Reward Shaping for Reinforcement Learning with An Assistant Reward Agent

Haozhe Ma [1] Kuankuan Sima [2] Thanh Vinh Vo [1] Di Fu [1] Tze-Yun Leong [1]

Abstract

Reward shaping is a promising approach to tackle the sparse-reward challenge of reinforcement learning by reconstructing more informative and dense rewards. This paper introduces a novel dual-agent reward shaping framework, composed of two synergistic agents: a policy agent to learn the optimal behavior and a reward agent to generate auxiliary reward signals. The proposed method operates as a self-learning approach, without reliance on expert knowledge or hand-crafted functions. By restructuring the rewards to capture future-oriented information, our framework effectively enhances the sample efficiency and convergence stability. Furthermore, the auxiliary reward signals facilitate the exploration of the environment in the early stage and the exploitation of the policy agent in the late stage, achieving a self-adaptive balance. We evaluate our framework on continuous control tasks with sparse and delayed rewards, demonstrating its robustness and superiority over existing methods.

1. Introduction

Model-free reinforcement learning (RL) has achieved notable advancements across various domains. However, challenges persist in environments with sparse and delayed rewards, especially in long-horizon tasks like robotic control and video games. The lack of immediate or rich feedback hinders the agent's ability to discriminate the values of different actions. Effective exploration strategies become crucial in this context, as they determine whether an agent can promptly find a beneficial trajectory. Efficient early-stage exploration helps the agent gather numerous positive samples, minimizing sampling costs and hastening learning progression (Ladosz et al., 2022). Additionally, the distribution of sparse rewards, typically characterized by a binary step function, poses difficulties for gradient-based learning methods. These issues highlight the promise of developing auxiliary informative rewards, potentially derived from external expertise or the agent's own accumulated experiences, while ensuring the agent retains its original optimal policy (Singh et al., 2010; Sorg et al., 2010b;a).

Reward shaping (RS) has proven to be effective in addressing the sparse-reward challenge. It supplements environmental rewards with detailed, dense signals, thereby (a) providing fine-grained, meaningful, and informative insights, (b) offering immediate feedback without waiting for the end of long trajectories, and (c) typically presenting the most direct objectives for agents at each interaction with the environment (Gupta et al., 2022; Ng et al., 1999). However, RS methods face certain limitations. One branch uses manually crafted reward functions or learns from expert demonstrations (Bıyık et al., 2022; Cheng et al., 2021; Wu et al., 2021a), but this relies heavily on prior knowledge, which is often limited and expensive to acquire, and may introduce human biases that could degrade agent performance. Alternatively, another branch seeks to autonomously learn intrinsic rewards by uncovering hidden features, such as incorporating exploration bonuses (Devidze et al., 2022; Ostrovski et al., 2017; Tang et al., 2017), driving agent by curiosity (Pathak et al., 2017), or rewarding for novel states (Burda et al., 2018). These strategies often set additional criteria to motivate the agent to simultaneously optimize extra goals. Without careful balance, agents risk becoming trapped in local optima, deviating from the original optimal policy.

To address the challenge of sparse and delayed rewards and the limitations of current methods, we propose a novel framework, **Re**inforcement **L**earning with an **A**ssistant **R**eward **A**gent (ReLara). This framework automatically learns a dense assistant reward function without requiring external human knowledge or introducing additional objectives. It also effectively maintains a self-adaptive balance between exploration and exploitation, as well as between intrinsic and extrinsic rewards, thereby ensuring the consistency of the learned optimal policies and avoiding the risk of local optima. ReLara consists of two agents that cooperate with each other: a *Reward Agent* and a *Policy Agent*. The

[1]School of Computing, National University of Singapore, Singapore [2]College of Design and Engineering, National University of Singapore, Singapore. Correspondence to: Haozhe Ma <haozhe.ma@u.nus.edu>, Tze-Yun Leong <leongty@nus.edu.sg>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

Reward Agent learns a dense reward function to reshape the environmental reward signals. The Policy Agent learns the optimal control policy to be actually applied by maximizing the cumulative reshaped rewards. The two agents jointly interact with the environment, but each of them presents a self-contained RL problem, which can be flexibly solved by independent RL algorithms, making ReLara easy to implement. We summarize the main contributions as follows:

- We propose ReLara[1], a self-learning reward-shaping framework that requires no domain knowledge or human intervention. By considering the future-oriented and trajectory-contextual values, as well as transforming sparse rewards into gradient-friendly denser rewards, ReLara improves data efficiency, learning stability, and convergence speed while maintaining the consistency of the optimal policy.
- ReLara introduces a new self-adaptive approach to balancing exploration and exploitation. Initially, the reward agent provides random rewards, enriching the diversity of directions for the policy agent to optimize and encouraging exploration. As the reward function evolves, it subsequently provides more informative signals to guide the policy agent towards better exploitation.
- We evaluate ReLara on various continuous control tasks in the *MuJoCo* (Todorov et al., 2012), arm robot (de Lazcano et al., 2023) and physical control (Towers et al., 2023) domains, where the extrinsic rewards are sparse binary signals indicating only the task completion. We compare ReLara with state-of-the-art baselines and demonstrate its superior performance and robustness.

2. Preliminaries

Markov Decision Process (MDP) mathematically models the interaction between an agent and the environment in sequential decision-making under uncertainty. An MDP is represented as $\langle S, A, T, R, \gamma \rangle$, where S is the state space; A is the action space; $T : S \times A \times S \to [0, 1]$ is the transition function, specifying the probability of transitioning from one state to another given an action; $R : S \times A \to \mathbb{R}$ is the reward function indicating the logic to give environmental rewards; $\gamma \in [0, 1]$ is the discount factor to weight the importance of future rewards. A stochastic policy $\pi : S \times A \to [0, 1]$ is a probability distribution over state-action pairs, where $\pi(a|s)$ is the probability of taking action $a \in A$ in state $s \in S$. The objective of RL algorithms is to find an optimal policy π^* that maximizes the expected cumulative rewards over time, i.e., $\pi^* = \arg\max_\pi \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)|\pi]$, where s_t and a_t are the state and action at time step t, respectively.

[1]The source code is accessible at: `https://github.com/mahaozhe/ReLara`

Soft Actor-Critic (SAC) is a state-of-the-art model-free RL algorithm that combines the advantages of actor-critic architecture, off-policy learning, and maximum entropy (Haarnoja et al., 2018a;b). The actor-critic method involves separate policy and value networks, which are updated by alternating between *policy evaluation* and *policy improvement* (Sutton & Barto, 2018). The off-policy approach allows the agent to learn from past experiences, improving sample efficiency; the introduced maximum entropy term encourages the policy to explore more diverse actions. The objective function of the policy network is defined as

$$J(\pi) = \sum_{t=0}^{\infty} \mathbb{E}_{a_t \sim \pi(\cdot|s_t)} \left[\gamma^t R(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t)) \right],$$

where $\mathcal{H}(\pi(\cdot|s_t))$ is the entropy of the policy distribution $\pi(\cdot|s_t)$, and α is a temperature parameter that controls the relative importance of the entropy term.

3. Methodology

Our proposed ReLara framework integrates reinforcement learning agents with an assistant reward agent. Figure 1 illustrates the overall framework, where two agents collaborate to accomplish a given task. In ReLara, the conventional agent in RL is denoted as the policy agent $\mathcal{A}_P$, which learns a policy to be actually applied. We introduce an additional reward agent $\mathcal{A}_R$, whose goal is to generate a dense reward signal to enhance the initial sparse reward. The shaped rewards are added to the original environmental rewards adjusted by a weight factor $\beta \in (0, 1]$, forming the final augmented rewards for the policy agent.

3.1. Assistant Reward Agent

The objective of the reward agent $\mathcal{A}_R$ is to generate a shaped reward signal that assists the policy agent $\mathcal{A}_P$. We define a *reward space* $\mathcal{R}$ that constrains the generated reward to a range of real numbers, $\mathcal{R} = [R_{min}, R_{max}] \subseteq \mathbb{R}$. The state space and the action space of the environment $\mathcal{E}$ are denoted as $S^{\mathcal{E}}$ and $A^{\mathcal{E}}$, respectively. In this case, the reward agent $\mathcal{A}_R$ learns a policy to generate a *suggested reward* given the current state s_t from the environment and the action a_t from the policy agent: $\pi^{\mathcal{A}_R} : S^{\mathcal{E}} \times A^{\mathcal{E}} \to \mathcal{R}$. The suggested reward is denoted as $r_t^{\mathcal{S}}$, which will be provided to the policy agent $\mathcal{A}_P$ as an additional reward alongside the environmental reward, denoted as $r_t^{\mathcal{E}}$. We also explore a variant where the reward agent's policy $\pi^{\mathcal{A}_R} : S^{\mathcal{E}} \to \mathcal{R}$ is independent of the policy agent's action. This variant will be discussed in detail in Section 4.3. Our primary focus, however, remains on the original definition.

The reward agent $\mathcal{A}_R$ does not directly interact with the environment $\mathcal{E}$ but operates through the policy agent $\mathcal{A}_P$. The suggested reward guides $\mathcal{A}_P$ in taking actions, subsequently

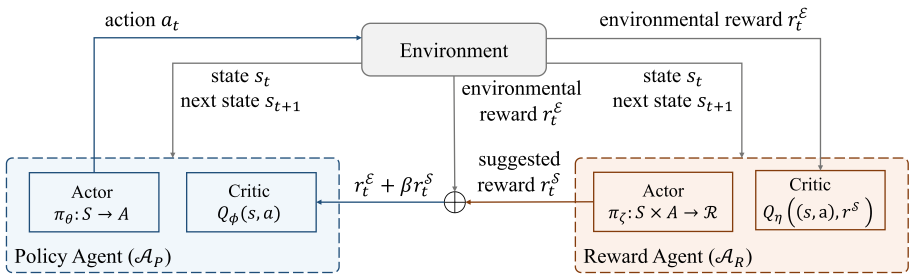

action a_t
Environment
environmental reward $r_t^{\mathcal{E}}$
state s_t
next state s_{t+1}
environmental reward $r_t^{\mathcal{E}}$
state s_t
next state s_{t+1}
suggested reward $r_t^{\mathcal{S}}$
$r_t^{\mathcal{E}} + \beta r_t^{\mathcal{S}}$
Actor $\pi_\theta: S \to A$
Critic $Q_\phi(s,a)$
Policy Agent ($\mathcal{A}_P$)
Actor $\pi_\zeta: S \times A \to \mathcal{R}$
Critic $Q_\eta\left((s,a), r^{\mathcal{S}}\right)$
Reward Agent ($\mathcal{A}_R$)

Figure 1: The overview of the Reinforcement Learning with an Assistant Reward Agent (ReLara) framework.

influencing $\mathcal{E}$. As a result, $\mathcal{A}_R$ receives the corresponding environmental reward. To simplify the process, we can regard $\mathcal{E}$ and $\mathcal{A}_P$ as a combined entity, forming a new environment $\langle \mathcal{E}, \mathcal{A}_P \rangle$ that interacts with $\mathcal{A}_R$ directly. The policy agent acts as a post-process that adopts the suggested reward from $\mathcal{A}_R$. We denote the behavior policy of $\mathcal{A}_P$ as $\pi^{\mathcal{A}_P}$, and Θ denotes its parameter space. Then the **direct** interaction between $\mathcal{A}_R$ and $\langle \mathcal{E}, \mathcal{A}_P \rangle$ is formulated as a Partially Observable MDP (POMDP), defined by the following components:

- The state space $S^{\mathcal{A}_R} = S^{\mathcal{E}} \times A^{\mathcal{E}} \times \Theta$, where $S^{\mathcal{E}}$ and $A^{\mathcal{E}}$ are environmental state and action spaces, respectively, while Θ denotes the parameter space of policy $\pi^{\mathcal{A}_P}$.
- The action space for the reward agent $\mathcal{A}_R$ is defined as the suggested reward space $\mathcal{R}$.
- The transition function is $T^{\mathcal{A}_R}(s_{t+1}^{\mathcal{A}_R} | s_t^{\mathcal{A}_R}, r_t^{\mathcal{S}}) = T^{\mathcal{E}}(s_{t+1}|s_t, a_t)\pi^{\mathcal{A}_P}(a_{t+1}|s_{t+1}; \theta_{t+1})P(\theta_{t+1}|\theta_t; r_t^{\mathcal{S}})$, where $T^{\mathcal{E}}$ is the environmental transition function, $P(\theta_{t+1}|\theta_t; r_t^{\mathcal{S}})$ is the parameter updating distribution given the suggested reward $r_t^{\mathcal{S}}$, while $s_t^{\mathcal{A}_R} = (s_t, a_t, \theta_t)$ and $s_{t+1}^{\mathcal{A}_R} = (s_{t+1}, a_{t+1}, \theta_{t+1})$, respectively.
- The reward function $R^{\mathcal{A}_R}(s^{\mathcal{A}_R}) = R^{\mathcal{E}}(s, a)$ remains the same as the environmental reward function, but only evaluates the state-action pair, ensuring that $\mathcal{A}_R$ optimizes the extrinsic reward only.
- The observation space is defined as $\Omega^{\mathcal{A}_R} = S^{\mathcal{E}} \times A^{\mathcal{E}}$, since the reward agent only observes the (s, a) pair from the environment $\langle E, \mathcal{A}_P \rangle$. Although the parameter θ is not directly observed by $\mathcal{A}_R$, it can be implied from the observed (s, a) pair, because we have $s \xrightarrow{\theta} a$, which means the action is dependent on the corresponding parameter θ.

Given the aforementioned POMDP model, $\mathcal{A}_R$ can be treated as a conventional RL agent tasked with learning a policy to propose continuous reward signals and, consequently, can be optimized using a general RL algorithm. In our framework, we employ an actor-critic approach to manage the continuous reward space. This approach involves two separate neural networks: a policy network denoted as $\pi_\zeta^{\mathcal{A}_R}$ and a Q-network denoted as $Q_\eta^{\mathcal{A}_R}$, as described by Konda & Tsitsiklis (1999). The policy network maps the state-action pair $s_t^{\mathcal{A}_R} = (s_t, a_t)$ as an observation to a suggested reward $r_t^{\mathcal{S}}$, while the Q-value for the pair $(s_t^{\mathcal{A}_R}, r_t^{\mathcal{S}})$ estimates the expected cumulative environmental rewards resulting from the indirect interaction process between $\mathcal{A}_R$ and the environment.

The Q-function is optimized by minimizing the mean squared error (MSE) between the predicted Q-value and the temporal difference target (TD-target) $y_t^{\mathcal{A}_R}$:

$$\mathcal{L}(\eta) = \left(Q_\eta^{\mathcal{A}_R}(s_t^{\mathcal{A}_R}, r_t^{\mathcal{S}}) - y_t^{\mathcal{A}_R}\right)^2, \quad (1)$$

where $y_t^{\mathcal{A}_R}$ is computed with a secondary frozen target network to maintain a fixed objective (Mnih et al., 2015):

$$y_t^{\mathcal{A}_R} = r_t^{\mathcal{E}} + \gamma Q_{\eta'}^{\mathcal{A}_R}(s_{t+1}^{\mathcal{A}_R}, r_{t+1}^{\mathcal{S}}),$$

where η' is the set of parameters of the target network which is updated by a soft update rule (Lillicrap et al., 2015).

The objective function to optimize the policy network is defined as the expected Q-value:

$$\mathcal{L}(\zeta) = -\mathbb{E}_{r_t^{\mathcal{S}} \sim \pi_\zeta(\cdot|s_t^{\mathcal{A}_R})}\left[Q_\eta^{\mathcal{A}_R}(s_t^{\mathcal{A}_R}, r_t^{\mathcal{S}})\right]. \quad (2)$$

3.2. Policy Agent with Suggested Rewards

The policy agent $\mathcal{A}_P$ interacts with the environment $\mathcal{E}$ by executing actions $a \in A^{\mathcal{E}}$ and receiving two types of rewards: the reward $r^{\mathcal{E}}$ from $\mathcal{E}$ and the suggested reward $r^{\mathcal{S}}$ from $\mathcal{A}_R$. The augmented reward for the policy agent is given by: $r^{\mathcal{A}_P} = r^{\mathcal{E}} + \beta r^{\mathcal{S}}$, where $\beta \in (0, 1]$ is a scaling weight factor. We can model the interaction between the policy agent and the environment as an MDP: $\langle S^{\mathcal{E}}, A^{\mathcal{E}}, T^{\mathcal{E}}, R^{\mathcal{A}_P} \rangle$. The state space, action space, and transition function remain the same as the environmental MDP, while the reward function for the policy agent is modified as

$$R^{\mathcal{A}_P}(s, a) = R^{\mathcal{E}}(s, a) + \beta r^{\mathcal{S}}, \qquad r^{\mathcal{S}} \sim \pi^{\mathcal{A}_R}(\cdot|s, a), \quad (3)$$

where the suggested reward is generated from the policy network of the reward agent. The conventional reward function can be recovered in the limit as $\beta \to 0$.

We adopt the soft actor-critic (SAC) (Haarnoja et al., 2018a;b) algorithm as the backbone for the policy agent, where the policy network and the Q-network are parameterized as $\pi_\theta^{\mathcal{A}_P}$ and $Q_\phi^{\mathcal{A}_P}$, respectively. The Q-function is optimized by minimizing the MSE loss:

$$\mathcal{L}(\phi) = \left(Q_\phi^{\mathcal{A}_P}(s_t, a_t) - y_t^{\mathcal{A}_P}\right)^2, \tag{4}$$

where $y^{\mathcal{A}_P}$ is computed with a target network $Q_{\phi'}^{\mathcal{A}_P}$ using the augmented reward:

$$y_t^{\mathcal{A}_P} = r_t^{\mathcal{E}} + \beta r_t^{\mathcal{S}} + \gamma Q_{\phi'}^{\mathcal{A}_P}(s_{t+1}, a_{t+1}), \tag{5}$$

where $r_t^{\mathcal{S}} \sim \pi^{\mathcal{A}_R}(\cdot|s, a)$. The policy network is optimized by maximizing the expected Q-value and the entropy of the policy $\mathcal{H}\big(\pi^{\mathcal{A}_P}(\cdot|s_t)\big)$. The objective function is defined as:

$$\mathcal{L}(\theta) = \mathbb{E}_{a_t \sim \pi_\theta^{\mathcal{A}_P}(\cdot|s_t)}\Big[-Q_\phi^{\mathcal{A}_P}(s_t, a_t) + \log \pi_\theta^{\mathcal{A}_P}(a_t|s_t)\Big]. \tag{6}$$

3.3. The ReLara Algorithm

The ReLara algorithm employs an off-policy approach to simultaneously train the policy agent $\mathcal{A}_P$ and the reward agent $\mathcal{A}_R$. Two distinct replay buffers, $\mathcal{D}_P$ and $\mathcal{D}_R$, are initialized for each agent, respectively. At each interaction with the environment, the policy agent collects a transition $\{s_t, a_t, s_{t+1}, r_t^{\mathcal{E}}\}$, while the reward agent collects a transition $\{s_t^{\mathcal{A}_R}, r_t^{\mathcal{S}}, s_{t+1}^{\mathcal{A}_R}, r_t^{\mathcal{E}}\}$. During the agent updating steps, both agents sample batches of transitions from their corresponding replay buffers and calculate the losses. Note that the suggested reward in Equation 5 is obtained from the most recently updated reward agent. In our implementation, we adopt a common strategy to parameterize double Q-functions for both $\mathcal{A}_P$ and $\mathcal{A}_R$, using the minimum of the two Q-values for the value gradient and policy gradient, following the approach proposed by Fujimoto et al. (2018). We summarize ReLara in Algorithm 1.

The policy agent and the reward agent are independently optimized by their respective RL algorithms. This allows for adaptability and context-specific customization, demonstrating generalization capabilities. Moreover, given their cooperative nature, the convergence of the overall ReLara framework is supported by the convergence of the two sub-agents. If the backbone algorithms, such as SAC, converge within their respective RL settings, then the alternating optimization of the two sub-agents should ensure the convergence of the entire framework, as they are mutually reinforcing. Nevertheless, achieving overall convergence in general remains challenging due to the inherent complexities such as exploration-exploitation balance, stochastic gradient descent updates in neural networks, and the stochastic policies and environments involved. Consequently, while empirical results indicate that ReLara can achieve convergence in practice, a formal proof is not yet available. Addressing these theoretical aspects and proving the convergence of ReLara will be a focus of future research.

Algorithm 1 RL with an Assistant Reward Agent (ReLara)

```
Require: Environment $\mathcal{E}$.
Require: Suggested reward space $\mathcal{R}$.
Require: Policy agent $\mathcal{A}_P$ with actor $\pi_\theta$ and critic $Q_\phi$.
Require: Reward agent $\mathcal{A}_R$ with actor $\pi_\zeta$ and critic $Q_\eta$.
Require: Experience replay buffers $\mathcal{D}_P$ and $\mathcal{D}_R$.
 1: for each iteration do
 2:     for each environment step do
 3:         $s_t \sim \mathcal{E}$
 4:         $a_t \sim \pi_\theta(\cdot|s_t)$
 5:         $r_t^{\mathcal{S}} \sim \pi_\zeta(\cdot|s_t^{\mathcal{A}_R})$            ▷ where $s_t^{\mathcal{A}_R} = (s_t, a_t)$
 6:         $s_{t+1}, r_t^{\mathcal{E}} \sim \mathcal{E}(a_t)$
 7:         $a_{t+1} \sim \pi_\theta(\cdot|s_{t+1})$       ▷ get $s_{t+1}^{\mathcal{A}_R} = (s_{t+1}, a_{t+1})$
 8:         $\mathcal{D}_P \leftarrow \mathcal{D}_P \cup \{s_t, a_t, s_{t+1}, r_t^{\mathcal{E}}\}$
 9:         $\mathcal{D}_R \leftarrow \mathcal{D}_R \cup \{s_t^{\mathcal{A}_R}, r_t^{\mathcal{S}}, s_{t+1}^{\mathcal{A}_R}, r_t^{\mathcal{E}}\}$
10:     end for
11:     for each gradient step do
12:         $\{s_t, a_t, s_{t+1}, r_t^{\mathcal{E}}\}_i \sim \mathcal{D}_P$
13:         $r_t^{\mathcal{S}} \sim \pi_\zeta(\cdot|s, a)$       ▷ sample the shaped rewards
14:         $\phi \leftarrow \phi - \alpha_\phi \nabla_\phi \mathcal{L}(\phi)$   ▷ Update critic $Q_\phi$ of $\mathcal{A}_P$
15:         $\theta \leftarrow \theta - \alpha_\theta \nabla_\theta \mathcal{L}(\theta)$     ▷ Update actor $\pi_\theta$ of $\mathcal{A}_P$
16:         $\{s_t^{\mathcal{A}_R}, r_t^{\mathcal{S}}, s_{t+1}^{\mathcal{A}_R}, r_t^{\mathcal{E}}\}_i \sim \mathcal{D}_R$
17:         $\eta \leftarrow \eta - \alpha_\eta \nabla_\eta \mathcal{L}(\eta)$    ▷ Update critic $Q_\eta$ of $\mathcal{A}_R$
18:         $\zeta \leftarrow \zeta - \alpha_\zeta \nabla_\zeta \mathcal{L}(\zeta)$     ▷ Update actor $\pi_\zeta$ of $\mathcal{A}_R$
19:     end for
20: end for
```

We summarize the advantages of ReLara from three perspectives. First, our reward agent provides insightful long-term information by involving cumulative environmental signals in one-step rewards, without introducing any additional objectives, which ensures the optimized policy remains consistent with the original task target. Second, ReLara manages the exploration-exploitation trade-off in an adaptive manner. At the initial learning stage, unlike conventional off-policy RL algorithms, where the transitions stored in the replay buffer predominantly consist of 0s with occasional 1s, the transitions collected for ReLara contain random specifically valued rewards, emitted by the initial reward agent. Although lacking in meaningful information, these random rewards allow the policy agent to be optimized in various directions in fractional steps. This process effectively expands its exploration range, improving the likelihood of encountering positive samples. As learning progresses and the reward agent approaches an optimal policy, it begins to generate informative rewards. These evolved rewards encourage the policy agent to shift its focus toward exploitation. (Refer to Section 4.2 for supporting experimental results). Third, the reshaped dense reward signals can enhance the gradient-based learning methods of $\mathcal{A}_P$. Previous studies (Mohtasib et al., 2021; Wu et al., 2021b) have shown that dense rewards can improve the convergence stability and efficiency of neural networks compared to sparse rewards.

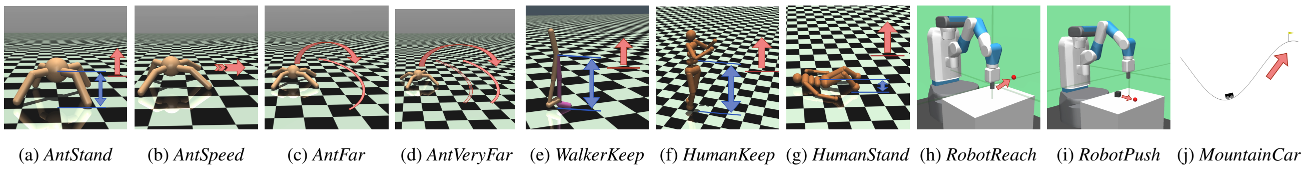


Figure 2: Environments used in our experiments. The agent receives a reward of 1 only upon completing the corresponding targets, which are given as follows: (a) *AntStand*: the ant robot stands up and keeps its center of gravity above a threshold; (b) *AntSpeed*: the ant robot crawls at a given speed; (c) *AntFar* and (d) *AntVeryFar*: both require the ant to crawl beyond a certain distance, while *AntVeryFar* aims for a greater distance; (e) *WalkerKeep*: the bipedal robot keeps its center of gravity height above a threshold; (f) *HumanKeep*: the humanoid robot keeps its center of gravity height above a threshold; (g) *HumanStand*: the humanoid robot stands up from lying down and raises its center of gravity above a threshold; (h) *RobotReach*: the end of the robot arm touches a randomly set goal point; (i) *RobotPush*: push the cube to the target point; (j) *MountainCar*: the car reachs the flag.

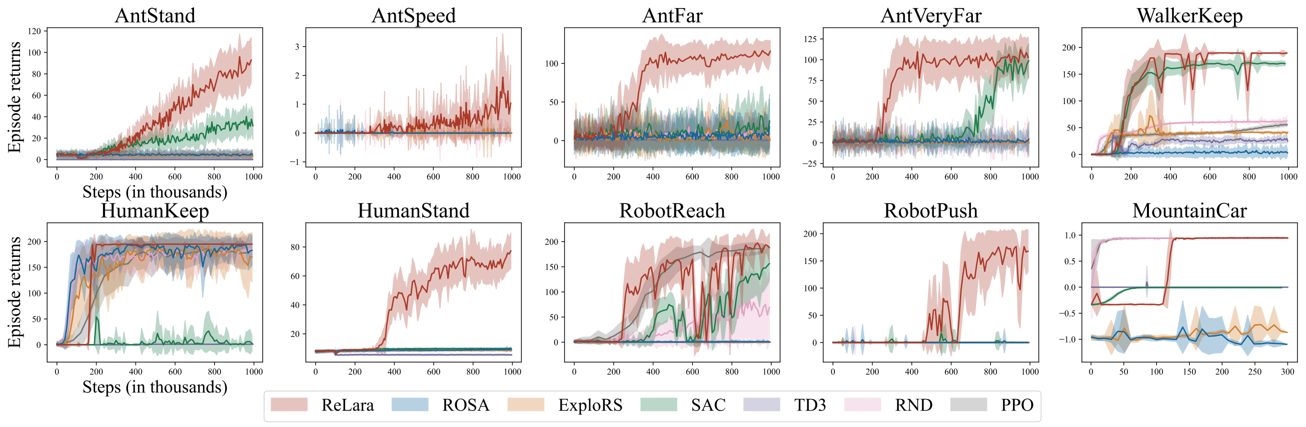


Figure 3: Comparison of the learning performance of ReLara with the baselines.

4. Experiments

We conduct experiments in continuous control tasks with challenging sparse and delayed rewards, including *MuJoCo* (Todorov et al., 2012), arm robot (de Lazcano et al., 2023) and physical control (Towers et al., 2023) domains. In these tasks, the agent receives a reward of 1 only if it accomplishes the final objective within the allowed maximum number of steps, and receives 0 for all other states. Figure 2 illustrates the tasks used in our experiments.

4.1. Comparative Evaluation

We compare ReLara with six state-of-the-art baselines: (a) The RL Optimizing Shaping Algorithm (ROSA) (Mguni et al., 2023), a reward-shaping algorithm maintaining an auxiliary agent through a two-player Markov game, which shares the same structure as ours. (b) The Exploration-Guided Reward Shaping (ExploRS) algorithm (Devidze et al., 2022), learning an intrinsic reward function in combination with exploration-based bonuses. Both ROSA and ExploRS are reward shaping algorithms, dedicated to addressing the sparse-reward challenge. (c) The Soft Actor-Critic (SAC) (Haarnoja et al., 2018a), an advanced off-policy stochastic policy algorithm that uses maximum entropy to enhance exploration. Here, we compare with the improved version with automatic entropy adjustment (Haarnoja et al., 2018b). (d) The twin delayed deep deterministic policy gradient (TD3) (Fujimoto et al., 2018), a representative deterministic policy algorithm that largely improved from the deep deterministic policy gradient (DDPG) (Lillicrap et al., 2015). (e) Random Network Distillation (RND) (Burda et al., 2018), an algorithm to utilize a fixed auxiliary network to encourage exploration, which aimed to solve sparse-reward environments. (f) Proximal Policy Optimization (PPO) (Schulman et al., 2017), a stable and effective on-policy algorithm. We use the *CleanRL* library (Huang et al., 2022) or the author-provided codes for the implementation of these baselines.

For our implementation of ReLara, we use the same hyperparameters and network structures across all experiments, the details are provided in Appendix B. We train ReLara with burn-in phases for the reward agent and the policy agent, respectively, during which, the agents randomly sample from their action spaces. We find that having the reward

Table 1: The average episodic returns and standard errors of ReLara and the baselines.

Environments	ReLara	ROSA	ExploRS	SAC	TD3	RND	PPO
AntStand	**28.66 ± 1.82**	3.80 ± 0.03	4.52 ± 0.04	15.93 ± 0.69	0.00 ± 0.00	4.23 ± 0.03	4.54 ± 0.04
AntSpeed	**0.33 ± 0.02**	0.02 ± 0.00	0.01 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.02 ± 0.00	0.00 ± 0.00
AntFar	**67.77 ± 4.30**	4.71 ± 0.28	5.42 ± 0.22	9.64 ± 0.57	0.81 ± 0.02	6.66 ± 0.25	6.33 ± 0.24
AntVeryFar	**64.07 ± 4.17**	0.64 ± 0.07	1.67 ± 0.10	20.73 ± 2.87	0.81 ± 0.02	1.00 ± 0.09	1.80 ± 0.11
WalkerKeep	**77.14 ± 8.77**	32.14 ± 1.19	2.47 ± 0.13	70.96 ± 8.10	18.62 ± 0.75	44.78 ± 1.39	33.77 ± 1.11
HumanKeep	**160.31 ± 7.30**	152.38 ± 4.98	158.09 ± 4.42	4.59 ± 0.84	0.55 ± 0.03	159.79 ± 4.27	138.13 ± 12.64
HumanStand	**29.72 ± 1.85**	8.55 ± 0.03	8.63 ± 0.03	9.31 ± 0.05	5.72 ± 0.04	8.67 ± 0.03	8.36 ± 0.03
RobotReach	103.56 ± 7.18	0.27 ± 0.03	0.79 ± 0.04	45.03 ± 4.92	0.00 ± 0.00	28.18 ± 2.53	**110.44 ± 15.00**
RobotPush	**58.71 ± 6.98**	0.00 ± 0.00	0.20 ± 0.08	0.55 ± 0.21	0.00 ± 0.00	0.04 ± 0.04	0.00 ± 0.00
MountainCar	0.89 ± 0.01	−0.90 ± 0.02	−0.99 ± 0.02	−0.05 ± 0.02	0.00 ± 0.00	**0.94 ± 0.00**	0.93 ± 0.00

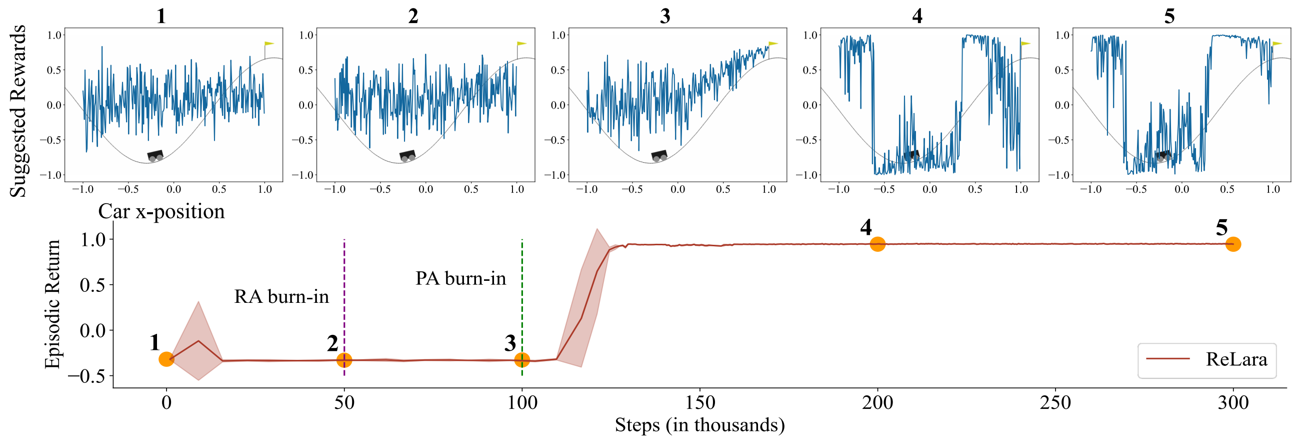


Figure 4: Distribution of the suggested rewards over the car positions in the *MountainCar* task, along the training process.

agent end the burn-in phase earlier than the policy agent achieves better learning performance.

We train multiple instances with different random seeds for each task. Figure 3 and Table 1 present the average episodic returns and standard errors throughout training. We observe that ReLara outperforms the baselines in terms of sample efficiency, convergence speed, and learning stability. Although in the *RobotReach* and *MountainCar* environments, the initial burn-in stage causes ReLara to reach the optimal policy slower than PPO and RND, which do not include burn-in phases, after the initial period, ReLara's convergence rate is comparable to or surpasses that of PPO and RND, achieving close overall average episodic returns. The primary challenge in these environments is the delayed, sparse rewards coupled with extremely long horizons, where the agent receives no feedback until it completes the task at least once. Therefore, the policy's update critically depends on the exploration ability to obtain successful trajectories. This challenge is particularly salient in tasks *AntFar*, *HumanStand*, and *RobotPush*, where all baselines failed to learn effective policies. Although approaches like ROSA, ExploRS, RND, and SAC have advanced agent exploration strategies in sparse reward contexts, these methodologies predominantly reward new actions or states by introducing extra objectives. The emphasis on novelty, while neglecting their intrinsic values, can lead to indiscriminate exploration, which often results in getting stuck in local optima.

In contrast, ReLara effectively tackles sparse rewards without incorporating extraneous objectives. Its reward agent combines immediate environmental feedback with trajectory-contextual suggested rewards for each state-action pair, assessing the intrinsic quality and future potential. This approach not only empowers the policy agent to evaluate action implications more precisely but also mitigates the risks of local optima and domination of external rewards, ensuring focus on the primary optimization goals. Additionally, the rewards densification and smoothing improve the efficiency of the policy agent's gradient descent. ReLara also promotes exploration by initially injecting random reward signals and progressively shifts the agent's focus from exploration to exploitation by evolving into meaningful reward values. Detailed experiments and discussions on this mechanism are elaborated in Section 4.2. Collectively, these strategies significantly enhance ReLara's effectiveness and robustness in long-horizon tasks with sparse rewards.

4.2. Self-Adaptive Exploration-Exploitation Balance

ReLara achieves a self-adaptive trade-off between exploration and exploitation through the cooperation of the re-

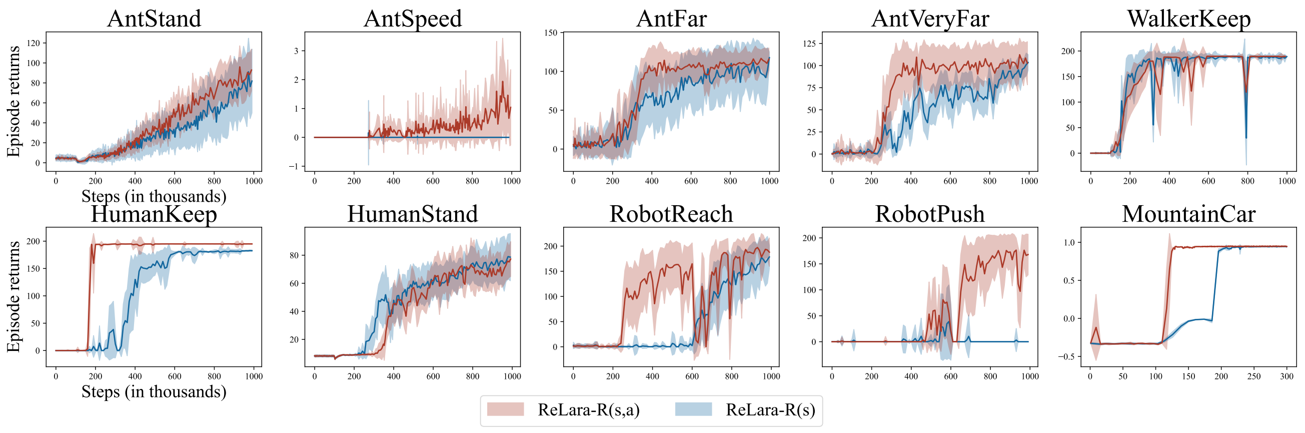


Figure 5: Comparison of ReLara with the state-action-dependent function $R(s, a)$ and state-dependent function $R(s)$.

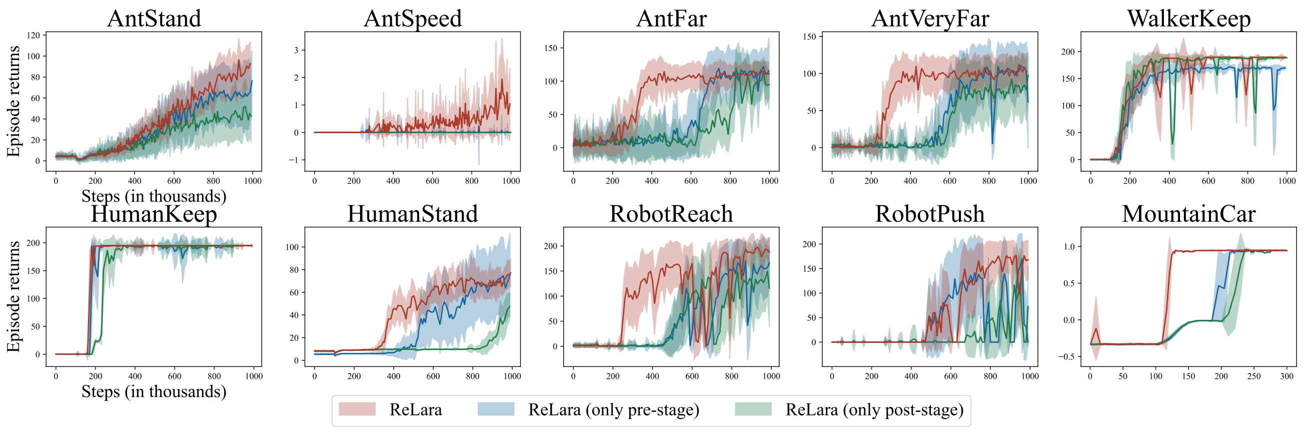


Figure 6: The comparison of ReLara with the reward agent is only involved in the first- or second-half stages.

ward and policy agents. To better illustrate, we use the *MountainCar* task with its one-dimensional state space to visualize the suggested rewards for different car positions in Figure 4. The lower part of the figure is the convergence curve, while the upper five plots display the evolving distributions of the suggested rewards across the state space at different learning stages. We set the action dimension to 0 so that the reward agent only considers the car's position.

In the reward agent burn-in stage (<50,000th step), the suggested rewards act as a random and uniform perturbation to the environmental rewards. As mentioned earlier, unlike conventional off-policy methods, which predominantly store transitions with sparse rewards in their replay buffers, ReLara stores transitions with specifically valued rewards. Although the rewards are not meaningful initially, they still optimize the policy agent in multiple directions for small steps, as each state has an even potential to be highly rewarding, making it worthwhile for the policy agent to have a try. This strategy encourages extensive exploration, increasing the likelihood of gathering positive samples. As training progresses, the reward agent learns a more informative reward function. For instance, at the 100,000th step, states near the flag are assigned higher rewards, offering essential guidance to the policy agent in its initial learning phase, thus improving sample efficiency. Ultimately, as the reward agent converges to a meaningful suggested reward function, it encourages the policy agent towards exploitation, reinforcing actions to reach the flag consistently. Lastly, the convergence of the reward agent is also dependent on the policy agent's learning, creating a mutually beneficial relationship and enabling the self-adaptive exploration-exploitation balance mechanism.

4.3. Ablation Studies

We perform ablation studies to understand the effects of (1) the policy agent's actions as part of the input for the reward agent, (2) the role the reward agent plays during pre- and post-training stages, and (3) different scales of the suggested rewards controlled by the hyperparameter β.

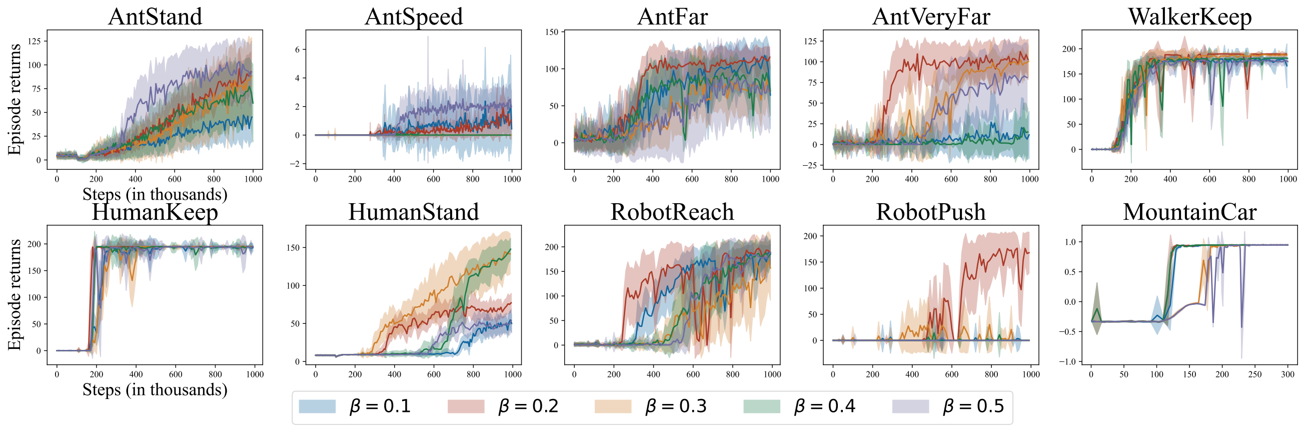


Figure 7: The comparison of different suggested reward weight factor β in ReLara.

Reward function $R(s,a)$ vs. $R(s)$. We compare ReLara that learns a state-action dependent reward function $R(s,a) : S \times A \to \mathcal{R}$ with a variant that learns a state-dependent function $R(s) : S \to \mathcal{R}$. Figure 5 shows the episodic returns along training, the detailed data is provided in Appendix A.1. ReLara$-R(s,a)$ generally outperforms or matches ReLara$-R(s)$ across tasks. This suggests that incorporating the actions from the policy agent enables the reward agent to learn a more informative reward function, which also assesses the quality of the selected actions. In relatively simpler tasks, like *WalkerKeep* and *HumanKeep*, where the state information largely outweighs the action information, including actions shows limited improvement.

Reward agent involvement in first vs. second-half stages. We investigate how the reward agent affects the learning process at different stages. We compare two scenarios: one with the augmented reward function (Equation 3) applied only in the first half of training, followed by only the environmental rewards in the latter half, and another with the reverse order. The results are shown in Figure 6, and the detailed data is provided in Appendix A.2. We observe that the reward agent $\mathcal{A}_R$ is essential in both phases, but it has a stronger impact on the agent's early exploration stage. The absence of $\mathcal{A}_R$ in the initial training delays optimal policy achievement. Furthermore, learning without the $\mathcal{A}_R$ in the later phase exhibits more oscillation.

Suggested reward weight factor β. We examine the influence of different scales of the suggested rewards, controlled by the hyperparameter β, as shown in Figure 7, and detailed data is provided in the Appendix A.3. Our findings indicate that ReLara's performance is generally robust to changes in β, although larger values can introduce higher variance. Notably, optimal results are frequently achieved with suggested rewards at immediate values, particularly when $\beta = 0.2$ or $\beta = 0.3$. Too small a suggested reward fails to effectively reshape the original reward, while too large a reward can overwhelm the environmental feedback, causing excessive perturbation and potentially hindering the policy agent's convergence.

5. Related Work

Our work is mostly motivated by Reward Shaping (RS) approaches due to their proven effectiveness in sparse-reward environments. We review previous reward shaping works in the following three categories.

Demonstration-Based RS extracts instructive rewards from human demonstrations. Inverse reinforcement learning studies to extract reward functions from observed optimal behavior of expert actings, which is a front-loaded, isolation task, with minimal consideration given to subsequent applications of the extracted reward functions (Arora & Doshi, 2021; Hadfield-Menell et al., 2016; Ziebart et al., 2008; Ramachandran & Amir, 2007). Brys et al. (2015) utilized a potential-based approach to bias the RL process, Ellis et al. (2021); Bıyık et al. (2022) used Bayesian reward inference to integrate multiple sources of human feedback, while Cheng et al. (2021) constructed heuristics from offline data to reshape the reward and shorten the horizon of the original problem. Another line of works combined with imitation learning. Wu et al. (2021a) utilized state-action potentials trained from generative models, while Brown et al. (2020) used pairwise comparisons of expert demonstrations to elicit the posterior distribution over reward functions. The manual knowledge engineering required by these methods is often expensive, and the ability to transform into out-of-distribution tasks is limited.

Intrinsic-Motivation-Based RS studies self-supervised approaches free from human knowledge to excavate the intrinsic rewards. The potential functions based RS algorithms guarantee that the induced optimal policy is also optimal under the extrinsic reward function (Ng et al., 1999; Wiewiora, 2003; Asmuth et al., 2008; Devlin & Kudenko, 2012). Another branch of RS algorithms leverages the exploration bonuses to measure the novelty of the states to encourage exploring the environment (Devidze et al., 2022; Sun et al., 2022; Bellemare et al., 2016; Ostrovski et al., 2017; Tang et al., 2017). However, these exploration-based rewards usually dominate environmental rewards to trap the agent in distractive zones. Curiosity-driven methods alleviated this limitation by rewarding the appearance of surprising unanticipated states. Pathak et al. (2017) generated intrinsic rewards based on the error in predicting the consequences of its own actions in a learned feature space. Burda et al. (2018) computed the error by mimicking a fixed randomly initialized neural network. Additionally, Zheng et al. (2018) followed the idea of minimizing the KL-divergence between the intrinsic and extrinsic reward distributions. Trott et al. (2019) involved distance-to-goal shaped rewards. Devidze et al. (2021) investigated discrete optimization to design explicable reward functions. Memarian et al. (2021) used a classifier to infer rewards by ranking trajectories that the agent observed. Mezghani et al. (2023) learned from the pre-collected data to understand the structure and dynamics of the environment through hindsight relabelling. However, these algorithms may fail in environments with extremely sparse extrinsic rewards. ReLara addresses these limitations by achieving a self-adaptive exploration-exploitation and intrinsic-extrinsic balance.

Multi-Module-Based RS typically involves multiple agents or policies. Mguni et al. (2023) introduced the RL Optimizing Shaping Algorithm (ROSA), in which a *Shaper* dynamically switched shaped rewards for the *Controller* through a two-player Markov game. ReLara shares the same two-agent structure as ROSA, but the difference is that ROSA plays a zero-sum game where agents compete against each other, while ReLara is a cooperative framework where agents collaborate to complete the task. Besides, Guo et al. (2018) introduced Generative Adversarial Networks to encourage the agent to move closer to its previous beneficial experiences, while based on which, Altmann et al. (2023) trained a policy to discriminate between the current trajectory and previously generated beneficial trajectories. Establishing hierarchical structures is also a popular approach. Yi et al. (2022) trained a high-level policy to share rewards with neighboring low-level agents in a networked multi-agent RL setting. Gupta et al. (2024) used a bi-level objective to learn behavior alignment reward functions. Hu et al. (2020) proposed a framework where a high-level policy optimized the shaped weight function, while a low-level policy maximized the modified reward function. Stadie et al. (2020) used Self-Tuning Networks to learn the response function that mapped intrinsic reward function parameters to optimal policy parameters. Ma et al. (2024) incorporated Markov random field to formalize a Bayesian approach. However, hierarchical approaches usually train multiple agents alternately, in contrast, ReLara optimizes both agents concurrently, resulting in a more efficient process.

6. Discussion and Conclusion

We present ReLara, a framework that integrates reward shaping with a dual-agent cooperative system, consisting of a policy-learning agent and a reward-suggesting agent. ReLara integrates future-oriented dense rewards with sparse environmental rewards, providing informative and fine-grained insights for the policy agent to effectively discern among states. A distinctive feature of ReLara is the initial injection of random rewards, which are progressively evolved into optimized signals, adaptively regulating the exploration-exploitation balance. This strategy significantly improves learning efficiency and stability. Tested on long-horizon continuous control tasks with sparse rewards, ReLara demonstrates its robustness and superior performance.

We acknowledge that the advantages of ReLara rely on the premise that the environment offers sparse and delayed rewards. In scenarios where the environments provide dense and informative rewards, the auxiliary agent may be redundant or even counterproductive. For instance, in the *HumanStand* task, if the environmental reward is defined by the robot's center of gravity height – a direct and pertinent objective, introducing supplementary rewards could potentially hinder early training. Moreover, optimizing two agents presents more complexity than a single agent. Hence, adapting ReLara in dense-reward tasks and minimizing algorithmic complexity are important future directions.

Acknowledgments

This research is supported by an Academic Research Grant No. MOE-T2EP20121-0015 from the Ministry of Education in Singapore.

Impact Statement

This paper introduces the ReLara framework, which significantly enhances reinforcement learning algorithms in sparse-reward environments. These environments are very common in real-world scenarios where defining informative and dense rewards is often impractical and resource-consuming. ReLara improves sample efficiency, requiring fewer interactions to achieve optimal results. This is particularly crucial in domains such as robotics, autonomous driving, and other fields where agent-environment interactions are costly or potentially hazardous. By optimizing learning processes in these contexts, ReLara contributes to safer, more efficient, and effective applications, ultimately advancing the field of Machine Learning.

References

Altmann, P., Phan, T., Ritz, F., Gabor, T., and Linnhoff-Popien, C. Direct: Learning from sparse and shifting rewards using discriminative reward co-training. *arXiv preprint arXiv:2301.07421*, 2023.

Arora, S. and Doshi, P. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297:103500, 2021.

Asmuth, J., Littman, M. L., and Zinkov, R. Potential-based shaping in model-based reinforcement learning. In *AAAI Conference on Artificial Intelligence*, pp. 604–609, 2008.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. Unifying count-based exploration and intrinsic motivation. *Advances in Neural Information Processing Systems*, 29, 2016.

Bıyık, E., Losey, D. P., Palan, M., Landolfi, N. C., Shevchuk, G., and Sadigh, D. Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences. *The International Journal of Robotics Research*, 41(1):45–67, 2022.

Brown, D., Coleman, R., Srinivasan, R., and Niekum, S. Safe imitation learning via fast bayesian reward inference from preferences. In *International Conference on Machine Learning*, pp. 1165–1177. PMLR, 2020.

Brys, T., Harutyunyan, A., Suay, H. B., Chernova, S., Taylor, M. E., and Nowé, A. Reinforcement learning from demonstration through shaping. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation. In *International Conference on Learning Representations*, 2018.

Cheng, C.-A., Kolobov, A., and Swaminathan, A. Heuristic-guided reinforcement learning. *Advances in Neural Information Processing Systems*, 34:13550–13563, 2021.

de Lazcano, R., Andreas, K., Tai, J. J., Lee, S. R., and Terry, J. Gymnasium robotics, 2023. URL `http://github.com/Farama-Foundation/Gymnasium-Robotics`.

Devidze, R., Radanovic, G., Kamalaruban, P., and Singla, A. Explicable reward design for reinforcement learning agents. *Advances in Neural Information Processing Systems*, 34:20118–20131, 2021.

Devidze, R., Kamalaruban, P., and Singla, A. Exploration-guided reward shaping for reinforcement learning under sparse rewards. *Advances in Neural Information Processing Systems*, 35:5829–5842, 2022.

Devlin, S. M. and Kudenko, D. Dynamic potential-based reward shaping. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, pp. 433–440. IFAAMAS, 2012.

Ellis, C., Wigness, M., Rogers, J., Lennon, C., and Fiondella, L. Risk averse bayesian reward learning for autonomous navigation from human demonstration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 8928–8935. IEEE, 2021.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pp. 1587–1596. PMLR, 2018.

Guo, Y., Oh, J., Singh, S., and Lee, H. Generative adversarial self-imitation learning. *arXiv preprint arXiv:1812.00950*, 2018.

Gupta, A., Pacchiano, A., Zhai, Y., Kakade, S., and Levine, S. Unpacking reward shaping: Understanding the benefits of reward engineering on sample complexity. *Advances in Neural Information Processing Systems*, 35:15281–15295, 2022.

Gupta, D., Chandak, Y., Jordan, S., Thomas, P. S., and C da Silva, B. Behavior alignment via reward function optimization. *Advances in Neural Information Processing Systems*, 36, 2024.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pp. 1861–1870. PMLR, 2018a.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

Hadfield-Menell, D., Russell, S. J., Abbeel, P., and Dragan, A. Cooperative inverse reinforcement learning. *Advances in Neural Information Processing Systems*, 29, 2016.

Hu, Y., Wang, W., Jia, H., Wang, Y., Chen, Y., Hao, J., Wu, F., and Fan, C. Learning to utilize shaping rewards: A new approach of reward shaping. *Advances in Neural Information Processing Systems*, 33:15931–15941, 2020.

Huang, S., Dossa, R. F. J., Ye, C., Braga, J., Chakraborty, D., Mehta, K., and AraÃšjo, J. G. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022.

Konda, V. and Tsitsiklis, J. Actor-critic algorithms. *Advances in Neural Information Processing Systems*, 12, 1999.

Ladosz, P., Weng, L., Kim, M., and Oh, H. Exploration in deep reinforcement learning: A survey. *Information Fusion*, 85:1–22, 2022.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Ma, H., Vo, T. V., and Leong, T.-Y. Mixed-initiative bayesian sub-goal optimization in hierarchical reinforcement learning. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, pp. 1328–1336, 2024.

Memarian, F., Goo, W., Lioutikov, R., Niekum, S., and Topcu, U. Self-supervised online reward shaping in sparse-reward environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2369–2375. IEEE, 2021.

Mezghani, L., Sukhbaatar, S., Bojanowski, P., Lazaric, A., and Alahari, K. Learning goal-conditioned policies offline with self-supervised reward shaping. In *Conference on Robot Learning*, pp. 1401–1410. PMLR, 2023.

Mguni, D., Jafferjee, T., Wang, J., Perez-Nieves, N., Song, W., Tong, F., Taylor, M., Yang, T., Dai, Z., Chen, H., et al. Learning to shape rewards using a game of two partners. In *AAAI Conference on Artificial Intelligence*, volume 37, pp. 11604–11612, 2023.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Mohtasib, A., Neumann, G., and Cuayáhuitl, H. A study on dense and sparse (visual) rewards in robot policy learning. In *Towards Autonomous Robotic Systems: 22nd Annual Conference, TAROS 2021, Lincoln, UK, September 8–10, 2021, Proceedings 22*, pp. 3–13. Springer, 2021.

Ng, A. Y., Harada, D., and Russell, S. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*, volume 99, pp. 278–287. Citeseer, 1999.

Ostrovski, G., Bellemare, M. G., Oord, A., and Munos, R. Count-based exploration with neural density models. In *International Conference on Machine Learning*, pp. 2721–2730. PMLR, 2017.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, pp. 2778–2787. PMLR, 2017.

Ramachandran, D. and Amir, E. Bayesian inverse reinforcement learning. In *International Joint Conference on Artificial Intelligence*, volume 7, pp. 2586–2591, 2007.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Singh, S., Lewis, R. L., Barto, A. G., and Sorg, J. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2):70–82, 2010.

Sorg, J., Lewis, R. L., and Singh, S. Reward design via online gradient ascent. *Advances in Neural Information Processing Systems*, 23, 2010a.

Sorg, J., Singh, S. P., and Lewis, R. L. Internal rewards mitigate agent boundedness. In *International Conference on Machine Learning*, pp. 1007–1014, 2010b.

Stadie, B., Zhang, L., and Ba, J. Learning intrinsic rewards as a bi-level optimization problem. In *Conference on Uncertainty in Artificial Intelligence*, pp. 111–120. PMLR, 2020.

Sun, H., Han, L., Yang, R., Ma, X., Guo, J., and Zhou, B. Exploit reward shifting in value-based deep-rl: Optimistic curiosity-based exploration and conservative exploitation via linear reward shaping. *Advances in Neural Information Processing Systems*, 35:37719–37734, 2022.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Tang, H., Houthooft, R., Foote, D., Stooke, A., Xi Chen, O., Duan, Y., Schulman, J., DeTurck, F., and Abbeel, P. # exploration: A study of count-based exploration for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 30, 2017.

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

Towers, M., Terry, J. K., Kwiatkowski, A., Balis, J. U., Cola, G. d., Deleu, T., Goulão, M., Kallinteris, A., KG, A., Krimmel, M., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J. J., Shen, A. T. J., and Younis, O. G. Gymnasium, March 2023. URL `https://zenodo.org/record/8127025`.

Trott, A., Zheng, S., Xiong, C., and Socher, R. Keeping your distance: Solving sparse reward tasks using self-balancing shaped rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

Wiewiora, E. Potential-based shaping and q-value initialization are equivalent. *Journal of Artificial Intelligence Research*, 19:205–208, 2003.

Wu, Y., Mozifian, M., and Shkurti, F. Shaping rewards for reinforcement learning with imperfect demonstrations using generative models. In *IEEE International Conference on Robotics and Automation*, pp. 6628–6634. IEEE, 2021a.

Wu, Z., Lian, W., Unhelkar, V., Tomizuka, M., and Schaal, S. Learning dense rewards for contact-rich manipulation tasks. In *IEEE International Conference on Robotics and Automation*, pp. 6214–6221. IEEE, 2021b.

Yi, Y., Li, G., Wang, Y., and Lu, Z. Learning to share in networked multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 35:15119–15131, 2022.

Zheng, Z., Oh, J., and Singh, S. On learning intrinsic rewards for policy gradient methods. *Advances in Neural Information Processing Systems*, 31, 2018.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., Dey, A. K., et al. Maximum entropy inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

A. Supplementary Experimental Results

A.1. Ablation Study – Comparison of Reward Functions $R(s, a)$ and $R(s)$

The detailed average episodic returns and standard errors of the two variants of the ReLara algorithm, one to learn the reward function $R(s, a)$ and another to learn $R(s)$, is presented in Table 2. Combining the results in Figure 5, both variants achieve similar performance in most environments, except in the two robot tasks, where the $R(s, a)$ variant outperforms the $R(s)$ variant significantly because the actions taken by the agent play a more critical role in evaluating the reward values in these tasks, thus incorporating them into the reward shaping provides better insights.

Table 2: The average episodic returns and standard errors of ReLara with two different reward functions.

Environments	ReLara-$R(s, a)$	ReLara-$R(s)$
AntStand	$\mathbf{28.66 \pm 1.82}$	25.87 ± 1.47
AntSpeed	$\mathbf{0.33 \pm 0.02}$	0.00 ± 0.00
AntFar	$\mathbf{67.77 \pm 4.30}$	56.12 ± 3.50
AntVeryFar	$\mathbf{64.07 \pm 4.17}$	41.95 ± 3.23
WalkerKeep	77.14 ± 8.77	$\mathbf{79.27 \pm 6.38}$
HumanKeep	$\mathbf{160.31 \pm 7.30}$	108.89 ± 7.95
HumanStand	29.72 ± 1.85	$\mathbf{31.89 \pm 1.89}$
RobotReach	$\mathbf{103.56 \pm 7.18}$	48.42 ± 6.43
RobotPush	$\mathbf{58.71 \pm 6.98}$	2.22 ± 0.68
MountainCar	$\mathbf{0.89 \pm 0.01}$	0.78 ± 0.04

A.2. Ablation Study – The Reward Agent is Only Involved in the First or Second Half Stages

The detailed average episodic returns and standard errors of the comparison of the ReLara algorithm with the reward agent only involved in the first or second half stages are shown in Table 3.

Table 3: The average episodic returns and standard errors of ReLara with two variants that the reward agent is only involved in the first or second half stages.

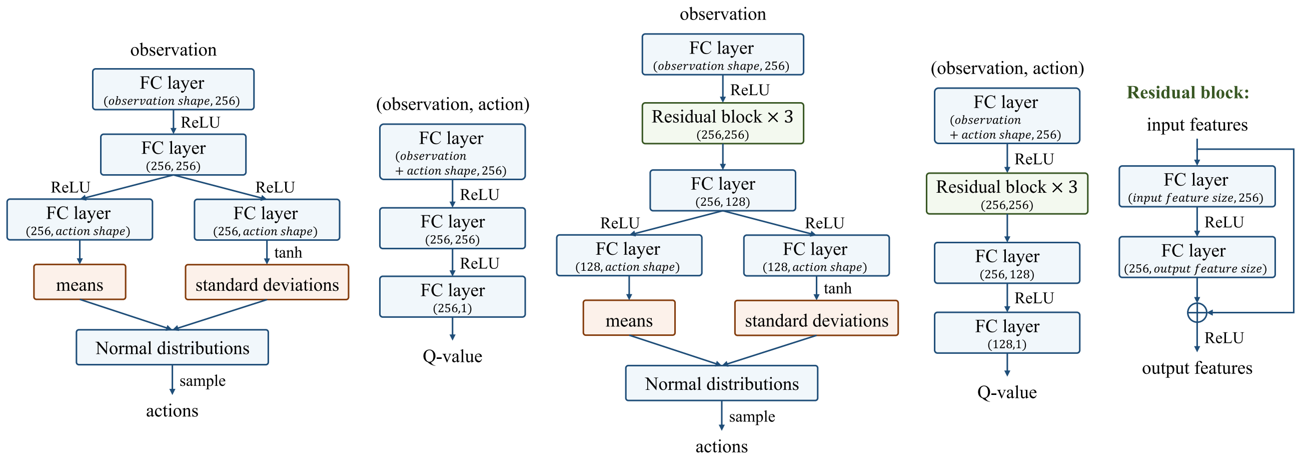
Environments	ReLara	ReLara ($\mathcal{A}_R$ only pre-stage)	ReLara ($\mathcal{A}_R$ only post-stage)
AntStand	$\mathbf{28.66 \pm 1.82}$	24.72 ± 1.64	19.74 ± 0.97
AntSpeed	$\mathbf{0.33 \pm 0.02}$	0.01 ± 0.00	0.00 ± 0.00
AntFar	$\mathbf{67.77 \pm 4.30}$	40.94 ± 3.98	26.49 ± 2.99
AntVeryFar	$\mathbf{64.07 \pm 4.17}$	34.60 ± 3.85	28.69 ± 3.03
WalkerKeep	$\mathbf{77.14 \pm 8.77}$	69.28 ± 5.63	68.63 ± 6.37
HumanKeep	$\mathbf{160.31 \pm 7.30}$	156.93 ± 7.42	146.64 ± 8.01
HumanStand	$\mathbf{29.72 \pm 1.85}$	18.22 ± 1.51	10.26 ± 0.36
RobotReach	$\mathbf{103.56 \pm 7.18}$	52.52 ± 6.25	49.39 ± 5.45
RobotPush	$\mathbf{58.71 \pm 6.98}$	40.04 ± 5.31	12.27 ± 2.64
MountainCar	$\mathbf{0.89 \pm 0.01}$	0.77 ± 0.04	0.66 ± 0.05

A.3. Ablation Study – Comparison of Different Suggested Reward Weight Factor

We present the detailed average episodic returns and standard errors of the ReLara algorithm with different β values, the factor that controls the scale of the suggested reward in Table 4. Generally speaking, when the scale of the suggested reward is around half of the original environmental reward, it can achieve better performance, while too large or too small scales will have relatively negative effects.

Table 4: The average episodic returns and standard errors with different suggested reward weight factor β in ReLara.

Environments	$\beta = 0.1$	$\beta = 0.2$ (default)	$\beta = 0.3$	$\beta = 0.4$	$\beta = 0.5$
AntStand	18.76 ± 0.90	28.66 ± 1.82	29.75 ± 1.68	28.39 ± 1.83	$\mathbf{41.23 \pm 2.79}$
AntSpeed	0.48 ± 0.03	0.33 ± 0.02	0.00 ± 0.00	0.01 ± 0.00	$\mathbf{1.43 \pm 0.06}$
AntFar	56.00 ± 3.54	$\mathbf{67.77 \pm 4.30}$	37.20 ± 2.70	49.17 ± 3.27	37.32 ± 2.57
AntVeryFar	5.41 ± 0.49	$\mathbf{64.07 \pm 4.17}$	39.06 ± 3.69	1.51 ± 0.21	13.93 ± 1.88
WalkerKeep	75.48 ± 6.02	77.14 ± 8.77	$\mathbf{78.76 \pm 6.30}$	73.45 ± 5.92	71.50 ± 5.91
HumanKeep	150.38 ± 7.60	$\mathbf{160.31 \pm 7.30}$	147.79 ± 7.72	156.46 ± 7.51	153.67 ± 7.45
HumanStand	15.66 ± 0.96	29.72 ± 1.85	$\mathbf{36.95 \pm 3.01}$	23.85 ± 2.59	17.36 ± 1.08
RobotReach	92.92 ± 7.16	$\mathbf{103.56 \pm 7.18}$	49.72 ± 5.34	68.49 ± 7.26	66.64 ± 7.72
RobotPush	0.31 ± 0.11	$\mathbf{58.71 \pm 6.98}$	5.58 ± 0.88	0.41 ± 0.19	0.11 ± 0.06
MountainCar	0.88 ± 0.02	$\mathbf{0.89 \pm 0.01}$	0.82 ± 0.02	0.89 ± 0.02	0.78 ± 0.03


observation
FC layer (observation shape, 256)
ReLU
FC layer (256, 256)
FC layer (256, action shape)
tanh
means
standard deviations
Normal distributions
sample
actions
(observation, action)
FC layer (observation + action shape, 256)
FC layer (256,1)
Q-value
Residual block × 3 (256,256)
FC layer (256, 128)
FC layer (128, action shape)
FC layer (128,1)
Residual block:
input features
FC layer (input feature size, 256)
FC layer (256, output feature size)
output features

(a) policy network for $\mathcal{A}_P$ (b) Q-network for $\mathcal{A}_P$ (c) policy network for $\mathcal{A}_R$ (d) Q-network for $\mathcal{A}_R$ (e) Residual block

Figure 8: The network structures of the policy agent and the reward agent.

Table 5: The hyperparameters used in the ReLara algorithm.

Hyperparameters	For Reward Agent $\mathcal{A}_R$	For Policy Agent $\mathcal{A}_P$
batch size	256	256
actor module learning rate	3×10^{-4}	3×10^{-4}
critic module learning rate	1×10^{-3}	1×10^{-3}
maximum entropy term	False	True
entropy term factor α learning rate	-	1×10^{-4}
policy networks update frequency (steps)	2	2
target networks update frequency (steps)	1	1
target networks soft update weight τ	0.005	0.005
burn-in steps	$5,000$	$10,000$

B. Network Structures and Hyperparameters

B.1. Network Structures

Figure 8 illustrates the structures of the policy networks and Q-networks for the policy agent and reward agent, respectively. The reward agent employs three additional residual blocks in its networks to prevent gradient vanishing or exploding, while the policy agent adopts simple multilayer perceptron (MLP) models in its networks. Since we use stochastic policies, the policy networks have distinct heads to produce the means and standard errors of the inferred normal distributions, which are subsequently used to sample actions (and suggested rewards) from the corresponding distributions.

B.2. Hyperparameters

We have observed that ReLara demonstrated high robustness and was not sensitive to hyperparameter choices. Table 5 shows the set of hyperparameters that we used in all of our experiments. The reward agent and the policy agent had the same hyperparameters except that the policy agent incorporated an extra maximum entropy term. We have observed that setting a longer burn-in phase for the policy agent than the reward agent leads to better learning performance.