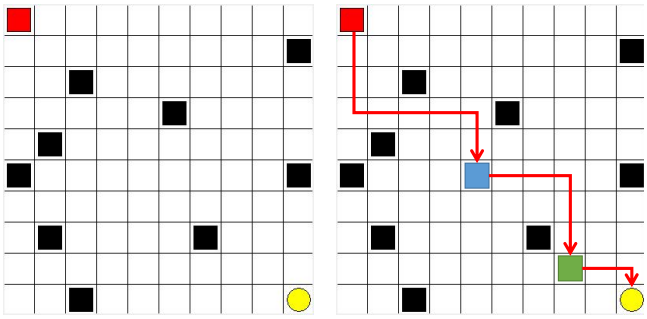

Figure 3: A simple maze (left) and the policy which is obtained by the tabular Q-learning with PotER (right). There are four actions in this environment: up, down, left, and right. The red square refers to the initial state, the black squares refer to the obstacles and the yellow circle refers to the state which contains a reward (set to 1). If the agent encounters the obstacles, it will lose its life and the environment will be reset.

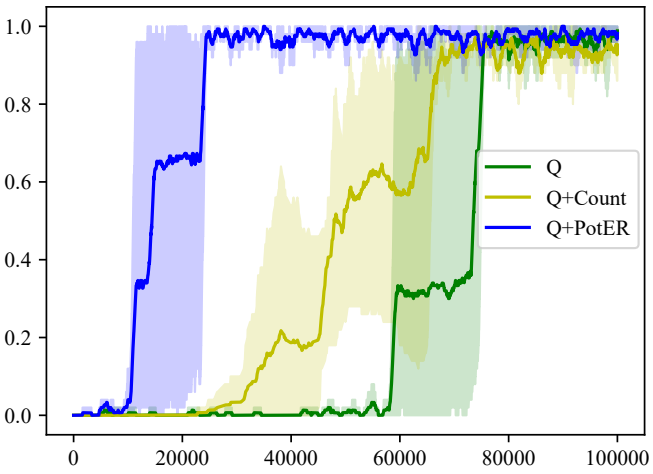

Figure 4: The success rate of the Q, Q+Count and Q+PotER algorithms in the maze environment. The learning curve shows the success rate (y-axis) of 3 different seeds at each time step (x-axis).

5 Experiments

To verify the efficiency of the PotER sampling algorithm, we evaluate its performance on two separate domains: 1) a simple maze with discrete state space (Fig. 3 left) and 2) the hard exploration Atari games with continuous state space.

5.1 Ablation Studies

We take the maze and the first room in *Montezuma's Revenge* as the experimental environments for our ablation studies. The main hyper-parameters of our algorithm are the number of iterations used to set goals N, the influence distance of the repulsive potential field d_o, the attractive parameter k_a and the repulsive parameter k_r. Because in different games, agents have different average steps to lose health, we set N as 50 to ensure there are not too many interactive samples and make our method suitable for most situations. Because our algorithm's core idea is to avoid death, we set k_r to $-\infty$ and k_a to any positive value. Such parameters can help agents try their best to avoid setting goal-states around the obstacles.

We conduct the following experiments on d_o. The agent loses its life in at least 4 and 32 time steps in the maze and the first room of *Montezuma's Revenge*, respectively. Table 1 shows the mean time steps to obtain 0.95 success rate in maze environment when $d_o \in \{0, 1, 2, 3, 4\}$ and Table 2 shows the mean time steps to get the first reward in the first room of *Montezuma's Revenge* when $d_o \in \{0, 10, 20, 30\}$. We use the best-performing d_o for the following experiments. In specific, for the maze games, we set d_o to 1. For the Atari games, we set d_o to 10.

5.2 Maze

The maze is a commonly used environment in the RL research community. The maze environment on the left of Fig. 3 is designed to verify that PotER can be well combined with tabular RL algorithms. We compare the tabular Q-learning with our PotER (Q+PotER) with the vanilla tabular Q-leaning (Q) and the tabular Q-learning with Count based exploration (Q+Count) [Strehl and Littman, 2008]. In specific, Q+Count gives an exploration bonus reward $r_{exp} = \beta/\sqrt{N(s)}$ for each state, where $N(s)$ is the visit count of state s and β is a hyper-parameter. In our experiment, we set β to 0.01. Fig. 4 shows the success rate of the Q, Q+Count, and Q+PotER algorithms in the maze environment.

It is clear that the baseline Q takes a long time to learn a good policy. Q+Count learns faster than Q because the exploration bonus reward r_{exp} helps the agent explore more effectively. Compared with Q and Q+Count, the Q+PotER algorithm learns fastest. The reason for this is Q+PotER helps the agent set valuable and safe goals like the blue one in Fig. 3 if the agent does not get any reward. Meanwhile, once the agent gets a reward by chance, Q+PotER helps exploit such good experiences by setting goals like the green one in Fig. 3 and quickly learns to get to the final goal. These experimental results demonstrate that PotER helps agents form a curriculum automatically and make good use of experiences.

d_o	0	1	2	3	4
Performance	2.81×10^4	2.42×10^4	2.74×10^4	3.86×10^4	7.42×10^4

Table 1: Mean time steps to obtain 0.95 success rate in the maze environment. The results are reported by using 3 random seeds.

5.3 Hard Exploration Atari Games

In the Atari experiments, we convert the 84×84 input RGB frames to gray-scale images. The input of the convolutional neural networks in DQN [Mnih *et al.*, 2015] and A2C [Mnih *et al.*, 2016] are the last 4 stacked gray-scale frames. For the SIL and SIL+PotER algorithms, we perform four SIL updates

d_o	0	10	20	30
Performance	2.9×10^6	2.38×10^6	2.59×10^6	2.73×10^6

Table 2: Mean time steps to get the first reward in the first room of *Montezuma's Revenge*. The results are reported by using 3 random seeds.

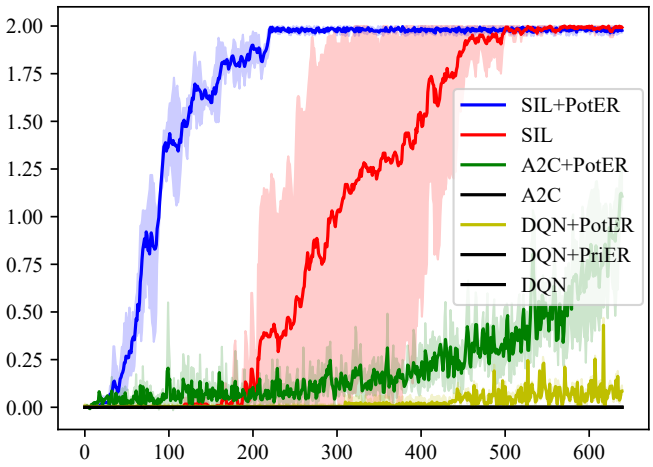

Figure 5: The learning curves in *Montezuma's Revenge* to get two rewards. The comparison algorithms include DQN, DQN+PriER, DQN+PotER, A2C, A2C+PotER, SIL and SIL+PotER. The learning curve shows the average score (y-axis) of 3 different seeds at each iteration (x-axis). Each iteration represents 4K time steps.

in training each model.

We use the first room of *Montezuma's Revenge* as the experimental environment to verify our PotER can be well combined with the Deep RL and the SIL algorithms. In particular, for the off-policy algorithms, we test DQN, DQN with Prioritized Experience Replay (DQN+PriER) [Schaul *et al.*, 2016], and DQN with our PotER (DQN+PotER). For the on-policy algorithms, we test A2C [Mnih *et al.*, 2016], A2C with PotER (A2C+PotER) , A2C with SIL (SIL) [Oh *et al.*, 2018] and A2C with both SIL and our PotER (SIL+PotER). The learning curves in the first room of *Montezuma's Revenge* to get two rewards are shown in Fig. 5. Each reward is adjusted to 1 to observe the learning speed of different methods easily.

As shown in Fig. 5, DQN, DQN +PriER, and A2C fail to learn a better-than-random policy. In contrast, DQN+PotER learns a policy better than the random policy, A2C+PotER learns a policy which obtains the first reward, SIL learns a policy which obtains both two rewards and SIL+PotER learns faster by incorporating PotER. The experimental results validate that PotER is very beneficial for RL algorithms in exploration tasks. Finally, the results of A2C+PotER and SIL+PotER show that PotER and SIL are complementary and can cooperate to enhance the performance further.

We use visualization to further verify that PotER gives a way to set up goals in both cases when there is a new reward, and when there is no reward at all. Fig. 6 shows the initial state and the 9 goal-states, which are automatically discovered by PotER in *Montezuma's Revenge*. In particular, if the agent learns how to obtain all the previous goal-states, then we set the last goal-state as the initial state and find the state with the maximum potential energy as the next goal-state. These discovered goal-states are similar to the goals set by human players, and the policy the agent learned is shown using the red arrow. These intuitive visualizations demonstrate the effectiveness of PotER in hard exploration tasks.

We investigate how useful our PotER is for several other

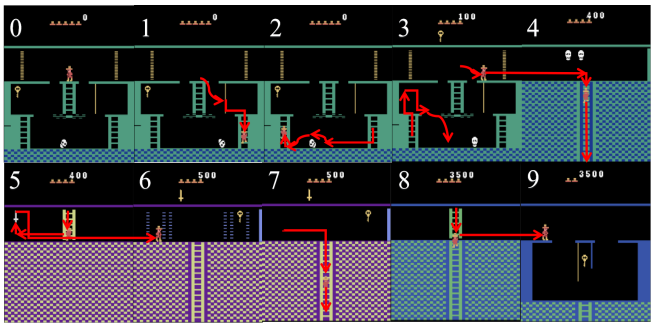

Figure 6: The initial state and the 9 goal-states which are automatically discovered by PotER in *Montezuma's Revenge*, the learned policy is shown as red arrows.

hard exploration Atari games. In specific, we apply the SIL+PotER algorithm to some hard exploration games like *Freeway*, *Montezuma's Revenge*, *Gravitar* and *Private Eye*. These four environments have common characteristics, *i.e.*, sparse rewards, and moving distractor objects.

We compare SIL+PotER against TRPO-AE-SimHash (SimHash) [Tang *et al.*, 2017], Curiosity-Driven Exploration (ICM) [Pathak *et al.*, 2017], Self-Imitation Learning (SIL) [Oh *et al.*, 2018], Random Network Distillation (RND) [Burda *et al.*, 2019] and Exploration with Mutual Information (EMI) [Kim *et al.*, 2019]. Table 3 shows that our SIL+PotER achieves better results on the four hard exploration games. These results confirm that PotER is an effective method to set goals for RL agents using experiences. The results of SIL+PotER and SIL demonstrate that PotER can help SIL learn more quickly and explore more effectively. In SIL+PotER, not only the past good experiences but also some bad experiences can be effectively exploited to help the agent imitate and explore.

6 Conclusion

In this paper, we propose PotER, a new sampling method for effective exploration in reinforcement learning, which aims to drive the agent to the goal-states to achieve the final goal. PotER helps the agent to set up goal-states automatically through experience replays. The detailed experimental results on the maze and some hard exploration games demonstrate that PotER is beneficial for a wide range of RL algorithms. PotER provides a simple yet competing baseline for solving hard exploration tasks.

	Montezuma	Gravitar	Private Eye	Freeway
SimHash	75	482	N/A	33.5
ICM	1011	424	**15277**	33.6
SIL	2497	2312	8325	**33.8**
RND	4017	552	4782	33.7
EMI	387	558	N/A	**33.8**
SIL+PotER	**6439**	**2908**	9203	**33.8**

Table 3: Comparisons on four hard exploration Atari games. The results are reported by using 5 random seeds in 50M time steps.

References

[Andrychowicz *et al.*, 2017] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058, 2017.

[Aytar *et al.*, 2018] Yusuf Aytar, Tobias Pfaff, David Budden, Thomas Paine, Ziyu Wang, and Nando de Freitas. Playing hard exploration games by watching youtube. In *Advances in Neural Information Processing Systems*, pages 2935–2945, 2018.

[Bellemare *et al.*, 2016] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

[Bengio *et al.*, 2009] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning*, pages 41–48, 2009.

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

[Dorigo and Birattari, 2010] Marco Dorigo and Mauro Birattari. *Ant colony optimization*. Springer, 2010.

[Gruslys *et al.*, 2018] Audrunas Gruslys, Mohammad Gheshlaghi Azar, Marc G Bellemare, and Remi Munos. The reactor: A sample-efficient actor-critic architecture. *International Conference on Learning Representations*, 2018.

[Hester *et al.*, 2018] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep Q-learning from demonstrations. In *AAAI Conference on Artificial Intelligence*, pages 3223–3230, 2018.

[Khatib, 1986] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous Robot Vehicles*, pages 396–404, 1986.

[Kim *et al.*, 2019] Hyoungseok Kim, Jaekyeom Kim, Yeonwoo Jeong, Sergey Levine, and Hyun Oh Song. Emi: Exploration with mutual information. In *International Conference on Machine Learning*, pages 3360–3369, 2019.

[Le *et al.*, 2018] Hoang Le, Nan Jiang, Alekh Agarwal, Miroslav Dudik, Yisong Yue, and Hal Daumé, III. Hierarchical imitation and reinforcement learning. In *International Conference on Machine Learning*, pages 2917–2926, 2018.

[Lillicrap *et al.*, 2016] 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 *International Conference on Learning Representations*, 2016.

[Machado *et al.*, 2018] Marlos C. Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. Eigenoption discovery through the deep successor representation. In *International Conference on Learning Representations*, 2018.

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

[Mnih *et al.*, 2016] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

[Nair *et al.*, 2018] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *IEEE International Conference on Robotics and Automation*, pages 6292–6299, 2018.

[Oh *et al.*, 2018] Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *International Conference on Machine Learning*, pages 3878–3887, 2018.

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

[Pohlen *et al.*, 2018] Tobias Pohlen, Bilal Piot, Todd Hester, Mohammad Gheshlaghi Azar, Dan Horgan, David Budden, Gabriel Barth-Maron, Hado van Hasselt, John Quan, Mel Večerík, et al. Observe and look further: Achieving consistent performance on atari. *arXiv preprint arXiv:1805.11593*, 2018.

[Ren *et al.*, 2019] Zhizhou Ren, Kefan Dong, Yuan Zhou, Qiang Liu, and Jian Peng. Exploration via hindsight goal generation. In *Advances in Neural Information Processing Systems*, pages 13485–13496, 2019.

[Rimon and Koditschek, 1992] Elon Rimon and Daniel E Koditschek. Exact robot navigation using artificial potential functions. In *International Conference on Robotics and Automation*, pages 501–518, 1992.

[Savinov *et al.*, 2019] Nikolay Savinov, Anton Raichuk, Raphaël Marinier, Damien Vincent, Marc Pollefeys, Timothy Lillicrap, and Sylvain Gelly. Episodic curiosity through reachability. In *International Conference on Learning Representations*, 2019.

[Schaul *et al.*, 2016] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations*, 2016.

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

[Stadie *et al.*, 2015] Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. In *Advances in Neural Information Processing Systems Workshop*, 2015.

[Strehl and Littman, 2008] 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.

[Tang *et al.*, 2017] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. Exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2753–2762, 2017.

[Wang *et al.*, 2017] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. In *International Conference on Learning Representations*, 2017.

[Zhao and Tresp, 2018] Rui Zhao and Volker Tresp. Energy-based hindsight experience prioritization. In *Conference on Robot Learning*, pages 113–122, 2018.