Policy Gradient From Demonstration and Curiosity

Jie Chen and Wenjun Xu

Abstract—With reinforcement learning, an agent can learn complex behaviors from high-level abstractions of the task. However, exploration and reward shaping remain challenging for existing methods, especially in scenarios where extrinsic feedback is sparse. Expert demonstrations have been investigated to solve these difficulties, but a tremendous number of high-quality demonstrations are usually required. In this work, an integrated policy gradient algorithm is proposed to boost exploration and facilitate intrinsic reward learning from only a limited number of demonstrations. We achieved this by reformulating the original reward function with two additional terms, where the first term measured the Jensen–Shannon divergence between current policy and the expert’s demonstrations, and the second term estimated the agent’s uncertainty about the environment. The presented algorithm was evaluated by a range of simulated tasks with sparse extrinsic reward signals, where only limited demonstrated trajectories were provided to each task. Superior exploration efficiency and high average return were demonstrated in all tasks. Furthermore, it was found that the agent could imitate the expert’s behavior and meanwhile sustain high return.

Index Terms—Curiosity-driven exploration, learn from demonstration, policy gradient, reinforcement learning (RL).

I. INTRODUCTION

OVER the last decade, reinforcement learning (RL) [1] has achieved impressive success in various applications. Based on experiences collected through interaction with the environment, an agent learns a decision-making strategy by means of trial and error. Mnih et al. [2] trained an agent with deep Q networks (DQNs) to play Atari games and achieve professional-level performance across a set of 49 games. In 2016, by incorporating human knowledge, Monte-Carlo tree search (MCTS), and self-play, Silver et al. [3] built the very first agent, AlphaGo, to defeat a professional human Go player. Recently, with the help of a large-scale distributed training infrastructure, RL has been applied to real-time strategy multiplayer video games, which are thought to be very challenging due to issues, such as long time horizons, partially observable environments, and high-dimensional state and action spaces. The OpenAI Five defeated the Dota 2 world champion in 2019 [4]. Vinyals et al. [5] proposed the AlphaStar agent to master the game of StarCraft II and was rated at Grandmaster level. Apart from game AI, another line of research centers on the application of RL to solve challenging continuous nonlinear control problems, where the system dynamics are not fully known, by integrating RL with traditional robust [6] and adaptive [7] control algorithms, or by training end-to-end RL policies through extensive reward shaping [8].

Alongside the tremendous success of RL, exploration [9] and reward shaping [10] remain challenging for existing algorithms. The agent struggles to learn, especially when the extrinsic reward signals are sparse or the exploration spaces are huge. Recently, RL from demonstration has attracted intensive research interest as a promising way to address the aforementioned problems. However, existing algorithms usually require a tremendous number of high-quality demonstrations [11] or include a human expert in the learning loop [12], which are often difficult or unavailable. Exploration bonuses [13] have been studied to augment the original environmental rewards to alleviate exploration difficulty, which leads to the discovery of novel states; and improved knowledge about the environmental dynamics represented by information gain [14] and prediction models [15]. However, without sufficient guidance, extra exploration may drive the agent to undesirable or even dangerous regions.

To this end, we propose an integrated algorithm in this work, called policy gradient from demonstration and curiosity (PGfDC), with the aim of facilitating exploration boosting and intrinsic reward learning from a limited number of demonstrations in scenarios where the extrinsic reward signals are extremely sparse. The intuition behind PGfDC is that during interaction with the environment, when the extrinsic reward signals are sparse or even absent, an agent should imitate the demonstrated behaviors. When it struggles in states where neither extrinsic reward nor demonstration data are available, an agent should attempt to explore novel states to minimize its uncertainty about the environment. After a sufficient number of iterations, the agent can explore the environment on its own.

To facilitate PGfDC, the original extrinsic reward function is reformulated by two additional terms, which are derived from demonstration and curiosity, respectively. The demonstration term is established by computing the Jensen–Shannon divergence [16] between the agent’s current policy and that of the expert. The concept of occupancy measure is introduced to approximate the policy divergence, by measuring the difference between self-generated data and expert demonstration. To estimate the curiosity term, a neural network has been implemented to embed the agent’s observations and predict the consequences of its actions. The uncertainties about the environment are measured to represent the curiosity reward. Most policy gradient algorithms, such as proximal policy...
optimization (PPO) [17] and trust region policy optimization (TRPO) [18], are compatible with PGfDC. In this work, PGfDC was evaluated on a range of tasks, including grid world navigation and robotics control tasks with extremely sparse rewards or high-dimensional spaces. PGfDC has the potential to tackle real-world RL problems, for instance, human-robot interaction, autonomous driving, and game AI.

We summarize our contributions as follows.

1) We reformulate the original sparse reward signals with two additional intrinsic reward terms: a) the demonstration reward and b) the curiosity reward, to fully leverage the benefits of the two.
2) We propose practical algorithms by integrating the intrinsic reward learning modules into PPO in both synchronous and asynchronous ways.
3) With extensive experimental evaluations, we demonstrated that PGfDC has the potential to: a) reduce the required number of demonstrations; b) improve exploration efficiency; and c) imitate the expert and, meanwhile, achieve high return.

II. RELATED WORK

Curiosity-Driven Exploration: The concept of curiosity has been explored in the RL community to derive intrinsic rewards to boost exploration in environments with sparse rewards. The curiosity reward is designed to encourage the agent to explore states that can improve knowledge of the environment or to minimize uncertainty about the environment. Curiosity can be represented by prediction models [19], such as the forward dynamics of the environment. Stadie et al. [20] trained a forward dynamics model in the encoding space whose normalized prediction error was defined as the intrinsic reward. Similarly, Pathak et al. [15] designed an intrinsic curiosity module (ICM) by learning prediction models with self-supervised inverse dynamic models as state encodings. Burda et al. [21] performed a large-scale study of purely curiosity-driven learning across 54 standard benchmark environments. However, purely curiosity-driven learning might sometimes become infeasible or dangerous in real-world settings without other constraints. For example, in autonomous driving and human robot interaction, unexpected movements might occur and lead to catastrophe.

RL From Demonstration: Expert demonstrations have been introduced to guide the learning process. Hester et al. [22] proposed deep Q-learning from demonstrations (DQfDs) and stored the demonstrations in an experience replay buffer. In [3], demonstration data were used to pretrain the policy network. Ho and Ermon proposed generative adversarial imitation learning (GAIL) [11], which trains an RL policy by minimizing the Jensen–Shannon (J-S) divergence between the behavior policy and the expert policy, with generative adversarial training without accessing the environment reward. Although RL from demonstration has the potential to relieve the exploration dilemma in hard-exploration scenarios, existing algorithms tend to require a tremendous amount of high-quality data but often fail to fully leverage the value of the demonstrations. One better way of utilizing expert demonstration that can potentially reduce the amount of demonstration data is through reward shaping, as presented in [23]. A potential function represented by multivariate Gaussian was generalized from the demonstration, which can be integrated as the intrinsic reward. We derive the demonstration reward differently, by calculating the J-S entropy instead of Gaussian between two policies, eliminating the need of tuning additional parameters.

Inverse Reinforcement Learning (IRL): IRL and inverse optimal control (IOC) have provided a set of algorithms to directly learn the reward functions from demonstrations, as in Ng and Russell [24], Pieter and Ng [25], Ziebart et al. [26], and Finn et al. [27]. However, it is difficult to make an IRL algorithm effective since: 1) IRL asks for a large number of high-quality expert demonstrations; 2) IRL is inherently under-defined, as different reward functions might result in similar behaviors; and 3) IRL is expensive to run as it requires the RL procedure in an inner-loop.

Reward Learning From Preference: A large amount of work has been conducted on RL from human preferences or ratings. Christiano et al. [28] explored learning objectives defined in terms of human preferences between pairs of trajectory segments, and demonstrated the effectiveness of the method on Atari games and simulated robot locomotion without access to extrinsic reward signals. In [29], expert demonstrations and trajectory preferences were combined, where a reward function was learned from the preferences and the demonstrations were used by a DQfDs algorithm. However, preference learning might struggle when learning tasks where qualified experts are not available. Furthermore, as environments become more complex, the number and quality of preferences required by an agent increase, making the learning process inefficient and, in some cases, intractable.

III. PRELIMINARIES

A. Markov Decision Process

In this work, the problems considered are under the standard Markov decision process (MDP) setting. An MDP is formalized by the tuple: \((S, A, r, p_0, T, \gamma)\), where \(S\) and \(A\) represent the state space and action space, \(r = r(s, a, s')\) is the reward function, \(p_0\) is the probability distribution of the initial state, \(T = T(s'|s, a)\) denotes the transition function of the environment, and \(\gamma \in (0, 1)\) is the discount factor. An agent interacts with the environment over time based on policy \(\pi(a|s)\), mapping state to action probability. At time step \(t\), the agent receives \(s_t\) from the state space \(S\), selects \(a_t\) from the action space \(A\) according to \(\pi(a_t|s_t)\), transits to the next state \(s_{t+1}\) based on \(T = T(s_{t+1}|s_t, a_t)\), and receives a scalar reward signal \(r_t = r(s_t, a_t, s_{t+1})\). The discounted return is \(R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}\), and expectation of \(R_t\) is usually evaluated to reflect performance of the policy \(\pi\)

\[
J(\pi) = \mathbb{E}_\pi \left[ r(s_t, a_t, s') \right] = \mathbb{E}_{(s_0, a_0, s_1, a_1, s_2, \ldots)} [R_t] \tag{1}
\]

where \((s_0, a_0, s_1, a_1, s_2, \ldots)\) is a trajectory generated from interaction with the environment. Correspondingly, the value function can be defined as \(V_\pi(s) = \mathbb{E}_{\pi} [R_t | s_t = s]\), the action value function is \(Q_\pi(s, a) = \mathbb{E}_{\pi} [R_t | s_t = s, a_t = a]\), and the
advantage function is \( A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s) \). The objective of RL algorithms is to discover the optimal policy that can maximize the expectation of discounted return \( \mathbb{E}_\pi[R_t] \).

B. Policy Gradient

Unlike value-based RL, the policy gradient methods directly model and optimize the policy \( \pi_\theta(a|s) \) parameterized by \( \theta \). The learning objective is defined as

\[
J(\pi_\theta) = \mathbb{E}_{\pi_\theta}[r(s, a, s')] = \sum_{s \in S} d^\pi(s)V_{\pi_\theta}(s) = \sum_{s \in S} d^\pi(s) \sum_{a \in A} \pi_\theta(a|s)Q_{\pi_\theta}(s, a) \tag{2}
\]

where \( J(\pi_\theta) \) can be used to measure the performance of policy \( \pi_\theta(a|s) \), and \( d^\pi(s) \) represents the stationary distribution of Markov chain for \( \pi_\theta(a|s) \). According to the policy gradient theorem

\[
\nabla_\theta J(\pi_\theta) = \sum_{s \in S} d^\pi(s) \sum_{a \in A} \nabla_\theta \pi_\theta(a|s)Q_{\pi_\theta}(s, a) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)Q_{\pi_\theta}(s, a)] \tag{3}
\]

\( \theta \) can be optimized via gradient ascent. To solve \( \nabla_\theta J(\pi_\theta) \), \( Q_{\pi_\theta}(s, a) \) should be computed. Normally, \( Q_{\pi_\theta}(s, a) \) can be approximated with methods like Monte-Carlo estimation (REINFORCE [30]), Temporal-Difference learning [31], or with an auxiliary critic model (actor–critic policy gradient [32]). Furthermore, to reduce variance, the advantage function \( A_{\pi_\theta}(s, a) \) is introduced to substitute \( Q_{\pi_\theta}(s, a) \), and hence, \( \nabla_\theta J(\pi_\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)A_{\pi_\theta}(s, a)] \).

IV. METHODOLOGY

With the widespread use and advancement of RL, the significance and difficulty of exploration and reward design have been highlighted. In real-world scenarios, the extrinsic reward signal is usually extremely sparse and hard to reshape, which affects the exploration efficiency. Introducing demonstrations or curiosity has proven to be effective in sparse reward settings. The demonstrations are often exploited in the following ways: 1) save it to the experience replay buffer; 2) pretrain the policy network; and 3) calculate an intrinsic reward function. Curiosity is deployed to encourage the agent to explore novel states or perform actions to reduce its uncertainty about the environment’s dynamics. In this work, to fully leverage demonstration data and curiosity, the above two ideas were combined to formulate a new policy gradient method, which can be boosted from both demonstration and curiosity (PGfDC). PGfDC is supposed to outperform existing methods since: 1) it requires a limited number of demonstrations; 2) it can guarantee superior exploration efficiency; and 3) it can imitate the expert and, meanwhile, achieve a high return. These properties are desired in areas like human–robot interaction, autonomous driving, and game AI.

The overall workflow of the proposed PGfDC algorithm is shown in Fig. 1. There are two submodules, namely: 1) the normal RL module and 2) the intrinsic reward learner module. For the RL module, the agent interacts with the environment and receives reward signals estimated by the current reward function \( \hat{r}_k \)

\[
\hat{r}_k = r_e + \lambda_d r_d^k + \lambda_c r_c^k \tag{4}
\]

where \( r_e \) is the original extrinsic reward function of the environment, \( r_d^k \) represents the intrinsic reward function learned from demonstrations at the \( k \)th iteration, \( r_c^k \) is the intrinsic reward function learned from curiosity at the \( k \)th iteration, and \( \lambda_d \) and \( \lambda_c \) are the corresponding weighting coefficients. The collected interaction data are stored as \( \{s, a, r, s'\} \), and are sent to the intrinsic reward learner. Within the intrinsic reward learner, the discriminator network is updated with pre-stored expert demonstrations and the interaction data, and the curiosity network is simultaneously optimized with gradients computed from the collected interaction data. Then, the reward function is updated to \( \hat{r}_k+1 \). The intrinsic reward learner can work synchronously or asynchronously [33] with the standard RL module.

A. Reward Learning From Demonstration

RL from demonstration has proved to be an efficient and intuitive way of transferring an expert’s knowledge and preferences to the agent. The agent can either infer a reward function from the demonstrations, as in inverse RL, or boost its exploration through a pretrained policy. However, existing methods usually ask for a tremendous amount of high-quality demonstration data while failing to fully leverage the data. To address these issues, the demonstrations are used to formulate

---

Image: Fig. 1. Workflow of the PGfDC algorithm.
an additional penalty term to the original learning objective in this work, measuring the Jensen–Shannon divergence between the current policy \( \pi_\theta(a|s) \) and the demonstration policy \( \pi_{E}(a|s) \). Specifically, the provided demonstrations are expressed as a set of trajectories of state–action pairs, \( D^E = \{t_0, t_1, \ldots, t_N\} \), where \( t_i = (s'_i, a'_i), (s'_i, a'_i), \ldots, (s'_i, a'_i) \), and \( D^E \) are generated from an implicit expert policy \( \pi_{E} \). Then, the reformulated learning objective is obtained

\[
\min_{\theta} \mathcal{L}(\pi_\theta) = -J(\pi_\theta) + \lambda_d \mathcal{D}_{IS}(\pi_\theta, \pi_{E})
\]

where \( \lambda_d \in (0, 1) \) is the weighting coefficient and \( \mathcal{D}_{IS}(\cdot, \cdot) \) denotes the Jensen–Shannon divergence. It is impossible to directly estimate \( \mathcal{D}_{IS}(\pi_\theta, \pi_{E}) \) as \( \pi_{E} \) is unknown, thus the concept of occupancy measure is introduced to approximate \( \mathcal{D}_{IS}(\pi_\theta, \pi_{E}) \).

Definition 1 (Occupancy Measure): Let \( \rho_{E}(s) : S \rightarrow \mathbb{R} \) denote the unnormalized distribution of state visitation by following policy \( \pi_{E} \) in the environment, \( \rho_{E}(s) = \sum_{\tau=0}^{\infty} y^\tau P(s_t = s|\pi_{E}) \), then the unnormalized distribution of state–action pairs \( \rho_{E}(s, a) = \rho_{E}(s, a|\pi_{E}) \) is termed occupancy measure of policy \( \pi_{E} \).

According to [34, Th. 2], \( \pi_{E} \) is the only policy whose occupancy measure is \( \rho_{E}, \) given that \( \rho_{E} \) is the occupancy measure for \( \pi_{E}(a|s) = \langle \rho_{E}(s, a)/[\sum_{a} \rho_{E}(s, a')] \rangle \). Therefore, the Jensen–Shannon divergence between \( \pi_\theta \) and \( \pi_{E} \) can be substituted by

\[
\mathcal{D}_{IS}(\pi_\theta, \pi_{E}) = \mathcal{D}_{IS}(\rho_{E}, \rho_{E})
\]

Zhang et al. introduced the risk measure [35, 36], which is defined as the concave utility of the state–action occupancy measure. Equation (6) can be viewed as a special case of risk measurement, where prior demonstrations are incorporated. However, unlike converting the original RL problem into its dual form and solving it by stochastic primal–dual policy gradient, we approach the problem with generative adversarial training. Bingyi et al. [37] derived a lower bound for \( \mathcal{D}_{IS}(\rho_{E}, \rho_{E}) \), which can be reformulated as

\[
\mathcal{D}_{IS}(\rho_{E}, \rho_{E}) \geq \max_{w} \mathbb{E}_{(s, a)\sim \rho_{E}} \left[ \log(D_w(s, a)) \right] + \mathbb{E}_{(s, a)\sim \rho_{E}} \left[ 1 - \log(D_w(s, a)) \right]
\]

where \( D_w(s, a) : S \times A \rightarrow (0, 1) \), which can be represented by a neural network model with \( w \) as the parameters. Actually, the right-hand side of (7) can be viewed as the learning objective of the discriminator in a generative adversarial network (GAN), with \( \pi_{\theta} \) working as the generator

\[
\mathcal{L}_D = \mathbb{E}_{\rho_{E}} \left[ \log(D_w(s, a)) \right] + \mathbb{E}_{\rho_{E}} \left[ 1 - \log(D_w(s, a)) \right].
\]

In order to train \( D_w(s, a) \), state–action pairs from \( \rho_{E} \) are labeled as true, while the state–action pairs generated by \( \rho_{E} \) are labeled as false. Substituting (7) into (5), the following learning objective is obtained:

\[
\min_{\theta} \max_{w} -J(\pi_\theta) + \lambda_d \left( \mathbb{E}_{(s, a)\sim \rho_{E}} \left[ \log(D_w(s, a)) \right] + \mathbb{E}_{(s, a)\sim \rho_{E}} \left[ 1 - \log(D_w(s, a)) \right] \right)
\]

which is equivalent to

\[
\min_{\theta} \max_{w} -\mathbb{E}_{\rho_{E}} \left[ r(s, a, s') \right] + \lambda_d \mathbb{E}_{\rho_{E}} \left[ \log(D_w(s, a)) \right] + \lambda_d \mathbb{E}_{\rho_{E}} \left[ 1 - \log(D_w(s, a)) \right].
\]

Furthermore, (10) can be reorganized as

\[
\min_{\theta} \max_{w} -\mathbb{E}_{\rho_{E}} \left[ r(s, a, s') \right] + \lambda_d \mathbb{E}_{\rho_{E}} \left[ \log(D_w(s, a)) - 1 \right] + \lambda_d \mathbb{E}_{\rho_{E}} \left[ \log(D_w(s, a)) \right].
\]

In (11), the original reward function is reshaped by \( \lambda_d \mathbb{E}_{\rho_{E}} \left[ \log(D_w(s, a)) \right] \), as the constant \(-1\) can be removed. Thus, based on (11), provided with an input tuple \( (s, a, s') \), the demonstration reward is given as

\[
r_{d}(s, a, s') = \log(D_w(s, a)).
\]

B. Reward Learning From Curiosity

Following [15], curiosity reward in PGfDC is related to the prediction error of the learned forward dynamics model, which measures the agent’s knowledge about the environment. It is broken down into three submodules: 1) feature embedding \( G_f \); 2) inverse model \( G_i \); and 3) the forward model \( G_f \). In \( G_f \), the input state \( s \in S \) is encoded as a feature vector \( F(s) \). Then, the feature vectors of two consecutive states, \( F(s_{t}) \) and \( F(s_{t+1}) \), are concatenated and fed into the inverse model \( G_i \) to generate prediction for the action \( \tilde{a}_t \) taken by the agent to move from \( s_t \) to \( s_{t+1} \). By learning the inverse dynamics model, the feature space only captures those changes in the environment related to the actions of our agent, and ignores the rest. Hence, \( G_e \) and \( G_i \) can be combined to formulate a joint model

\[
G_{ei} = G_{ei}(\tilde{a}_t | s_t, s_{t+1}; \theta_{ei})
\]

where \( \theta_{ei} \) is the network parameters and is optimized through minimizing \( \mathcal{L}_{ei}(\tilde{a}_t, a_t) \). If discrete actions are used, \( \mathcal{L}_{ei} \) can be cross-entropy, whereas if continuous actions are adopted, \( \mathcal{L}_{ei} \) can be mean squared error. For the forward model \( G_f \), feature vector \( F(s_t) \) and the corresponding action \( a_t \) are used as the input to predict feature vector \( \tilde{F}(s_{t+1}) \) of the state at the next time step

\[
G_f = G_f(\tilde{F}(s_{t+1}), F(s_t), a_t; \theta_f)
\]

where the network parameters \( \theta_f \) are optimized by minimizing the mean squared loss function \( \mathcal{L}_{f}(\tilde{F}(s_{t+1}), F(s_{t+1})) = (1/2)\|\tilde{F}(s_{t+1}) - F(s_{t+1})\|^2 \). In this work, \( \theta_{ei} \) and \( \theta_f \) are jointly updated and the loss functions \( \mathcal{L}_{ei} \) and \( \mathcal{L}_{f} \) are combined and formulated as

\[
\min_{\theta_{ei}, \theta_f} \mathcal{L}_{curiosity} = (1 - \beta)\mathcal{L}_{ei} + \beta \mathcal{L}_{f}
\]

where \( \beta \) is the weighting factor and \( \beta \in (0, 1) \). The training data are collected while the agent is interacting with the environment and is stored in the tuple \( (s_t, a_t, s_{t+1}) \). \( \mathcal{L}_{f} \) is used to calculate the curiosity reward, and a transformation function \( (e^x - 1)/(e^x + 1) \) is applied to \( \mathcal{L}_{f} \) to scale it to the range of \([0, 1] \). Therefore, given the input tuple \( (s, a, s') \), the curiosity reward is given as

\[
r_c(s, a, s') = \frac{\mathcal{L}_{f}(F(s'), F(s')) - 1}{\mathcal{L}_{f}(F(s'), F(s')) + 1}.
\]
Closely related to curiosity is another type of exploration-based bonus in Bayesian inference, which is defined by heuristic functions related to state visitation counts [13]. Such rewards encourage the agent to explore novel states, but they cannot infer meaningful information about the environment’s dynamics, and hence, they cannot predict the consequences of their own actions. Fig. 2 illustrates the workflow of the curiosity reward module.

C. Practical Algorithm

The proposed reward learning modules are integrated with the state-of-the-art policy gradient algorithm: PPO [17]. The policy parameter \( \theta \) is updated by maximizing the PPO objective, which contains the clipping loss, the value function loss, and an entropy bonus

\[
J(\pi_\theta) = \mathbb{E}[L_{CCLD}(\pi_\theta) - c_1L_{VF}(\pi_\theta) + c_2H(\pi_\theta)]
\] (17)

where \( L_{CCLD}(\pi_\theta) = \min(r(\pi_\theta)\tilde{A}_c, \text{clip}(r(\pi_\theta), 1 - \epsilon, 1 + \epsilon)\tilde{A}) \), and \( L_{VF}(\pi_\theta) = \text{MSE}(V_{\text{target}}(s_t) - \tilde{V}(s_t)) \). The advantage estimate \( \tilde{A} \) and the state value estimate \( \tilde{V} \) are computed using the current augmented reward \( \tilde{r} \), containing both extrinsic and intrinsic rewards as defined in (4) with interaction data \( \langle s, a, r, s' \rangle \) stored in \( D^G \). The demonstration reward \( r_d \) is updated by training the discriminator network according to the objective in (8), with both interaction data sampled from \( D^G \) and the prestored demonstration data sampled from \( D^E \). In order to obtain the curiosity reward \( r_c \), the curiosity network is simultaneously optimized with gradients computed from the interaction data sampled from \( D^P \), by minimizing the objective defined in (15). Then, the reward function is updated to \( \tilde{r}^{k+1} \). The intrinsic reward learner can work synchronously or asynchronously [33] with the standard RL module. In the asynchronous version, the main loop runs PPO training, with reward values retrieved from the intrinsic reward modules (discriminator and curiosity learner). In the meantime, the reward modules run periodically and output the latest values upon request.

In addition, to enhance the exploration efficiency at the earlier stage of training, while still guaranteeing steady improvement of the episode returns later on, we introduce an annealing strategy [38] on the reward weighting coefficients: \( \lambda_d \) and \( \lambda_c \)

\[
\lambda_i^{k+1} \leftarrow \lambda_i^k - \lambda_i^k \eta_i, i \in \{d, c\}
\] (18)

with annealing factor \( \eta \in [0, 1] \)

The details of synchronous and asynchronous PGfDC with PPO style training are summarized in Algorithms 1 and 2, respectively.

---

**Algorithm 1 Synchronous PGfDC With PPO**

1. **Input:** Expert demonstrations \( D^E \); initial policy parameters \( \theta^0_p \), discriminator parameters \( w^0 \) and curiosity parameters \( (\theta^0_e, \theta^0_f) \), initial weighting factor \( \lambda^0_d, \lambda^0_c \), annealing factor \( \eta \).

2. Initialize experience replay buffer \( D^G \) to store generated trajectories.

3. for \( k = 1, 2, \ldots \) do

   4. for \( l = 1, 2, \ldots \) do

      5. Collect set of trajectories \( D^l \) by running policy \( \pi(\theta^l_p) \) in the environment.

      6. Compute state value estimates, \( \tilde{V} \) and advantage estimates, \( \tilde{A} \) with current reward \( \tilde{r}^k \).

      7. Compute probability ratio, \( r(\pi^l_p) \).

      8. Update \( \theta^l_p \) by maximizing the PPO objective defined in (17).

   9. Store \( D^l \) into \( D^G \) in the tuple format \( \langle s_t, a_t, s_{t+1} \rangle \).

10. end for

11. for \( m = 1, 2, \ldots \) do

12. State-action pairs \( \langle s, a \rangle \) in \( D^G \) and \( D^E \) are labeled as 0 and 1, respectively.

13. Sample batches from \( D^E \) and \( D^G \).

14. Update \( w^m \) by maximizing the discriminator objective defined in (8).

15. end for

16. for \( n = 1, 2, \ldots \) do

17. Sample batches from \( D^G \).

18. Update \( \theta^m_e \) and \( \theta^m_f \) by minimizing the objective defined in (14).

19. end for

20. Update the reward function according to (4).

21. Annealing the reward weighting factor:

22. end for

---

Fig. 2. Reward learning from curiosity. The curiosity network in Fig. 1 is further illustrated here, which is composed of three submodules: the feature embedding network, the inverse model network, and the forward model network. The loss of the forward model \( L_f \) was used to calculate the curiosity reward \( r_c \) by applying a transformation.
Algorithm 2 Asynchronous PGfDC With PPO

1: **Input:** Expert demonstrations \( D^E \); initial policy parameters \( \theta_p^0 \), discriminator parameters \( w^0 \) and curiosity parameters \( (\theta_{ei}^0, \theta_f^0) \), initial weighting factor \( \lambda_{ei}^0, \lambda_f^0 \), annealing factor \( \eta \),
2: Initialize experience replay buffer \( D^G \) to store generated trajectories
3: **Process 1:** PPO
   4: **repeat**
   5: Collect set of trajectories \( D^x \) by running policy \( \pi(\theta_p) \) in the environment
   6: Compute state value estimates, \( \tilde{V} \) and advantage estimates, \( A \) with current reward \( \tau \), compute probability ratio \( r(\theta_p) \)
   7: Update \( \theta_p \) by maximizing the PPO-Clip objective defined in (17)
   8: Store \( D^x \) into \( D^G \) in the tuple format \((s_t, a_t, s_{t+1})\)
   9: **Periodically**
   10: \( \longrightarrow \) Request for the latest discriminator parameters \( w \) and \( r_d \) from Process 2
   11: \( \longrightarrow \) Request for the latest curiosity parameters \( (\theta_{ei}, \theta_f) \) and \( r_c \) from Process 3
   12: \( \longrightarrow \) Anneal the weighting factor and update the reward function according to (4)
   13: **until** True

14:
15: **Process 2:** discriminator learner
   16: **repeat**
   17: State-action pairs \( (s, a) \) in \( D^G \) and \( D^E \) are labeled as 0 and 1, respectively
   18: Sample batches from \( D^E \) and \( D^G \)
   19: Update \( w \) by maximizing the objective in (8)
   20: Upon request, return the latest discriminator parameters \( w \) and the demonstration reward \( r_d \)
   21: **until** True

22:
23: **Process 3:** curiosity learner
   24: **repeat**
   25: Sample a batch from \( D^G \)
   26: Update \( \theta_{ei} \) and \( \theta_f \) by minimizing the loss defined in (15)
   27: Upon request, return the latest curiosity parameters \( (\theta_{ei}, \theta_f) \) and the curiosity reward \( r_c \)
   28: **until** True

V. EXPERIMENTAL EVALUATION

In this section, performance of the proposed PGfDC algorithm was experimentally evaluated on the following aspects.
1) Given the limited number of demonstrations, can PGfDC guarantee superior exploration efficiency?
2) Considering the sparse extrinsic reward of the environment, can PGfDC guarantee a high return at convergence?
3) Given demonstrations from the expert, can PGfDC imitate the expert’s behavioral preference and meanwhile achieve high empirical return?

A. Environment Settings

In order to comprehensively investigate the performance of the proposed algorithm, the following environments have been designed and evaluated.
1) Three variants of the grid world environment [39] with extremely sparse rewards and discrete state/action spaces.
2) A soft continuum arm reaching environment [8] with sparse reward and continuous state/action spaces.
3) The BipedalWalker-v3 locomotion environment from OpenAI Gym [40] with continuous state/action spaces but dense rewards.

The three grid worlds are: 1) an empty grid world with a size of \( 14 \times 14 \), where the agent is required to reach for the terminal state \((14 \times 14 – \text{GridWorld}, \text{Fig. 3(a)})\); 2) a grid world environment of the size \( 14 \times 14 \), where the agent has to pick up a key first, and then opens a door with the key to reach for the terminal state \((14 \times 14 – \text{KeyWorld}, \text{Fig. 3(b)})\); 3) a grid world maze composed of four connected rooms, where each room has a size of \( 8 \times 8 \), the agent is required to navigate through the rooms and reach for the terminal state \([4 – \text{RoomMaze}, \text{Fig. 3(c)}]\).

For all of the three environments, a sparse extrinsic reward is given only when the agent reaches the terminal state

\[
r_e = 1 - 0.9 \frac{n_t}{N_{\text{max}}}
\]

where \( n_t \) is the number of time steps taken by the agent, and \( N_{\text{max}} \) denotes the maximum number of time steps.

The goal of the soft arm reaching task in Fig. 3(d) is to have the tip of the arm reach toward a target location with the desired orientation. We sparsify the original reward setting in [8] with

\[
r_e = g(S) + \begin{cases} 4.5, & n \leq 0.005 \text{ and } 0.05 < p < 0.1 \\ 6, & n \leq 0.005 \text{ and } p < 0.05 \end{cases}
\]

where \( g(S) \) indicates a big penalty term of \(-1000\) when \( NaN \) is detected in state/action, \( n \) and \( p \) are the squared distance and the orientation difference between the arm tip and the target, respectively. In other words, the agent receives a bonus reward when it is in the vicinity of the target, and otherwise, the reward is 0. For the BipedalWalker-v3 environment, the agent obtains a positive reward proportional to the distance walked on the terrain and a negative reward proportional to the torque applied.

For each environment, only one to three demonstrated trajectories were provided to the agent: \( \tau^i = \{s^i_t, a^i_t\}_{t=0}^{n} \), \( i \in \{1, 2, 3\} \). The PGfDC algorithm was compared with the following baselines: 1) a human expert; 2) PPO; 3) a random policy; 4) GAIL; and 5) advantage actor critic (A2C). In the following sections, implementation details of PGfDC are briefly overviewed, including the policy network, the discriminator, and the curiosity module.

B. Network Architectures

1) Policy Network: For the three variants of grid world environments, PPO, A2C, GAIL, and PGfDC shared the same policy network architecture, where the input image \( s_t \subseteq \mathbb{R}^{3 \times 7 \times 7} \) was flattened to \( s_t^{\text{flat}} \subseteq \mathbb{R}^{147} \) before being fed
into the network. The flattened state $s_{t}^{\text{flat}}$ was fed into two separate fully connected layers to predict the action probability distribution and the value function, where each fully connected layer had a size of 64 and a Tanh activation function after it. To get the action probability distribution, another fully connected layer with a size equivalent to that of the action space was implemented, followed by a softmax operation. On the other hand, an output layer with 1 hidden unit was used to predict the value function.

For the robotics environments, MLP policies were implemented for PPO, GAIL, and PGfDC, with two hidden layers of size 64 and Tanh activation functions.

2) Discriminator Network: The input action $a_{t}$ was passed through a 2-layer MLP with 16 and 8 hidden units, respectively, and they were activated by RELU to obtain the action feature vector $a'_{t}$. For the grid world environments, the input image $s_{t} \subseteq \mathbb{R}^{5 \times 7 \times 7}$ was flattened to $s_{t}^{\text{flat}} \subseteq \mathbb{R}^{147}$ before being fed into the network. The flattened state $s_{t}^{\text{flat}}$ was fed into a 3-layer MLP with RELU activations and 256, 128, and 64 hidden units correspondingly to obtain the state feature vector $s'_{t}$.

For the robotics environments, the input states were passed through a two-layer MLP with a hidden size of 128 and RELU activation to obtain the feature vector $s'_{t}$. The action feature vector and the state feature vector were concatenated and passed through a fully connected layer with a size of 4, followed by a RELU activation. To predict the discriminator reward $r_{d}$, an output layer with one hidden unit and a sigmoid activation function was used. The learning rate of the discriminator was set to be $10^{-3}$.

3) Curiosity Network: Adapted from [15], architecture of the curiosity network is illustrated in Fig. 2. The curiosity module is made up of three parts: 1) the feature embedding $G_{e}$; 2) the inverse model $G_{i}$; and 3) the forward model $G_{f}$. The feature embedding for the grid world environments mapped the input states $s_{t}$ and $s_{t+1}$ into feature vectors $F(s_{t})$ and $F(s_{t+1})$ with a sequence of four convolution layers, each adopting the same filter number of 16 and kernel size of $3 \times 3$. A RELU activation function is used after each convolution layer. The output of the last convolution layer was flattened to generate a 32-dimensional feature vector. The feature embedding for the robotics environments adopted MLP structures with two hidden layers of size 64. For the inverse model, $F(s_{t})$ and $F(s_{t+1})$ were concatenated and passed through a fully connected layer with RELU activation and 64 hidden units, followed by an output layer activated by the sigmoid function to predict the action. In the forward model, the embedded feature vector $F(s_{t})$ and action $a_{t}$ were concatenated and fed into a fully connected layer with 128 hidden units and the RELU, followed by an output layer with 32 hidden units to predict the feature vector of $s_{t+1}$, $F(s_{t+1})$. For all the environments, the learning rate was set to be $10^{-3}$, and $\beta$ was $10^{-2}$.

C. Experimental Results

1) Evaluation With Grid-World Environments: Learning curves of the three Grid-World Environments in Fig. 3(a)–(c) are depicted in Fig. 4. Only one single demonstrated trajectory was provided to the agent in each environment. The hyperparameters of PGfDC as well as other experimental details are summarized in Table I. Fig. 4(a) illustrates the learning curves
of the environment 14 × 14 – GridWorld. The proposed PGfDC algorithm started to converge after about 2 × 10^4 time steps, PPO converged after approximately 9 × 10^4 time steps, while A2C and GAIL algorithms failed to converge within 15 × 10^4 time steps. Fig. 4(b) shows the learning curves in the environment 14 × 14 – KeyWorld, where PGfDC succeeded to converge at around 1.5 × 10^5 time steps, the PPO algorithm succeeded to converge after about 6 × 10^5 time steps, while A2C and GAIL still failed within 2 × 10^6 time steps. The learning curves for the environment 4 – RoomMaze are provided in Fig. 4(c). In this task, the proposed PGfDC started to converge at 2 × 10^5 time steps approximately, PPO converged after approximately 1 × 10^6 time steps, while A2C and GAIL algorithms failed to converge within 2 × 10^6 time steps. In addition, an ablation study has been performed by removing the demonstration reward term \( r_d \). Fig. 5 illustrates the comparison of PGfDC with and without demonstration information. Without demonstration, PGfDC degenerates to the method presented in [15], namely, PGfC. In all the tasks, the PGfDC with demonstration outperformed the one without demonstration.

2) Evaluation With Robotics Environments: In order to evaluate the capacity of PGfDC to boost exploration in challenging environments with high-dimensional spaces and sparse rewards, we implemented the algorithm in two robotic environments [Fig. 3(d) and (e)]. We compared the results with baseline PPO and strong baseline GAIL [11] (a widely adopted imitation learning algorithm that infers reward from demonstrations). Three demonstrations were collected for each environment, which were generated using PPO at convergence. The corresponding learning curves are shown in Fig. 6. The

![Figure 5](image1)

![Figure 6](image2)

**TABLE I**

| Hyperparameters Used by PGfDC |
|-----------------------------|
| **Case** | \( \gamma \) | \( \alpha \) | \( \lambda_{GAR} \) | \( \alpha_{entropy} \) | \( \alpha_{value} \) | \( clip \) | \( \lambda_c \) | \( \lambda_d \) | \( S \) | \( A \) | \( N_{max} \) | \( \eta \) |
| (a) | 0.99 | 10^{-3} | 0.95 | 10^{-2} | 0.5 | 0.2 | 10^{-2} | 10^{-3} | \( 3 \times 7 \times 7 \) | \( 7 \) | 1024 | NA |
| (b) | 0.99 | 10^{-3} | 0.95 | 10^{-2} | 0.5 | 0.2 | 10^{-4} | 10^{-3} | \( 3 \times 7 \times 7 \) | \( 7 \) | 1024 | NA |
| (c) | 0.99 | 10^{-3} | 0.95 | 10^{-2} | 0.5 | 0.2 | 10^{-4} | 10^{-3} | \( 3 \times 7 \times 7 \) | \( 7 \) | 1960 | NA |
| (d) | 0.99 | 10^{-3} | 0.95 | 10^{-2} | 0.5 | 0.2 | 10^{-1} | 10^{-2} | \( 8 \times 8 \) | \( 18 \) | NA | 1.4e^{-3} |
| (e) | 0.99 | 10^{-3} | 0.95 | 10^{-2} | 0.5 | 0.2 | 10^{-2} | 10^{-2} | \( 34 \) | \( 4 \) | NA | 4.3e^{-3} |
Fig. 7. Evaluation results of the trained policies. (a) Average return. (b) Average discriminator score.

curves are averaged over five runs with different random seeds for each algorithm.

Fig. 6(a) demonstrated that PGfDC can boost exploration even at an earlier stage of training than GAIL and PPO, and sustain higher mean episode returns without dropping significantly in this challenging environment with continuous spaces and sparse rewards. GAIL also achieves higher returns than baseline PPO, but the average return tends to fluctuate and introduce high variances when learning proceeds. This may be due to the fact that with a limited number of imperfect demonstration trajectories, the reward learner of GAIL easily overfits and converges to a local optimum, which leads to sudden drops in the episode return.

PGfDC and GAIL demonstrated similar and competitive performance in the dense reward scenario as shown in Fig. 6(b). In particular, both PGfDC and GAIL improved exploration efficiency and converged faster compared to PPO, but did not achieve significantly higher episode return at convergence than PPO.

3) Statistics Analysis: The above-mentioned experimental evaluations have validated that: 1) given limited demonstrated trajectories, the proposed PGfDC algorithm succeeded in convergence at much higher exploration efficiency compared with the A2C and PPO baselines and 2) PGfDC successfully achieved higher average returns when compared to A2C and PPO baselines, as well as outperforming strong baseline GAIL, particularly in difficult cases with both sparse rewards and continuous spaces. In order to investigate the third aspect of the proposed algorithm, which is: provided with demonstrations from the expert, can PGfDC imitate the expert’s behavior while achieving a high return at the same time, the $14 \times 14$ - GridWorld environment was used as the testbed. We provided a scoring method to quantify the similarity between the learned policy and the demonstration. Statistical results were provided to quantify these two aspects in Fig. 7.

To evaluate the average return of each policy, the learned policies were performed using 10 different random seeds, and for each seed, the average return was computed with ten independent rollout episodes. The evaluation results are shown in Fig. 7(a). In all of the three experiments, the proposed PGfDC algorithm achieved higher returns compared with A2C and GAIL. Specifically, the average returns of PGfDC were 0.973, 0.965, and 0.980 for the environments $14 \times 14$ - GridWorld, $14 \times 14$ - KeyWorld, and $4 \times 4$ - RoomMaze, respectively. In all three environments, neither A2C nor GAIL succeeded. A2C achieved 0.036, 0.006, and 0.006, while the average returns of GAIL were 0.039, 0.0, and 0.0, for the environments $14 \times 14$ - GridWorld, $14 \times 14$ - KeyWorld, and $4 \times 4$ - RoomMaze, respectively.

Two independent and distinct demonstrated trajectories were provided to the agent separately to facilitate two independent experimental runs, and thus, two discriminators have been learned with the demonstrations to determine whether the input sample came from the expert or not [based on (12)]. The experimental results are shown in Fig. 7(b), and the average discriminator score, $E_{\pi|\theta}[D_{\omega}(s, a)]$, was calculated to reflect the similarity between the expert and the policy. The evaluations were performed using ten different random seeds, and for each seed, the average discriminator score was computed with ten independent rollout episodes. Provided with demonstration-1, PGfDC achieved a score of 0.859 on average. As comparison, the discriminator scored the expert for 0.910, while A2C, GAIL, and PPO only received 0.121, 0.115, and 0.137, respectively, given that the random policy achieved an average discriminator score of 0.135. With demonstration-2, the proposed PGfDC algorithm achieved an average score of 0.550, the expert’s average score was 0.586, while A2C, GAIL, PPO, and the random policy received 0.162, 0.150, 0.156, and 0.194, respectively. Therefore, the experimental results have validated that the proposed algorithm has the potential to imitate experts while achieving a considerably high return.

VI. CONCLUSION AND FUTURE DIRECTIONS

A. Conclusion

Alongside the development of RL algorithms, reward shaping and exploration remain challenging for existing methods. An agent might struggle to discover useful information, especially when interacting with an environment where extrinsic feedback is sparse. An integrated algorithm, PGfDC, has been...
developed in this work with the purpose of boosting exploration and facilitating intrinsic reward learning from only a limited number of demonstrations. In PGfDC, the original reward function is reformulated by two additional terms, $r_d$ and $r_e$, where $r_d$ is the intrinsic reward learned from demonstrations with a discriminator network, and $r_e$ represents the intrinsic reward signal derived from curiosity. To comprehensively evaluate the performance of PGfDC, three grid world-like environments, one robotic locomotion environment, and one soft robotic arm reaching environment have been designed. For each environment, only one to three demonstrated trajectories were provided. Comparative studies were carried out to evaluate the proposed algorithm with baselines, including PPO, A2C, and GAIL. The experimental results validated that: 1) provided with a limited number of demonstrations, PGfDC can guarantee superior exploration efficiency; 2) in sparse reward scenarios, PGfDC can achieve high returns; and 3) PGfDC can imitate the expert’s behavioral preference and meanwhile, achieve high empirical return.

B. Future Directions

In theory, PGfDC is compatible with most policy gradient algorithms, so to further promote the application of PGfDC, one future direction will include the integration of our intrinsic reward learning modules to other state-of-the-art policy gradient methods.

Furthermore, the above-mentioned nice properties are especially desired in real-world sequential decision-making problems, such as large-scale game AI and autonomous driving, which are typical hard-exploration problems with even larger dimensions and sparser reward signals [3, 4, 41]. PGfDC has the potential to address these real-world problems and sufficiently reduce the training cost, with a faster convergence rate and limited human effort in reward shaping. Moreover, PGfDC has the potential of deriving competitive and human-like policies, which are usually desired in game AI and autonomous driving. So, another future direction is to deploy the algorithm to tackle complex real-world problems. As the complexity scales up, engineering effort is required to leverage the advantages of the asynchronous version of PGfDC and to develop distributed training platforms for stable deployment and efficient training of the algorithm.

Finally, the reward weighting coefficients $\lambda_d$ and $\lambda_e$ need to be properly adjusted for different environments and at different stages of training, which is difficult and nontrivial. Although we have introduced an annealing strategy, which alleviates the difficulty in some settings. We believe a systematic way of autotuning the reward weightings is required to fully demonstrate the superiority of our proposed method to baselines across various scenarios. Promising ways to treat this will be through multiobjective RL [42]–[44] and by building additional prediction models to guide parameter selection [45].

REFERENCES

[1] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction. Cambridge, MA, USA: MIT Press, 2018.

[2] V. Mnih et al., “Human-level control through deep reinforcement learning,” Nature, vol. 518, no. 7540, pp. 529–533, 2015.

[3] D. Silver et al., “Mastering the game of Go with deep neural networks and tree search,” Nature, vol. 529, no. 7587, pp. 484–489, 2016.

[4] C. Berner et al., “Dota 2 with large scale deep reinforcement learning,” 2019, arXiv:1912.06680.

[5] O. Vinyals et al., “Grandmaster level in starcraft II using multi-agent reinforcement learning,” Nature, vol. 575, no. 7782, pp. 350–354, 2019.

[6] A. Perrussil and W. Yu, “Neutral H 2 control using continuous-time reinforcement learning,” IEEE Trans. Cybern., early access, Nov. 24, 2020, doi: 10.1109/TCYB.2020.3028988.

[7] Y.-C. Liu and C.-Y. Huang, “DDPG-based adaptive robust task control for aerial manipulators with decoupling approach,” IEEE Trans. Cybern., early access, Feb. 2, 2021, doi: 10.1109/TCYB.2021.3049555.

[8] N. Naughton, J. Sun, A. Tekinpal, T. Parthasarathy, G. Chowdhary, and M. Gazzola, “Elastica: An compliant mechanics environment for soft robotic control,” IEEE Robot. Autom. Lett., vol. 6, no. 2, pp. 3389–3396, Apr. 2021.

[9] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, “Overcoming exploration in reinforcement learning with demonstrations,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), 2018, pp. 6292–6299.

[10] A. Singh, L. Yang, K. Hartikainen, C. Finn, and S. Levine, “End-to-end robotic reinforcement learning without reward engineering,” 2019, arXiv:1904.07854.

[11] J. Ho and S. Ermon, “Generative adversarial imitation learning,” in Proc. 30th Int. Conf. Neural Inf. Process. Syst., 2016, pp. 4572–4580.

[12] S. Ross, G. J. Gordon, and D. Bagnell, “A reduction of imitation learning and structured prediction to no-regret online learning,” in Proc. 14th Int. Conf. Artif. Intell. Stat., 2011, pp. 627–635.

[13] J. Z. Kolter and A. Y. Ng, “Near-Bayesian exploration in polynomial time,” in Proc. 26th Annua. Int. Conf. Mach. Learn., 2009, pp. 513–520.

[14] D. Russo and B. V. Roy, “Learning to optimize via information-directed sampling,” Oper. Res., vol. 66, no. 1, pp. 230–252, 2018.

[15] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, “Curiosity-driven exploration by self-supervised prediction,” in Proc. IEEE Comput. Vis. Pattern Recognit. Workshops, 2017, pp. 488–489.

[16] B. Fuglede and F. Topsøe, “Jensen-shannon divergence and hilbert space embedding,” in Proc. Int. Symp. Inf. Theory (ISIT), 2004, p. 31.

[17] J. Schulman, F. Wolski, P. Dhariwal, and O. Klimov, “Proximal policy optimization algorithms,” 2017, arXiv:1707.06347.

[18] J. Schulman, S. Levine, P. Moritz, M. Jordan, and P. Abbeel, “Trust region policy optimization,” in Proc. Int. Conf. Mach. Learn., 2015, pp. 1889–1897.

[19] J. Schmidhuber, “A possibility for implementing curiosity and boredom in model-building neural controllers,” in Proc. Int. Conf. Simulat. Adapt. Behav. Animals Animats, 1991, pp. 222–227.

[20] B. C. Stadie, S. Levine, and P. Abbeel, “Incentivizing exploration in reinforcement learning with deep predictive models,” 2015, arXiv:1507.00814.

[21] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros, “Large-scale study of curiosity-driven learning,” 2018, arXiv:1808.04355.

[22] T. Hester et al., “Deep Q-learning from demonstrations,” in Proc. 32nd AAAI Conf. Artif. Intell., 2018, pp. 3223–3230.

[23] T. Beys, A. Harutyunyan, H. B. Suay, S. Chernova, M. E. Taylor, and A. Nowé, “Reinforcement learning from demonstration through shaping,” in Proc. 24th Int. Joint Conf. Artif. Intell., 2015, pp. 3352–3358.

[24] A. Y. Ng and S. J. Russell, “Algorithms for inverse reinforcement learning,” in Proc. 7th Int. Conf. Mach. Learn. (ICML), 2000, pp. 663–670.

[25] A. Pieter and A. Y. Ng, “Apprenticeship learning via inverse reinforcement learning,” in Proc. 21st Int. Conf. Mach. Learn., vol. 1, 2004, pp. 1–8.

[26] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, “Maximum entropy inverse reinforcement learning,” in Proc. 23rd AAAI Conf. Artif. Intell., vol. 8, 2008, pp. 1433–1438.

[27] C. Finn, S. Levine, and P. Abbeel, “Guided cost learning: Deep inverse gradient following algorithms for connectionist reinforcement learning,” Mach. Learn., vol. 8, nos. 3–4, pp. 229–256, 1992.
[31] G. Tesauro, “Temporal difference learning and TD-gammon,” Commun. ACM, vol. 38, no. 3, pp. 58–68, 1995.
[32] V. R. Konda and J. N. Titsiklis, “Actor-critic algorithms,” in Proc. Adv. Neural Inf. Process. Syst., 2000, pp. 1008–1014.
[33] V. Mnih et al., “Asynchronous methods for deep reinforcement learning,” in Proc. Int. Conf. Mach. Learn., 2016, pp. 1928–1937.
[34] S. Umar, M. H. Bowling, and R. E. Schapire, “Apprenticeship learning using linear programming,” in Proc. 25th Int. Conf. Mach. Learn., 2008, pp. 1032–1039.
[35] J. Zhang, A. S. Bedi, M. Wang, and A. Koppel, “Cautious reinforcement learning via distributional risk in the dual domain,” IEEE J. Sel. Areas Commun., vol. 2, no. 2, pp. 611–626, Jun. 2021.
[36] J. Zhang, A. Koppel, A. S. Bedi, C. Szepesvari, and M. Wang, “Variational policy gradient method for reinforcement learning with general utilities,” 2020, arXiv:2007.02151.
[37] K. Bingyi, Z. Jie, and J. Feng, “Policy optimization with demonstrations,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 2474–2483.
[38] M. Jing et al., “Reinforcement learning from imperfect demonstrations under soft expert guidance,” in Proc. AAAI Conf. Artif. Intell., 2020, pp. 5109–5116.
[39] M. Chevalier-Boisvert, L. Willems, and S. Pal, “Minimalistic Gridworld Environment for OpenAI gym,” 2018. [Online]. Available: https://github.com/maximecb/gym-minigrid
[40] G. Brockman et al., “OpenAI gym,” 2016, arXiv:1606.01540.
[41] D. Ye et al., “Mastering complex control in MOBA games with deep reinforcement learning,” 2019, arXiv:1912.09729.
[42] X. Chen, A. Ghabdulzadeh, M. Björkman, and P. Jensfelt, “Meta-learning for multi-objective reinforcement learning,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), 2019, pp. 977–983.
[43] A. Pan, W. Xu, L. Wang, and H. Ren, “Additional planning with multiple objectives for reinforcement learning,” Knowl. Based Syst., vol. 193, no. 5, 2020, Art. no. 105392.
[44] K. Li, T. Zhang, and R. Wang, “Deep reinforcement learning for multiobjective optimization,” IEEE Trans. Cybern., vol. 51, no. 6, pp. 3103–3114, Jun. 2021.
[45] J. Xu, Y. Tian, P. Ma, D. Rus, S. Sueda, and W. Matusik, “Prediction-guided multi-objective reinforcement learning for continuous robot control,” in Proc. Int. Conf. Mach. Learn., 2020, pp. 10607–10616.

Jie Chen received the B.Eng. degree in mechanical and automation engineering from Soochow University, Suzhou, Jiangsu, China, in 2013, and the Ph.D. degree in robotics and machine learning from the University of Hong Kong, Hong Kong, in 2017.

From 2017 to 2018, he worked as a Postdoctoral Research Fellow with Harvard University, Cambridge, MA, USA, and the Brigham and Women’s Hospital, Boston, MA, USA. Since 2018, he has been working as a Senior Research Scientist of Machine Learning and Reinforcement Learning with Momenta and Tencent, Shenzhen, China. He is currently a Technical Lead with ByteDance, Beijing, China. His research interests include planning and decision making in game AI, autonomous driving, and robotics.

Wenjun Xu received the B.Eng. degree in electrical engineering and automation from Soochow University, Suzhou, China, in 2012, the M.Phil. degree in mechanical and automation engineering from the Chinese University of Hong Kong, Hong Kong, in 2014, and the Ph.D. degree in medical robotics with the National University of Singapore, Singapore, in 2019.

She is currently a Research Scientist with Peng Cheng Laboratory, Shenzhen, China. Her research interests include control and modeling of flexible/soft robots, machine learning, reinforcement learning, and their applications in robotics.