Hindawi
International Journal of Intelligent Systems
Volume 2023, Article ID 4705291, 14 pages
https://doi.org/10.1155/2023/4705291

Research Article

Learning Diverse Policies with Soft Self-Generated Guidance

Guojian Wang,[1,2,3] **Faguo Wu**,[2,3,4,5] **Xiao Zhang**,[1,2,3,5] **and Jianxiang Liu**[1,2,3]

[1]*School of Mathematical Sciences, Beihang University, Beijing, China*
[2]*Key Laboratory of Mathematics, Informatics and Behavioral Semantics, Education of Ministry, Beijing, China*
[3]*Peng Cheng Laboratory, Shenzhen, Guangdong, China*
[4]*Institute of Artificial Intelligence, Beihang University, Beijing, China*
[5]*Zhongguancun Laboratory, Beijing, China*

Correspondence should be addressed to Xiao Zhang; xiao.zh@buaa.edu.cn

Received 14 December 2022; Revised 3 February 2023; Accepted 21 March 2023; Published 18 April 2023

Academic Editor: Tao Li

Reinforcement learning (RL) with sparse and deceptive rewards is a significant challenge because nonzero rewards are rarely obtained, and hence, the gradient calculated by the agent can be stochastic and without valid information. Recent work demonstrates that using memory buffers of previous experiences can lead to a more efficient learning process. However, existing methods usually require these experiences to be successful and may overly exploit them, which can cause the agent to adopt suboptimal behaviors. This study develops an approach that exploits diverse past trajectories for faster and more efficient online RL, even if these trajectories are suboptimal or not highly rewarded. The proposed algorithm merges a policy improvement step with an additional policy exploration step by using offline demonstration data. The main contribution of this study is that by regarding diverse past trajectories as guidance, instead of imitating them, our method directs its policy to follow and expand past trajectories, while still being able to learn without rewards and gradually approach optimality. Furthermore, a novel diversity measurement is introduced to maintain the diversity of the team and regulate exploration. The proposed algorithm is evaluated on a series of discrete and continuous control tasks with sparse and deceptive rewards. In comparison with the existing RL methods, the experimental results indicate that our proposed algorithm is significantly better than the baseline methods in terms of diverse exploration and avoiding local optima.

1. Introduction

In recent years, deep reinforcement learning has been demonstrated to effectively solve sequential decision-making problems in a great deal of application domains, such as computer and board games playing [1, 2], continuous control [3–5], and robot navigation [6]. Despite these success stories, reinforcement learning with sparse and deceptive rewards remains a challenging problem in the field of RL [7–9] because maintaining a good trade-off between exploration and exploitation becomes more intractable in tasks with sparse and deceptive rewards.

Optimizing with the sparse feedback requires agents to reproduce past good trajectories efficiently and avoid being struck in local optima. In tasks with large state spaces and sparse rewards, a desired positive reward can only be received after the agent continuously executes many appropriate actions, and hence, the agent can rarely collect highly rewarded trajectories. In the meantime, the gradient-based parameter update of modern deep RL algorithms might result in a catastrophic forgetting of past experiences because the gradient-based parameter update is incremental and slow, and it has a global impact on all parameters of the policy and the value function [10]. Therefore, the agent might suffer from severe performance degradation when the ideal trajectories with highest returns are rarely collected, incurring the unstable policy optimization process. Finally, agents can adopt suboptimal myopic behaviors and be struck in local optima due to overly exploiting past imperfect experiences and do not explore the state-action space systematically.

Such tasks with sparse and deceptive reward signals are common in real-world problems. Recently, many workhas studied how making use of the nonparametric memory of past experiences improves policy learning in RL. Prioritized experience replay [11] proposes prioritizing past experiences before learning the policy parameters from them. Self-imitation learning [9, 12, 13] builds a memory buffer to store past good trajectories and thus can rapidly learn the right strategies from these past experiences when faced with a similar situation. Memory-augmented policy optimization [14] leverages a memory buffer of prior highly rewarded trajectories to reduce the estimate variance of the policy gradient. Episodic reinforcement learning [15] uses past good experiences that are stored in an episodic memory buffer to supervise an agent and force the agent to learn good strategies. Model-free episodic control [16] and neural episodic control [17] use episodic memory modules to estimate the state-action values. Diverse trajectory-conditionedself-imitation learning [10] proposes learning a novel trajectory-conditioned policy that follows and expands diverse trajectories in the memory buffer.

These existing work uses nonparametric memories of past good experiences to rapidly latch onto successful strategies and improve the learning efficiency of policy and value function. However, we must note that the exploitation of the past good trajectories described in the above-mentioned work might hurt the performance of the agent in tasks with sparse and deceptive reward functions. There are two main reasons that can cause the performance degradation of algorithms in tasks with sparse and deceptive rewards. First, the past self-generated trajectories stored in the memory buffer are imperfect. The trajectories in the memory buffer are not gold trajectories but highly rewarded trajectories collected by accident. Second, the RL agent usually limits its exploration to a small portion of the state-action space because of prior experience and network initialization [18]. In this way, the agent can easily generate trajectories leading to suboptimal goals. The exploitation of these successful suboptimal trajectories with limited directions might cause the agent to learn myopic behaviors. This will limit the agent's exploration region and prevent the agent from discovering alternative strategies with higher returns.

In this study, we conduct a practical RL algorithm by regarding previous diverse trajectories as guidance in the sparse reward setting, even if these trajectories are suboptimal or not highly rewarded. Our critical insight is that we can utilize imperfect trajectories with or without sparse rewards to regulate the direction of policy optimization while preserving the diversity of agents by virtue of two steps. In the first policy improvement (PI) step, we develop a new method that exploits the self-generated guidance to enable the agent to reproduce diverse past trajectories efficiently, while encouraging agents to smoothly expand these trajectories and visit underexplored regions of the state-action space gradually. Specifically, our method guides agents to revisit the regions where past good trajectories are located by minimizing the distance of state representations of trajectories. Meanwhile, our method allows for flexibility in the action choices to help the agent choose different actions and visit novel states. In the second policy exploration (PE) step, we introduce a novel diversity measurement to drive the different agents on the team to reach diverse regions of the state space and maintain the diversity of an ensemble of agents. By designing this new diversity measurement, our algorithm does not have to maintain a set of autoencoders [19] and can prevent the agents from being stuck in local optima. Our main contributions are summarized as follows:

(1) We develop a novel two-step RL framework that makes better use of diverse self-generated demonstrations to promote learning performance in tasks with sparse and deceptive rewards
(2) To the best of our knowledge, this is the first study that regards self-generated imperfect demonstration data as guidance and shows the importance of exploiting these previous experiences to indirectly drive exploration
(3) We illustrate that by regarding the agent's self-generated demonstration trajectories as guidance, the agent can reproduce diverse past trajectories quickly and then smoothly move beyond to result in a more effective policy
(4) A new diversity metric for the ensemble of agents has been proposed to achieve deep exploration and avoid being stuck in local optima
(5) Our method achieves superior performance over other state-of-the-art RL algorithms on several challenging physical control benchmarks with sparse and deceptive rewards in terms of diverse exploration and improving learning efficiency

The rest of this article is organized as follows: Section 2 describes the progress of the related work. Section 3 briefly describes the preliminary knowledge for the article. Section 4 introduces our proposed method for reinforcement learning with sparse and deceptive rewards. Experimental results are presented in Section 5. Finally, we draw our conclusions in Section 6.

2. Related Work

2.1. Exploration and Exploitation. It is a long-standing and intractable challenge to balance exploration and exploitation in the field of RL. The exploration enables the agent to visit the underdiscovered state-action space and collect trajectories with higher returns. The exploitation, on the contrary, encourages the agent to make use of what it already knows to maximize the expected returns. There is lots of work which aims to improve the exploration ability of the RL agent. Some work proposes to add stochastic noise to the output actions [1, 3–5, 20] or parameters of policy and value networks [21–23] to encourage exploration. Many other work defines the concept of intrinsic reward to promote the agent to visit the underexplored state-action space [24–26]. Furthermore, there are plenty of work which introduces a new optimization objective to change the gradient update direction of parameters [18, 27–29]. Another straightforward idea to expand the agent's exploratory area is to employ a team of agents to explore the environment collaboratively and share the collected experiences with each other [30–32]. In all these methods, although the agent can access the underdiscovered area due to randomness and artificial incentive, it is still difficult for the agent to get rid of local optima in long horizon, sparse reward tasks because it is rare for the agent to collect trajectories with nonzero rewards in these hard-exploration environments. Our method maintains different memory buffers to store past good trajectories for each agent in the team, and an agent only shares past good trajectories with each other in order to calculate a diversity measurement.

2.2. Memory-Based RL. The existence of a memory buffer enables the agent to store and utilize past experiences to aid in online RL training. Many prior work proposed storing past good experiences in replay buffers with a prioritized replay mechanism to accelerate the training process [6, 11, 33]. Episodic reinforcement learning methods [15–17] memorize past good episodic experiences by maintaining and updating a look-up table and act upon these good experiences in the decision-making process. Self-imitation learning (SIL) methods [9, 12] train the agent to imitate the highly rewarded trajectories with the SIL and GAIL objective, respectively. There are many other previous works where the agent learns a range of diverse exploratory policies based on episodic memory [10, 34]. Unlike these previous methods, our method encourages the agent to visit the part of the state space where the agent can obtain higher rewards by calculating the distance between the current trajectory and the past good trajectory.

2.3. Imitation Learning (IL). The goal of imitation learning is to train a policy by imitating the demonstration data generated by human experts. Behavior clone (BC) is regarded as a simple IL approach where the unknown expert policy is estimated from demonstration data by supervised learning. BC methods, however, usually suffer from the heavy distribution shift problem [35]. Inverse reinforcement learning (IRL) solves forward RL problems by recovering the reward function from demonstration data [36, 37]. Generative adversarial imitation learning (GAIL) [38] formulated the IL problem as a distribution matching problem, which can avoid estimating the reward function. All these IL methods rely on the availability of high-quality and sufficient human demonstrations. In contrast, our method treats the past good trajectories generated by the agent as demonstration data.

3. Preliminaries

In this study, we show that the MMD metric can be used as a distance constraint to prevent the agent from falling into local optima. Using the exterior penalty function method, we transform the constrained RL optimization problem into an unconstrained optimization problem, and the MMD distance can be regarded as a kind of intrinsic reward. Our method can be naturally combined with the hierarchical reinforcement learning algorithm, and the policy gradient can adjust pretrained skills and the high-level policy during the training phase.

3.1. Reinforcement Learning. We consider a discounted Markov decision process (MDP) defined by a tuple $M = (\mathcal{S}, \mathcal{A}, P, r, “_0, ”)$, in which $\mathcal{S}$ is a continuous state space, $\mathcal{A}$ is a (discrete or continuous) action space, P: $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \longrightarrow [0, 1]$ is the transition probability distribution, r: $\mathcal{S} \times \mathcal{A} \longrightarrow [R_{\min}, R_{\max}]$ is the reward function, in which we assume that the minimum and maximum value of the reward function is $\mathrm{R}_{\min}$ and $\mathrm{R}_{\max}$, respectively. Further,ρ_0: $\mathcal{S} \longrightarrow [0, 1]$ is the distribution of the initial state s_0, and $\gamma \in [0, 1]$ is a discount factor. A stochastic policy π_θ: $\mathcal{S} \longrightarrow \mathcal{P}(\mathcal{A})$, parametrized by θ, maps the state space $\mathcal{S}$ to the set of probability distributions $\mathcal{P}(\mathcal{A})$ over the action space $\mathcal{A}$. The state-action value function is defined as $Q(s_t, a_t) = \mathbb{E}_{\rho_0, P, \pi}[\sum_{t'=t}^{\infty} \gamma^{t'-t} r(s_{t'}, a_{t'})]$.

In general, the objective of RL algorithms is to find an optimal policy π_θ that maximizes the expected discounted return:

$$\eta(\pi_\theta) = \mathbb{E}_\tau\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]. \tag{1}$$

We use $\tau = (s_0, a_0, \ldots)$ to denote the entire history of the state and action pairs, where $s_0 \sim \rho_0(s_0)$, $a_t \sim \pi_\theta(a_t|s_t)$, and $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$.

Similar to [39], when $\gamma = 1$, we define the stationary state-visitation distribution for the policy π_θ by $\rho_\pi(s) = \lim_{t \longrightarrow \infty} P(s_t = s|s_0, \pi)$, where the initial state $s_0 \sim \rho_0$. The expected discounted return can be rewritten as $\mathbb{E}_{\rho_\pi(s,a)}[r(s, a)]$, where $\rho_\pi(s, a) = \rho_\pi(s)\pi_\theta(a|s)$ is the state-action visitation distribution.

3.2. Policy Gradient Algorithms. The study of this paper is based on policy gradient RL algorithms, which is one of two classes of RL algorithms. Policy gradient algorithms use gradients to iteratively optimize policy parameters of the agent. Here, we give a brief introduction of a well-known policy gradient algorithm: proximal policy optimization (PPO).

3.2.1. PPO. PPO [20] uses a clipped objective function to constrain the step size during an update and prevent drastic parameter changes. This leads to a stable training process compared with other algorithms. The PPO agent adjusts the parameters of the policy by maximizing the following clipped objective function:

$$\max_{\theta} \mathscr{L}(\theta) = \mathbb{E}_{\theta}\left[\min\left(\frac{\pi_{\theta}\left(a_t|s_t\right)}{\pi_{\theta_{old}}\left(a_t|s_t\right)}, \operatorname{clip}\left(\frac{\pi_{\theta}\left(a_t|s_t\right)}{\pi_{\theta_{old}}\left(a_t|s_t\right)}, 1-\varepsilon, 1+\varepsilon\right)\right)A\left(s_t, a_t\right)\right], \tag{2}$$

where $\pi_{\theta}(\cdot)$ and $\pi_{\theta_{\text{old}}}(\cdot)$ represent the current policy and the old policy, respectively, and ε is the clipping ratio, which is a hyperparameter that is empirically determined.

3.3. Maximum Mean Discrepancy. Maximum mean discrepancy (MMD) can be used to measure the difference (or similarity) between two probability distributions [29, 40–43]. Let $\mathbf{x}: = \{x_0, x_1, \ldots, x_l\}$ and $\mathbf{y}: = \{y_0, y_1, \ldots, y_m\}$ be two sets of samples, which are taken independently and identically from two distributions p and q. p and q are defined in a nonempty compact metric space $\mathbb{X}$. Then, we can define MMD as

$$\operatorname{MMD}(p, q, \mathscr{F}) \coloneqq \sup_{f \in \mathscr{F}}\left(\mathbb{E}_{x\sim p}[f(x)] - \mathbb{E}_{y\sim q}[f(y)]\right), \tag{3}$$

where $x \in \mathbf{x}$, $y \in \mathbf{y}$ and $\mathscr{F}$ is a class of functions on $\mathbb{X}$. If $\mathscr{F}$ satisfies the condition $p = q$, if and only if $\mathbb{E}_{x\sim p}[f(x)] = \mathbb{E}_{y\sim q}[f(y)]$, $\forall f \in \mathscr{F}$, then MMD is a metric in the space of probability distributions in $\mathbb{X}$ measuring the discrepancy between any two distributions p and q [44].

What kind of function class makes the MMD a metric? According to literature [40], the space of bounded continuous functions $\mathscr{F}$ on $\mathbb{X}$ satisfies the condition, but it is intractable to compute the MMD with finite samples in such a huge function class. Fortunately, when $\mathscr{F}$ is a reproducing kernel Hilbert space $\mathscr{H}$ defined by kernel $k(\cdot,\cdot)$, it is enough to uniquely identify whether $p = q$ or not, and the MMD is tractable in the space:

$$\operatorname{MMD}^2(p, q, \mathscr{H}) = \mathbb{E}\left[k\left(x, x^{'}\right)\right] - 2\mathbb{E}[k(x, y)] + \mathbb{E}\left[k\left(y, y^{'}\right)\right]. \tag{4}$$

4. Proposed Approach

In this section, we formulate a novel RL framework named Policy Optimization with Soft self-generated guidance and diverse Exploration (POSE). The proposed method utilizes a team of agents to explore the environment simultaneously and encourages them to visit nonoverlapping areas of state spaces. Every agent in the team maintains a memory buffer storing past good trajectories, and these offline data can be regarded as guidance to enable the agents to revisit diverse regions in the state space where the agents can receive high rewards and drive deep exploration.

4.1. Overview of POSE. One feasible approach to achieve better exploration in challenging tasks with sparse and deceptive rewards is to simultaneously employ a team of agents and enforce them to explore different parts of the state-action space. In this way, diverse policies can be learned by different agents in the team, which prevent agents from being stuck in local optima.

As shown in Figure 1, our POSE method employs a team of K agents to interact with the environment and generate many state-action sequences. Different from the multiagent RL setting [46], where all agents live in a shared environment and the action of an agent can affect other agents' states and action choices, in our design, each agent exists in an independent copy of the same environment and has no interaction with other agents in the team when sampling data. In each training iteration, every agent of the team collects an on-policy training batch $\mathscr{B}$ containing M trajectories in the environment. Meanwhile, we also maintain a replay buffer $\mathscr{M}$ for each agent to store specific trajectories generated in previous rollouts. Furthermore, POSE is introduced in Algorithm 1 of Appendix A in detail. Next, we will explain how to organize the trajectory buffer.

4.1.1. Organizing Trajectory Replay Memory for Exploration and Exploitation. We maintain a trajectory replay memory $\mathscr{M}^i = \left\{(\tau_{(1)}, e_{(1)}, n_{(1)}), (\tau_{(2)}, e_{(2)}, n_{(2)}), \cdots\right\}$ for the i-th agent of the team. The number of trajectories in $\mathscr{M}^i$ is no more than k, and hence, $\tau_{(j)}$ in $\mathscr{M}^i$ is one of the top-k trajectories ending with the state embedding $e_{(j)}$. $n_{(j)}$ is the number of steps of the trajectory $\tau_{(j)}$. Different from the SIL method [9], in which only the successful trajectory with a return above a certain threshold is eligible to be added to these memories, the imperfect trajectories that are not highly rewarded or even do not reach any goal can also be considered as offline demonstrations in our method. For example, the trajectory is on the path to some goal, although it does not reach the goal. Furthermore, every replay memory only stores trajectories with similar state embeddings, which correspond to the same goal or state region. If the embedding e of a new trajectory τ is similar with the trajectories in the i-th agent's memory and this trajectory is better than (i.e., higher return, shorter trajectory length, or shorter distance to the goal) the worst trajectory of this memory, we replace this worst trajectory by this new entry $\{\tau, e, n\}$.

4.1.2. Guiding Agent to Reproduce Trajectory to State of Interest. To achieve better performance than existing RL methods in the sparse and episodic reward setting, we introduce a novel method that is beneficial to improve the data efficiency of RL algorithms and help the agent to reproduce previous trajectories in the memory buffer efficiently. We introduce a new distance measure that calculates the difference between different trajectories and then develop an RL optimization problem with distance constraints by regrading diverse past demonstrations as guidance. Intuitively, POSE can be viewed as a simple method to encourage the agent to revisit the parts of state space where the past trajectories are located. We will introduce how to train the policy in detail in Section 4.2.

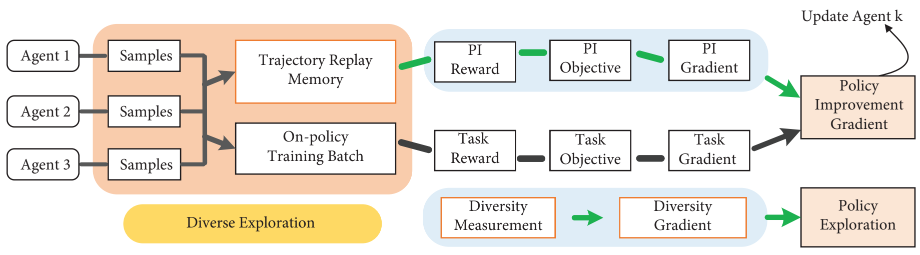


Figure 1: The framework of POSE. The diverse exploration in POSE leverages multiple agents to sample training batches and use the measurement of diversity to encourage agents to collect diverse trajectories, while the traditional RL, e.g., PPO [20] or SAC [45], uses a single agent to collect data. In the meantime, every agent maintains a replay buffer and stores specific trajectories in this buffer. These past good trajectories can be used to guide the agents to revisit the region where the agents can obtain rewards with a higher probability.

4.1.3. Improving Exploration by Generating Diverse Trajectories. Compared with typical distributed RL methods such as A3C [30] and IMPALA [32], our POSE method not only simply employs multiple agents to collect amounts of trajectories independently by interacting with the parallel environments but also uses a diverse exploration mechanism to ensure exploration efficiency. We propose a new diversity metric to drive the different agents on the team to reach diverse regions in the state space and maintain the diversity of an ensemble of agents. In our framework, an agent of the team is impelled to pay more attention to visiting the state underexplored by other agents. Consequently, this helps the agents in the team to explore the environment systematically and avoid being stuck in local optima.

4.2. Policy Improvement with Soft Self-Generated Guidance. We assign a trajectory τ to a domain-dependent behavior characterization function$b(\tau)$ to describe its behavior. For example, in MuJoCo Maze tasks described as the benchmark [47], $b(\tau)$ can be as simple as a two-dimensional vector sequence, and each component of the sequence represents the agent's (x, y) location in every timestep:

$$b(\tau) = b\{(s_0, a_0), \ldots, (s_n, a_n)\} = \{(x_0, y_0), \ldots, (x_n, y_n)\}. \tag{5}$$

The behavior characterization $b(\tau)$ suggests that states or position information, not actions, are used to distinguish different trajectories, and a similar approach is adopted by [48, 49]. If needed, other behavior characterization functions can also be defined to adjust the focus of the distance measurement based on different aspects such as state visit, action choices, or both.

A particular distance between the current trajectory τ and the replay memory $\mathscr{M}$ can be computed as follows:

$$\mathrm{dist}(\tau, \mu) = D_{\mathrm{MMD}}(b(\tau), b(\mu)) = \mathrm{MMD}^2(p, q, \mathscr{H}). \tag{6}$$

Here, $b(\cdot)$ extracts the coordinate information corresponding to each state-action pair of the trajectory. Meanwhile, τ and μ are viewed as two deterministic policies in equation (6), and p and q are the state-action distributions that are induced by deterministic policies τ and μ, respectively. Furthermore, the distance between the current trajectory τ and the trajectory replay memory $\mathscr{M}$ is defined as follows:

$$\mathrm{dist}(\tau, \mathscr{M}) = \min_{\mu\in\mathscr{M}} D_{\mathrm{MMD}}(b(\tau), b(\mu)), \tag{7}$$

i.e., $\mathrm{dist}(\tau, \mathscr{M})$ is the minimum value of distances between τ and each trajectory in the replay memory $\mathscr{M}$.

Existing methods maintaining a memory buffer may overly exploit those good experience data, but these trajectories collected during the training process can be imperfect. The excessive exploitation of the imperfect demonstrations might lead to myopic behaviors and hurt performance in some cases. We choose to update the parameters of the agent by regarding previous good trajectories as guidance rather than directly imitating these imperfect trajectory data. By imposing a distance constrains in the trajectory space, each agent of the team is encouraged to revisit the region where past good trajectories are located. In this way, the agent not only exploits what it already knows to maximize reward but also reduces the overuse of previous good trajectories. Furthermore, our method allows for flexibility in the action choices and enables the agent to smoothly move beyond to find near-optimal policies. This intuition is based on the following observation:

Assumption 1. For any given bounded tolerance factor d, there always exists trajectories with higher returns than the demonstration trajectory, and they stay in a region of radius d around the demenstration trajectory$d$, even when the demonstration trajectory is imperfect and generated by the agent interacting with the environment.

$$\forall\mu \in \mathscr{M}, \exists d \in (0, \infty), \mathrm{dist}(\tau, \mu) \le \mathrm{d}, R(\tau) \ge R(\mu). \tag{8}$$

Based on this distance in the trajectory space, we define a new RL optimization problem with constraints for the i-th agent as follows:

$$\begin{gathered}\theta_{k+1}^{i} = \max_{\theta^i} J\left(\theta_k^i\right),\\ \text{s.t. } \mathrm{dist}\left(\tau_{(j)}, \mathscr{M}^i\right) \le \delta, \forall\tau_{(j)} \in \mathscr{B}^i.\end{gathered} \tag{9}$$

Here, δ is a constant, and $\mathscr{M}^i$ is the replay memory of the i-th agent. $\mathscr{B}^i$ contains the trajectory data collected by the i-th agent at the current epoch, and $\tau_{(j)}$ represents a trajectory in the buffer $\mathscr{B}^i$.

From the perspective of policy optimization, it indicates that using constraints would better fit our problem settings for two reasons:

(1) *Convergence*. The constraint could affect the policy update when there are trajectories that do not satisfy the constraints. In this way, it directs the agent to generate trajectories that stay in the constraint domain defined by the distance on the trajectory space. According to Assumption 1, the more frequently the agent visits the state space around demonstration trajectories, the more likely the agent is to produce trajectories with higher returns.

(2) *Optimality*. The agent's replay buffer containing previous specific trajectories is maintained by a dynamic update mechanism. Once the agent collects better trajectories with higher returns, shorter trajectory length, or shorter distance to the goal, the worst trajectories in the buffer will be replaced. Consequently, compared with the self-imitation learning methods, our method can leverage the imperfect demonstration trajectories for guiding the policy while eliminating their side effects in optimization, thus working better with imperfect demonstration trajectories.

4.2.1. Optimization Process of Soft Self-Imitation RL Objective. This section mainly describes how to efficiently optimize the RL objective with the distance constraint on the trajectory space. For simplification, we have omitted the superscript i for some symbols, and these symbols have the same meaning as above unless otherwise specified.

First, using the Lagrange multiplier method, the optimization problem (9) can be converted into an unconstrained form:

$$L(\theta) = J(\theta) - \sigma \mathbb{E}_{\tau \in \mathscr{B}^i} \mathrm{d}\left(\tau, \mathscr{M}^i\right), \tag{10}$$

where $\sigma > 0$ is a Lagrange multiplier, which is used to determine the effect of the constraint, and $\mathrm{d}(\tau, \mathscr{M}^i) = 0$, if $\mathrm{dist}(\tau, \mathscr{M}^i) \leq \delta$, else $\mathrm{d}(\tau, \mathscr{M}^i) = \mathrm{dist}(\tau, \mathscr{M}^i)$. Then, the gradient with respect to the policy parameters θ of the objective (8) is given by

$$\nabla_\theta L = \nabla_\theta J - \sigma \mathbb{E}_{\tau \in \mathscr{B}^i}\left[\mathrm{d}\left(\tau, \mathscr{M}^i\right) \nabla_\theta \log p_\theta(\tau)\right], \tag{11}$$

where p_θ is a distribution induced by π_θ over the trajectory space, and p_θ represents the probability of the trajectory τ. p_θ can be expressed in terms of the environment dynamics model and the policy of the agent, i.e., $p_\theta(\tau) = \rho(s_0)\sum_{t=0}^{T}\pi_\theta(a_t|s_t)p(s_{t+1}|s_t, a_t)$. Therefore, the gradient of the score function of the trajectory distribution has the expression $\nabla_\theta \log p_\theta(\tau) = \sum_{t=0}^{T}\log\pi_\theta(a_t|s_t)$, and it does not contain the environment dynamic model.

4.3. Policy Exploration. As described in Section 4.2, if an agent has collected specific trajectories when interacting with the environment and stores them in the trajectory buffer, our method will regard these trajectories as guidance in the policy improvement step and direct the agent to revisit the regions where past good trajectories are located gradually. However, it might cause the agent to get stuck in local optima. To achieve better exploration performance in challenging tasks with sparse and deceptive rewards, we employ a group of heterogeneous agents to interact with the environment simultaneously. We hope to enable different agents on the team to reach diverse regions of the state space and drive deep exploration.

To maintain the diversity of the team, we first introduce a novel measurement of diversity. Considering the continuous control tasks that we focus on, we use the mean MMD distance of the different agents in the team as the diversity measurement. The mean MMD distance is computed with current trajectories collected by the team and their mean trajectories. The mean trajectory is generated by performing the mean of Gaussian action distribution of the agent at each timestep. Specifically, let $\overline{\tau}^i$ denote the mean trajectories of agent i, and let Π denote the ensemble of agents we employ, and let $\mathcal{S}$ be the set of mean trajectories of all agents. The diversity measurement of an agent is calculated as

$$\mathrm{div}(\Pi) = \frac{1}{|\Pi|}\sum_{i=1}^{|\Pi|} D_{\mathrm{MMD}}\left(\pi^i, \Pi \backslash \pi^i\right), \tag{12}$$

where $D_{\mathrm{MMD}}(\pi^i, \Pi\backslash\pi^i)$is defined as

$$D_{\mathrm{MMD}}\left(\pi^i, \Pi\backslash\pi^i\right) = \min_{\overline{\tau}^j \in \mathcal{S}} \mathbb{E}_{\tau \in \mathscr{B}^i}\left[\mathrm{MMD}\left(b(\tau), b\left(\overline{\tau}^j\right)\right)\right]. \tag{13}$$

For discrete control tasks, we utilize the current optimal trajectories to compute the diversity measurement of agent team T, and the optimal trajectory are produced by the agent through performingthe action with the highest Qvalueat each timestep. Finally, the objective function of policy exploration is given as

$$\begin{aligned} \theta_{k+1} &= \max_\theta \ \mathrm{div}\left(\Pi_{\theta_{k+1/2}}\right), \\ \text{s.t. } D_{\mathrm{KL}}&\left(\pi^i_{k+1}, \pi^i_{k+(1/2)}\right) \leq \delta, i \in 1, \ldots, |\Pi|. \end{aligned} \tag{14}$$

Intuitively, the policy exploration step prevents the team of agent from being stuck in the same local optimum by driving different agents to reach diverse regions of the state space. This is achieved by finding an ensemble of policies that maximizes the diversity measurement defined in equation (12). In the meantime, we require that every new policy π_{k+1} after the policy update lies inside the trust region around the old policy $\pi_{k_{+1/2}}$, which is defined as $\{\pi\colon D_{\mathrm{KL}}(\pi, \pi_{k+1/2}) \leq \delta\}$, and hence it can avoid suffering severe performance degradation.

4.3.1. Optimization Process for the Diversity RL Objective. To solve the constrained optimization problem, we propose to approximately solve it linearizing around current policies $\pi^i_k, i = 1, \ldots, |\Pi|$. The gradient of diversity measurement in equation (14) with respect to the policy parameters can be easily calculated with the mean trajectories and current rollouts. Denoting the gradient of diversity measurement as g and the Hessian matrix of the KL-divergence for agent i as H^i, the linear approximation to equation (14) is as follows:

$$\begin{aligned}\theta_{k+1} &= \max_{\theta}\ g^T(\theta-\theta_k),\\ \text{s.t. } &\frac{1}{2}(\theta-\theta_k)^T H_k^i(\theta-\theta_k)\le\delta, i\in 1,\ldots,|\Pi|.\end{aligned}\tag{15}$$

Then, we adopt the conjugate gradient method [4] to approximately compute the inverse of H_k^i and the gradient direction. However, the trust-region optimization can incur slow update of parameters and thus reduce the sample efficiency and improve the computational cost. In the beginning phase of the training process, we can use a first-order optimization method like [20] to solve the optimization problem in equation (14).

5. Evaluation of Results

In this section, we present our experimental results and compare our method's performance with other baseline methods. In Section 5.1, we present an overview of experiment setups we used to evaluate our methods. Sections 5.2~5.3 report the results in different experiment environments, respectively.

5.1. Experimental Setup

5.1.1. Environments

(1) Grid World. To illustrate our method's effectiveness, we design a huge 2D grid world based on Gym [50] with two different settings: sparse rewards and deceptive rewards. At each timestep, the agent observes its coordinates relative to the starting point and chooses from four possible actions: move east, move south, move west, and move north. At the start of each episode, the agent starts from the bottom-left corner of the map, and an episode terminates immediately once a reward is collected. In the sparse reward setting, shown in Figure 2(a), there is only a single goal located at the top-right corner, and the agent will be rewarded with 6 when reaching this goal. On the other hand, in the deceptive reward setting depicted in Figure 2(b), there are a misleading goal with a reward of 2,in the upper left room of the grid world that can lead to local optima.

(2) MuJoCo. For continuous control tasks, we also conduct different experiments in environments based on the MuJoCo physical engine [51]. Two continuous robotic control tasks, ant and swimmer mazes, are selected to evaluate the performance of our methodology and baseline methods. In each task, the agent takes a vector of physical state containing the agent's joint angles and task-specific attributes as the input of the policy. Examples of task-specific attributes include goals walls and sensor readings. Then, the control policy generates a vector of action values which the agent performs in the environment. We compare our method to previous algorithms in the following tasks:

(i) Swimmer maze: the swimmer is rewarded for reaching the goal positions in the maze shown in Figure 3(a). The agent can obtain suboptimal rewards (+200) by arriving at the leftmost goal, and the agent is rewarded by 500 when it reaches the optimal goal in the rightmost of the maze of maze.

(ii) Ant maze: the ant is rewarded for arriving at the specified positions in the maze, as shown in Figure 3(b). The ant can collect the small rewards when reaching the goal below the maze and maximize the reward if it reaches the goal up the maze.

5.1.2. Baseline Methods. The baseline methods used for performance comparison vary in different tasks. For discrete and continuous control tasks, we compare our algorithm with the following baseline methods: (1) PPO [20], (2) SAC [45], (3) DPPO: distributed PPO [52], (4) Div + A2C: A2C with a distance measure regularization [27], (5) PPO + SIL: PPO with self-imitation learning [9], and (6) PPO + EXP: PPO with count-based exploration bonus augmented reward function: $r(s,a)+(\lambda/\sqrt{N_e})$, where N_e is the number of times the state cluster under the state representation e is visited during training, and λ is the hyperparameter that controls the weight of the exploration bonus term. For each baseline, we adopt the parameters that produce the best performance during the parameter search, and not all baselines are adopted in each task.

5.2. Performance in Huge Grid World. In this experiment, we evaluate different methods in 2D mazes with different reward settings: sparse rewards and deceptive rewards. We consider five baseline methods in these experiments: A2C, PPO, PPO + SIL, PPO + EXP, and Div-A2C. The performances of each method are reported in terms of average return and success rate learning curves. All learning curves are averaged over 8 runs.

5.2.1. Sparse Reward. Figure 2(a) shows the sparse reward grid world. In this maze environment with the discrete state-action space, we implement our method based on the PPO [20] algorithm. In this experiment, we select A2C, vanilla PPO, PPO + SIL, Div-A2C, as well as PPO + EXP as the baselines. For each method, we adopt the settings that produce the highest performance during the hyperparameter search. The learning curves are presented in Figure 4. Compared with other baseline methods, we notice that our method is able to learn the optimal policy at a higher learning rate and achieve better performance evaluations in terms of average return and success rate in this task. Our method can encourage the agent to visit the underexplored regions of the state space and make better use of past good trajectories maintained in the memory buffers. Therefore, our method can prompt the agent to focus on the state space, from which the agent can collect the sparse rewards with

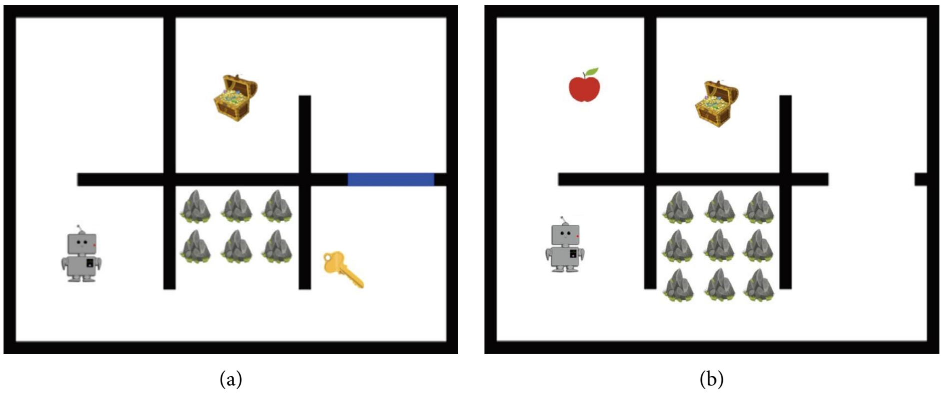


Figure 2: A collection of environments with discrete state-action spaces that we use. (a) Huge grid world with sparse rewards: key-door-treasure domain. The agent should pick up the key (+2) in the right-down room in order to open the blue door (+4) and collect the treasure (+4) in the middle-up room to maximize the reward. (b) Huge grid world with deceptive rewards. There is an apple in the left-up room that gives small rewards (+2) and a treasure in the middle-up room, which generates higher rewards (+10).

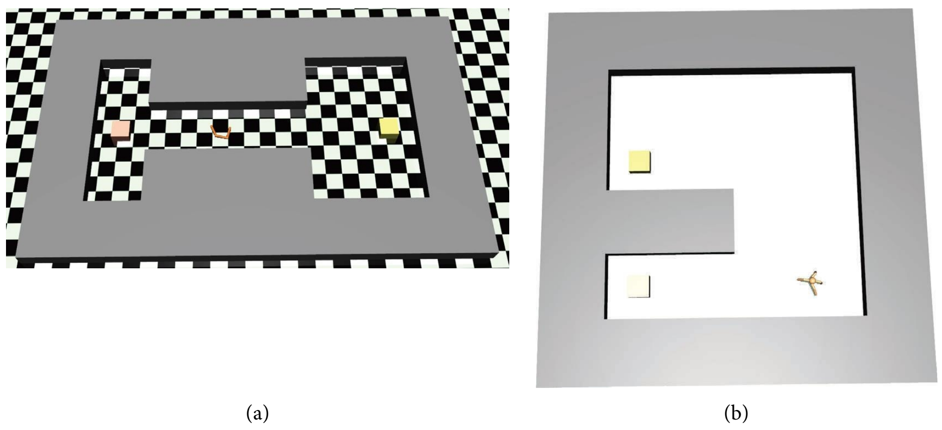


Figure 3: A collection of environments with continuous state-action spaces that we use. (a) Swimmer in the maze. (b) Ant in the maze.

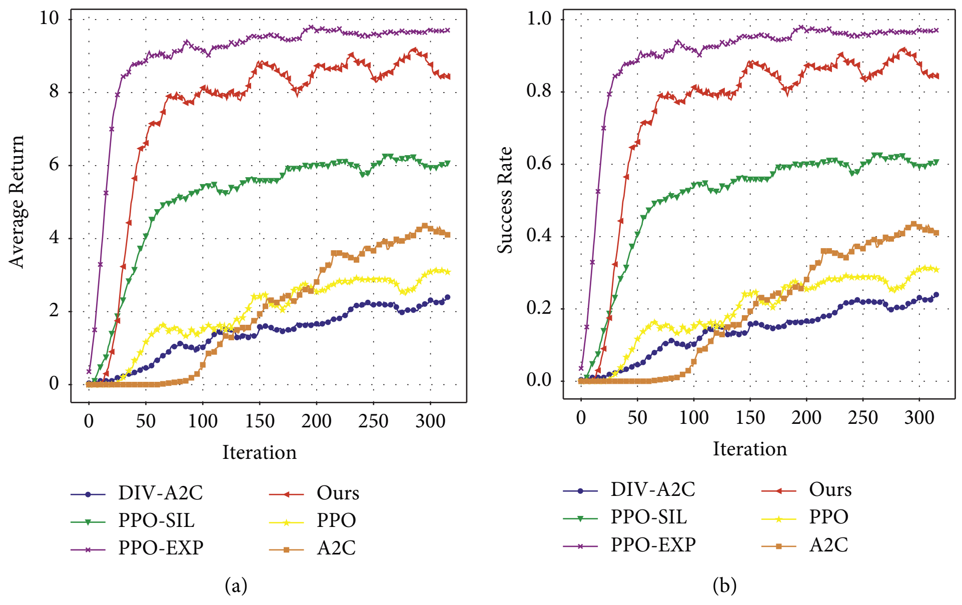


Figure 4: Learning curves of average return and success rate in the huge gridworld with sparse rewards. Specially, the success rate is used to illustrate the frequency at which agents reach the globally optimal goal during training process.

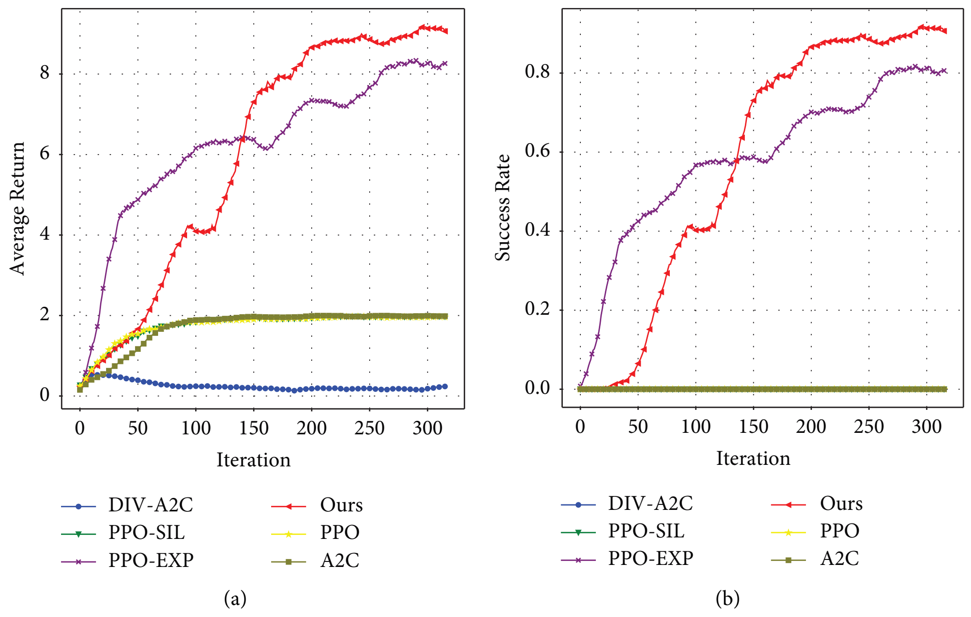


Figure 5: Learning curves of average return and success rate in the huge grid world with deceptive rewards. Specifically, the success rate is used to illustrate the frequency at which agents reach the globally optimal goal during the training process..

a higher probability. The PPO and A2C agents do not have any specially designed mechanisms for exploration; hence, they only arrive at the treasure and obtain the sparse rewards with a very low probability. These agents are not able to learn the optimal policy leading to the treasure due to the low percentage of valid samples. In the training process, the PPO + EXP agent can explore the environment better and occasionally collect treasure to achieve the best episode return. We also note that it is rare and difficult for the Div-A2C method to encounter the sparse rewards in one short burst. While the PPO + SIL agent can utilize past good trajectories, the success rates and average returns of this method are lower than those of our method at the end of the training process.

5.2.2. Deceptive Reward.

Figure 2(b) illustrates the deceptive grid world. From Figure 5, we notice that the average rewards of the baseline methods have a slight increase compared with those in the sparse reward settings. These methods, except for PPO + EXP, can only achieve suboptimal rewards by collecting the apples, and hence, the agents are stuck with the suboptimal policy due to deceptive rewards. In contrast, our method not only adopts myopic and suboptimal behaviors but also treats the past good trajectories as guidance and allows the agent to reach diverse regions in the state space by improving upon past trajectories to generate good new trajectories. Although PPO + EXP can reach the treasure and obtain the highest rewards, its learning process has greater instability, leading to inferior performance.

As shown in Figure 6, we also plot the state-visitation counts for all methodsin the maze with deceptive rewards, which explicitly illustrate how different agents explore the 2D gridworld environments. From the state-visitation count graph, it can be seen that the four baseline approaches are either prone to falling into a local optimum or they cannot explore the environment sufficiently to visit the goal with larger reward. PPO + EXP can obtain optimal rewards from the treasure, but it requires considerable computation to visit the meaningless region of the state-action space due to intrinsic rewards. However, it can be observed from Figure 6 that our POSE method is able to escape from the area of deceptive rewards, explore a wider and farther region of the 2D grid world significantly, and arrive at the goal with the optimal reward successfully without spending too much computation visiting the insignificant region of the state space.

5.3. Performance Comparison in MuJoCo Environments.

We evaluate our method on continuous control tasks shown in Figure 3 with sparse and deceptive rewards based on the MuJoCo physical engine and similarly plot the in-training median scores in Figures 7 and 8. We consider four baseline methods in these experiments: SAC, PPO, PPO + SIL, and DPPO. The performance of each method is reported in terms of average return and success rate learning curves. Similar to the experiments in the discrete maze, all learning curves are averaged over 5 runs.

Compared with baseline methods, it is observed that our method can learn considerably faster and obtains higher average returns and success rates. The average returns and success rates of POSE and other baseline methods in the swimmer maze are usually higher than those in the ant maze because the dimensions of the state and action space of the swimmer are lower than those of the ant, and the swimmer will not trip over due to incorrect action inputs. POSE achieves a success rate of almost 100% in less than 200 epochs.

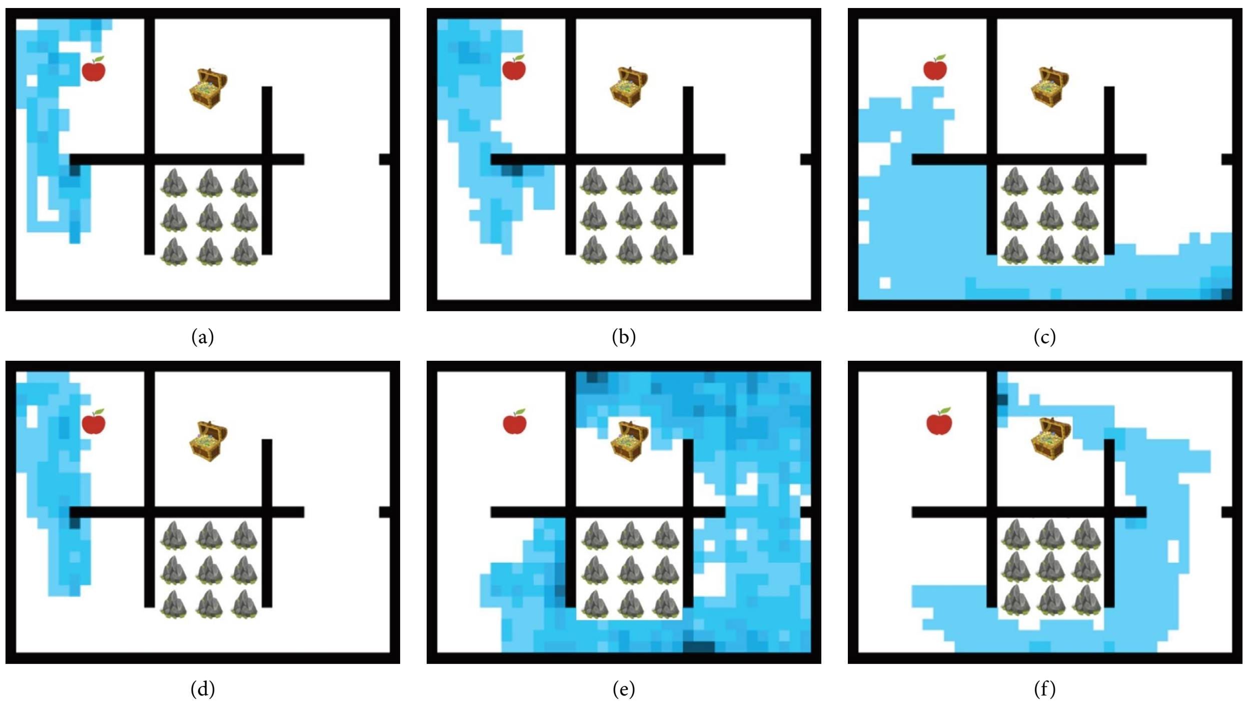


Figure 6: State-visitation counts of the grid world for different algorithms: (a) PPO, (b) A2C, (c) Div-A2C, (d) PPO-SIL, (e) PPO-EXP, and (f) ours. PPO, A2C, and PPO-SIL are easily trapped into local optimum. Div-A2C could visit different regions where the apple does not locate, but it cannot arrive at the treasure and obtain the optimal reward. PPO + EXP enables the agent to reach the treasure and obtain the highest rewards; however, it spends amounts of computation to visit parts of state-action space where it cannot obtain any useful rewards. Our algorithm can explore the state-action space systematically and collect the optimal rewards quickly, which enables the agent to learn the optimal policy at a high learning rate..

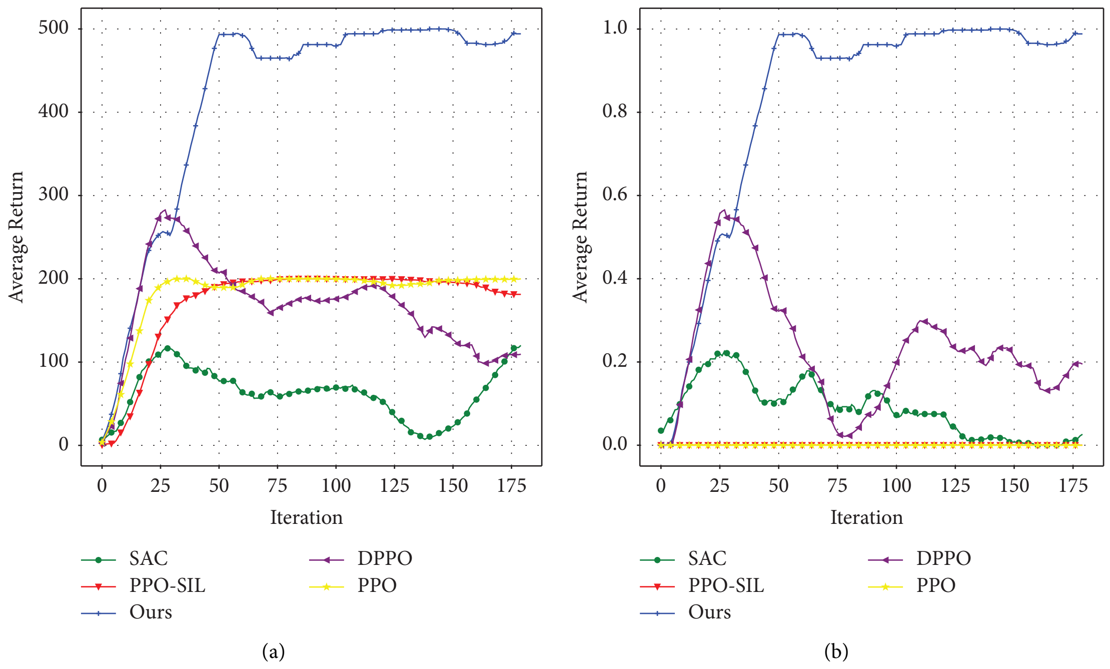
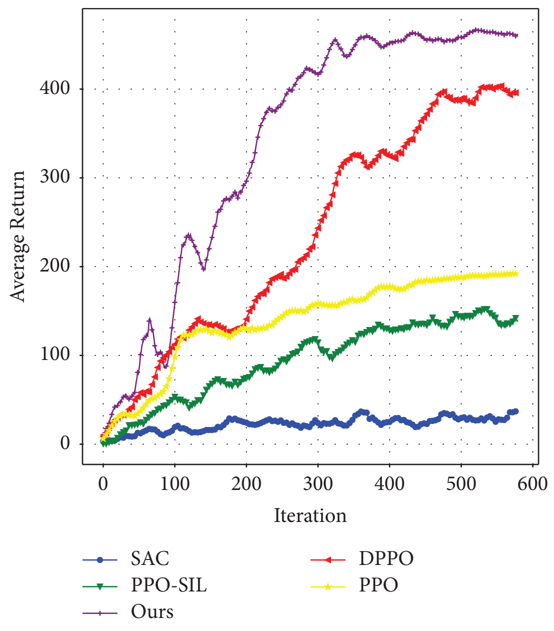
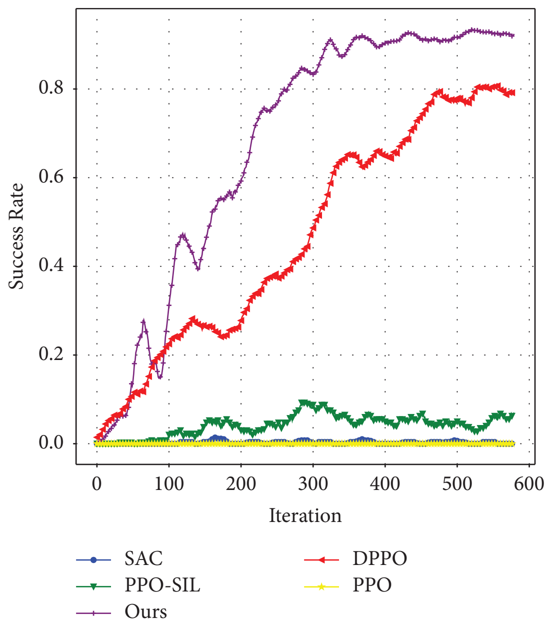


Figure 7: Learning curves of average return and success rate in swimmer maze. Specially, the success rate is used to illustrate the frequency at which agents reach the globally optimal goal during training process.



(a)



(b)

FIGURE 8: Learning curves of average return and success rate in ant maze. Specially, the success rate is used to illustrate the frequency at which agents reach the globally optimal goal during the training process.

```
Input: number of agents $N$, learning rate $\alpha$, and on-policy training buffer $\mathscr{R}_i$ for each agent, highly rewarded trajectory buffer $\mathscr{D}_i$
for each agent, sequence length $m$, and number of epochs $M$.
(1) Initialize policy weights $\theta$ of each agent.
(2) Initialize the prior good trajectory buffer $\mathscr{M}_\mu$.
(3) for $i = 0$ to $M$ do
(4)     Collect rollouts and store them in their own on-policy training $\mathscr{R}_i$ for each agent.
(5)     Update the soft self-imitation training batches $\mathscr{D}_i$ for each agent.
(6)     Compute advantage estimates $\widehat{A}_i$, $i = \{1, 2, \ldots, N\}$.
(7)     Estimate the distance between current trajectories $\tau_i$ and highly rewarded trajectories in soft self-imitation replay buffer $\mathscr{D}_i$ for
        each agent.
(8)     Estimate the gradient $\nabla_\theta \widetilde{L}_i$ in equation (11), $i = \{1, 2, \ldots, N\}$.
(9)     Perform policy improvement step: $\theta_{i+1/2} \longleftarrow \theta_i + \alpha \nabla_\theta \widetilde{L}_i$ for each agent.
(10)    Estimate $g$ and $H_k^i$.
(11)    Perform policy exploration step by update policy parameters according to equation (15).
(12) end for
```

ALGORITHM 1: Policy optimization with soft self-generated guidance and diverse exploration (POSE).

While PPO and PPO + SIL often adopt myopic behaviors and converge to suboptimal policies, POSE is able to eliminate the local optimum and find better strategies to obtain larger episode returns. We also compare our algorithm with the state-of-the-art RL methods, SAC and DPPO. SAC is based on the maximum entropy RL framework, which trains a policy by maximizing the trade-off between expected return and entropy. As a result, SAC cannot achieve a significantly better performance of exploration than PPO in our experiments. It rarely encounters the optimal rewards received from the treasure and occasionally gathers trajectories leading to the treasure in the swimmer maze. Thus, this off-policy method might forget the past good experience and fail to learn the optimal policy to achieve the best reward. DPPO can learn the policy leading to the optimal reward, but this method has a lower learning rate and success rate. In contrast, our POSE method can also successfully generate trajectories that visit novel states and save the trajectories with high returns in the buffer. The POSE agent is able to replicate the past good experience by using the highly rewarded trajectories as guidance and learning the optimal policies with the highest learning rate and success rate.

6. Conclusion

In this study, we study the problem of how to design a practical RL algorithm for tasks where only sparse and deceptive feedback is provided. We propose a novel two-step policy optimization framework called POSE, which exploits diverse imperfect demonstrations for faster and more efficient online RL. By regarding diverse past trajectories as soft guidance, the agent can reproduce the trajectories easily and smoothly move beyond them to find near-optimal policies, which regulates the direction of policy improvement and accelerates the learning speed. Furthermore, a novel diversity measure is introduced to drive each agent of the team to visit the different underexplored regions of the state space and achieve deep exploration. Experimental results on physical control benchmarks demonstrate the effectiveness of our approach over other baseline methods in terms of the efficient exploration and avoidance of local minima in tasks with long horizons and sparse or deceptive rewards.

Appendix

Algorithm training process.

Algorithm 1 describes our method in detail. At each iteration, the algorithm is executed according to the framework shown in Figure 1, and the experiences generated by each agent in the environment are stored in their respective on-policy training batches. Then, we use the on-policy trajectory data to update the soft self-imitation learning batches $\mathscr{M}$ according to Section 4.1. Furthermore, we compute the advantage of the current policy and the distance between the current trajectory τ and the soft self-imitation replay buffer $\mathscr{D}_i$ for each agent, and we utilize them to estimate the gradient of $\nabla_\theta \widetilde{L}$ in equation (11). Finally, we update the parameters of π_θ with the gradient ascent algorithm and adapt the penalty factor according to the MMD distance.

Data Availability

The datasets generated during and/or analysed during the current study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Authors' Contributions

Guojian Wang: Conceptualization, Methodology, Software, Formal analysis, Writing–Original Draft. Faguo Wu: Writing–Review and Editing, Supervision. Xiao Zhang: Validation, Supervision, Funding acquisition. Jianxiang Liu: Formal analysis, Software, Visualization.

Acknowledgments

This work was supported by the National Key R&D Program of China (grant no. 2022ZD0116401).

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] D. Silver, A. Huang, C. J. Maddison et al., "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[3] T. Lillicrap, J. J. Hunt, A. Pritzel et al., "Continuous control with deep reinforcement learning," 2016, https://arxiv.org/abs/1509.02971.

[4] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proceedings of the International Conference on Machine Learning*, pp. 1889–1897, Lille, France, July 2015.

[5] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proceedings of the International Conference on Machine Learning*, pp. 1587–1596, Stockholm, Sweden, January 2018.

[6] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6292–6299, IEEE, Brisbane, Australia, May 2018.

[7] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel, "Vime: variational information maximizing exploration," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[8] C. Florensa, Y. Duan, and P. Abbeel, "Stochastic neural networks for hierarchical reinforcement learning," in *Proceedings of the International Conference on Learning Representations*, Sydney, Australia, August 2017.

[9] J. Oh, Y. Guo, S. Singh, and H. Lee, "Self-imitation learning," in *Proceedings of the International Conference on Machine Learning*, pp. 3878–3887, PMLR, Xi'an, China, July 2018.

[10] Y. Guo, J. Choi, M. Moczulski et al., "Memory based trajectory-conditioned policies for learning from sparse rewards," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[11] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015, https://arxiv.org/abs/1511.05952.

[12] T. Gangwani, Q. Liu, and J. Peng, "Learning self-imitating diverse policies," in *Proceedings of the International Conference on Learning Representations*, New Orleans, LO, USA, May 2019.

[13] J. Oh, Y. Guo, S. Singh, and H. Lee, "Generative adversarial self-imitation learning," in *Proceedings of the International Conference on Learning Representations*, Montreal, Canada, May 2019.

[14] C. Liang, M. Norouzi, J. Berant, Q. V. Le, and N. Lao, "Memory augmented policy optimization for program synthesis and semantic parsing," in *Proceedings of the Advances in Neural Information Processing Systems*, Montréal, Canada, December 2018.

[15] Z. Lin, T. Zhao, G. Yang, and L. Zhang, "Episodic memory deep q-networks," 2018, https://arxiv.org/abs/1805.07603.

[16] C. Blundell, B. Uria, A. Pritzel et al., "Model-free episodic control," 2016, https://arxiv.org/abs/1606.04460.

[17] A. Pritzel, B. Uria, S. Srinivasan et al., "Neural episodic control," in *Proceedings of the International Conference on Machine Learning*, pp. 2827–2836, PMLR, Sydney, Australia, May 2017.

[18] Z. Peng, H. Sun, and B. Zhou, "Non-local policy optimization via diversity-regularized collaborative exploration," 2020, https://arxiv.org/abs/2006.07781.

[19] Y. Zhang, W. Yu, and G. Turk, "Learning novel policies for tasks," in *Proceedings of the International Conference on Machine Learning*, pp. 7483–7492, PMLR, Long Beach, CA, USA, January 2019.

[20] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, https://arxiv.org/abs/1707.06347.

[21] M. Plappert, R. Houthooft, P. Dhariwal et al., "Parameter space noise for exploration," in *Proceedings of the International Conference on Learning Representations*, Xi'an, China, August 2018.

[22] M. Fortunato, M. G. Azar, B. Piot et al., "Noisy networks for exploration," in *Proceedings of the International Conference on Learning Representations*, Vancouver, Canada, March 2018.

[23] I. Osband, B. Van Roy, D. J. Russo, and Z. Wen, "Deep exploration via randomized value functions," *Journal of Machine Learning Research*, vol. 20, no. 124, pp. 1–62, 2019.

[24] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *Proceedings of the International Conference on Machine Learning*, pp. 2778–2787, PMLR, Sydney, Australia, May 2017.

[25] J. Achiam and S. Sastry, "Surprise-based intrinsic motivation for deep reinforcement learning," 2017, https://arxiv.org/abs/1703.01732.

[26] N. Savinov, A. Raichuk, D. Vincent et al., "Episodic curiosity through reachability," in *Proceedings of the International Conference on Learning Representations*, Sydney, Australia, December 2019.

[27] Z.-W. Hong, T.-Y. Shann, S.-Y. Su, Y.-H. Chang, T.-J. Fu, and C.-Y. Lee, "Diversity-driven exploration strategy for deep reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[28] S. Han and Y. Sung, "Diversity actor-critic: sample-aware entropy regularization for sample-efficient exploration," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021*, pp. 4018–4029, PMLR, Sydney, Australia, August 2021.

[29] M. A. Masood and F. Doshi-Velez, "Diversity-inducing policy gradient: using maximum mean discrepancy to find a set of diverse policies," in *Proceedings of the International Joint Conferences on Artificial Intelligence Organization*, Vancouver, Canada, December 2019.

[30] V. Mnih, A. P. Badia, M. Mirza et al., "Asynchronous methods for deep reinforcement learning," in *Proceedings of the International Conference on Machine Learning*, pp. 1928–1937, PMLR, New York, NY, USA, June 2016.

[31] S. Schmitt, M. Hessel, and K. Simonyan, "Off-policyactor-critic with shared experience replay," in *Proceedings of the International Conference on Machine Learning*, pp. 8545–8554, PMLR, Vienna, Austria, May 2020.

[32] L. Espeholt, H. Soyer, R. Munos et al., "Impala: scalable distributed deep-rl with importance weighted actor-learner architectures," in *Proceedings of the International Conference on Machine Learning*, PMLR, Stockholm, Sweden, June 2018.

[33] T. Hester, M. Vecerík, O. Pietquin et al., "Deep q-learning from demonstrations," 2018, https://arxiv.org/abs/1704.03732.

[34] A. P. Badia, P. Sprechmann, A. Vitvitskyi et al., "Never give up: learning directed exploration strategies," in *Proceedings of the International Conference on Learning Representations*, Xi'an, China, September 2020.

[35] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 627–635, Fort Lauderdale, FL, USA, January 2011.

[36] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," *Icml*, vol. 1, p. 2, 2000.

[37] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," *Aaai*, vol. 8, pp. 1433–1438, 2008.

[38] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[39] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in Neural Information Processing Systems*, vol. 12, 1999.

[40] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. Smola, "A kernel method for the two-sample-problem," *Advances in Neural Information Processing Systems*, vol. 19, pp. 513–520, 2006.

[41] A. Gretton, D. Sejdinovic, H. Strathmann et al., "Optimal kernel choice for large-scaletwo-sample tests," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 1205–1213, Citeseer, Lake Tahoe, Nevada, December 2012.

[42] P. Thomas, B. C. Silva, C. Dann, and E. Brunskill, "Energetic natural gradient descent," in *Proceedings of the International Conference on Machine Learning*, pp. 2887–2895, PMLR, New York, NY, USA, June 2016.

[43] G. K. Dziugaite, D. M. Roy, and Z. Ghahramani, "Training generative neural networks via maximum mean discrepancy optimization," in *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, pp. 258–267, Amsterdam, Netherlands, July 2015.

[44] R. Fortet and E. Mourier, "Convergence de la répartition empirique vers la répartition théorique," *Annales Scientifiques de l'Ecole Normale Superieure*, vol. 70, no. 3, pp. 267–285, 1953.

[45] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the International Conference on Machine Learning*, pp. 1861–1870, PMLR, Stockholm, Sweden, January 2018.

[46] R. Lowe, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agentactor-critic for mixed cooperative-competitive environments," in *Proceedings of the Advances in Neural Information Processing Systems*, Long Beach, CA, USA, June 2017.

[47] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *Proceedings of the International Conference on Machine Learning*, pp. 1329–1338, PMLR, New York, NY, USA, May 2016.

[48] E. Conti, V. Madhavan, F. P. Such, J. Lehman, K. O. Stanley, and J. Clune, "Improving exploration in evolution strategies

for deep reinforcement learning via a population of novelty-seeking agents," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 5032–5043, Montr, Canada, December 2018.

[49] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine, "Diversity is all you need: learning skills without a reward function," in *Proceedings of theInternational Conference on Learning Representations*, New York, NY, USA, December 2018.

[50] G. Brockman, V. Cheung, L. Pettersson et al., *Open AI Gym*, Open AI, San Francisco, CA, USA, 2016.

[51] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: a physics engine for model-based control," in *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, IEEE, Vilamoura-Algarve, Portugal, October 2012.

[52] N. Heess, D. Tb, S. Sriram et al., "Emergence of locomotion behaviours in rich environments," 2017, https://arxiv.org/abs/1707.02286.