Deep intrinsically motivated exploration in continuous control

Baturay Saglam1 - Suleyman S. Kozat1

Received: 24 March 2022 / Revised: 9 March 2023 / Accepted: 27 June 2023 / Published online: 26 October 2023
© The Author(s), under exclusive licence to Springer Science+Business Media LLC, part of Springer Nature 2023, corrected publication 2023

Abstract
In continuous control, exploration is often performed through undirected strategies in which parameters of the networks or selected actions are perturbed by random noise. Although the deep setting of undirected exploration has been shown to improve the performance of on-policy methods, they introduce an excessive computational complexity and are known to fail in the off-policy setting. The intrinsically motivated exploration is an effective setup and hyper-parameter alternative to the undirected strategies, but they are usually studied for discrete action domains. In this paper, we investigate how intrinsic motivation can effectively be combined with deep reinforcement learning in the control of continuous systems to obtain a directed exploratory behavior. We adapt the existing theories on animal motivational systems into the reinforcement learning paradigm and introduce a novel and scalable directed exploration strategy. The introduced approach, motivated by the maximization of the value function’s error, can benefit from a collected set of experiences by extracting useful information and unify the intrinsic exploration motivations in the literature under a single exploration objective. An extensive set of empirical studies demonstrate that our framework extends to larger and more diverse state spaces, dramatically improves the baselines, and outperforms the undirected strategies significantly.

Keywords Deep reinforcement learning · Exploration · Intrinsic motivation · Actor-critic algorithms · Continuous control

1 Introduction
The balance between exploration and exploitation is a protracted and key challenge in modern deep reinforcement learning (RL) (Dayan & Sejnowski, 1996; Kearns & Singh, 2002). The main aim of the exploration is to ensure that agents collect a set of diverse
experiences not to adopt a premature behavior. If the exploration is insufficient, action decisions that yield high rewards can be missed, and the policy of an agent can converge to a local optimum. In contrast, with excessive exploration, agents may spend too much time and waste resources in trying many suboptimal actions without efficiently using the collected experiences (Meuleau & Bourgine, 1999). Therefore, an optimal balance for the exploration and exploitation trade-off, also known as the exploration-exploitation dilemma, must be well found through effective strategies (Moore & Atkeson, 1993; Thrun, 1992). However, designing an effective and efficient exploration structure is not trivial as it cannot be inferred from the reward function of the underlying Markov decision process (MDP), and high-dimensional state and action spaces increase the required time and resources for the exploration (Thrun, 1992).

Exploration in RL is usually studied in directed and undirected methods (Thrun, 1992). Undirected exploration techniques in high dimensional action spaces aim to perturb selected actions or weights of parameterized policies, where well-known examples are additive Gaussian noise (Williams, 1992) or deep parameter space noise (Fortunato et al., 2018; Plappert et al., 2018; Zhang & Van Hoof, 2021). Even though the deep strategies have advantages of being consistent, structured, and generalizable due to the exploration in parameter space conditioned on the states, they introduce excessive computational cost and heavily rely on randomness (Thrun, 1992). While undirected methods aim to inject randomness into the agents’ action decisions, directed strategies rely on extracting useful information from the collected experiences in a rule-based manner. Although these approaches require lower-level exploration to collect experiences and are shown to be effective, they are usually employed in hard exploration tasks where rewards are delayed, deceptive, or sparse (Zhang & Van Hoof, 2021).

As a counterpart to the directed exploration strategies in hard exploration tasks, intrinsic motivation has been effectively used from three aspects: prediction error, state novelty, and information gain (Barto, 2013). The first motivation leads the agents towards the state spaces where the prediction is difficult. The second intuition is to add an intrinsic bonus when the agent goes into a state which it does not usually go, and the latter is an intrinsic reward depending on the reduction of uncertainty on the environment dynamics. Although intrinsically motivated exploration methods are shown to perform well, they suffer from several limitations (Barto, 2013). These limitations include the generalizability to different domains in that they only focus on low- or high-dimensional environments and are mostly studied for discrete action spaces. Furthermore, the usage of representation learning may prevent the long-term control from being included in the exploration. Thus, undirected methods remain the only available exploration techniques in continuous control (Zhang & Van Hoof, 2021). However, they are known to suffer from off-policy learning due to the separation of the exploration from the control (Singh et al., 2000). When the evaluation of the exploration strategy is not on par with the policy update, undirected methods may not be an informative exploratory guide to the agents.

In this paper, we introduce a novel approach to the exploration-exploitation dilemma through the maximization of the temporal difference (TD) error that adapts the existing theories on animal motivational systems to the deep reinforcement learning for continuous control. We introduce a scalable deep directed high-frequency perturbation algorithm. This unified framework combines the motivations of prediction error, state novelty, and information gain by modeling an exploration policy that prevents agents from repeating actions through an additive learnable exploration direction. Our empirical studies on MuJoCo (Todorov et al., 2012) and Box2D (Parberry, 2013) continuous control tasks demonstrate that a learned internally motivated exploration policy outperforms the undirected
exploration strategies in terms of stability, learning speed, and highest evaluation returns by expanding to useful state spaces. The main contributions of this paper are as follows:

- We adapt the existing theories on animal motivational systems into the deep reinforcement learning paradigm to constitute an intrinsically motivated directed exploration strategy for continuous action spaces. We introduce a novel and scalable algorithm that directs agents to less-known state spaces with high value error through an additive explorer network.
- By maximizing the TD error, our method forces the approximate value function to learn the values of unseen or less-known states or state-action pairs and prevent premature behavior before sufficiently learning the environment. This motivation also benefits from off-policy learning by extracting useful information from the collected experiences.
- Our technique is readily applicable to actor-critic methods that employ TD learning (Precup et al., 2001; Sutton, 1988). We also show that the introduced framework comes with the cost of training a single network and optimizing a single hyper-parameter.
- We support our experimental results with theoretical analysis and prove that our algorithm has a faster convergence property than undirected strategies.
- Through an extensive set of experiments in challenging OpenAI Gym (Brockman et al., 2016) continuous control tasks, we show that our algorithm improves the performance of the baseline algorithms and attains significantly higher evaluation returns in fewer time steps than the undirected strategies in the majority of the tasks tested.
- To ensure reproducibility, we report precise experimental and implementation details, and our code and results are available at the GitHub repository.¹

2 Related work

Thrun (1992) classified exploration into directed and undirected strategies. Well-known methods for exploration in the control of continuous systems are random action selection with a certain probability (Whitehead & Ballard, 1991), additive Gaussian noise (Williams, 1992), and Ornstein-Uhlenbeck noise process (Uhlenbeck & Ornstein, 1930). However, they result in poor state space coverage due to the random walk behavior (Thrun, 1992). Alternative mechanisms include the employment of posterior distributions (Lee et al., 2019), dimension reduction (Nouri & Littman, 2010) or sampling the policy parameters from a search distribution (Kearns et al., 2002). Such parameter-space noise techniques can improve the exploratory behavior over the high-perturbation counterparts by reducing the variance when combined with actor-critic methods. Another alternative to the independent action perturbations is the correlation of the noises applied to the subsequent actions (Singh, 1992). Although these techniques have been proven to be successful in exploring parameter space, their use is restricted since they are solely tested with shallow (usually linear) function approximators and operate only in the low-dimensional state and action spaces (Zhang & Van Hoof, 2021).

To overcome the limited applicability of these approaches, the deep exploration techniques of NoisyNet (Fortunato et al., 2018) and Parameter Space Noise for Exploration (PSNE) (Plappert et al., 2018) were proposed. These approaches learn a distribution over

¹ https://github.com/baturaysaglam/DISCOVER.
policies for trajectory-based exploration in parameter space. One way to sample exploring policies is by perturbing the weights throughout all layers of a deep neural network. However, this sampling process introduces uncertainty, which can be addressed using Monte Carlo integration. Nonetheless, several factors limit the performance of these methods. First, trajectory-based strategies can be ineffective as only a single exploration strategy is evaluated per trajectory, which may be stuck at the local optima. Secondly, Monte-Carlo integration introduces a large variance in the gradient estimates. In addition, they are computationally intensive and heavily rely on randomness.

Considering the drawbacks of NoisyNet and PSNE, Zhang and Van Hoof (2021) proposed the state-of-the-art Deep Coherent Exploration (Coherent) algorithm. While NoisyNet and PSNE learn a parameter-space distribution over policies for trajectory-based exploration, Coherent generalizes the step-based and trajectory-based exploration to perturb the last layer of the policy networks. The framework further treats the last layer parameters of the policy network as latent variables. It incorporates a recursive inference step in the policy update process to manage these variables in a way that is efficient and can handle large amounts of data. Although Coherent learns to explore more efficiently compared to NoisyNet and PSNE, they altogether evaluate the exploration strategy only when the exploration is performed. This causes them to fail in the off-policy setting since the exploration is separated from learning, as our empirical results show.

As an alternative to the undirected methods in continuous control, Xu et al. (2018) introduced Meta-Policy Gradient that adapts the meta-learning to the exploration problem to obtain a learnable directed exploratory behavior through a student-teacher framework for the Deep Deterministic Policy Gradient (DDPG) algorithm (Lillicrap et al., 2016). Furthermore, the intrinsic motivation has been effectively employed for exploration under the standard MDP setting yet for discrete action domains (Kearns & Singh, 2002; Koenig & Simmons, 1996). Therefore, our comparative evaluations consider the directed exploration strategy of Meta Policy Gradient in the experiments with DDPG, and the undirected deep exploration techniques Deep Coherent Exploration, NoisyNet, and PSNE.

3 Background

3.1 Reinforcement learning

Reinforcement learning considers an agent that interacts with its environment to solve a sequential decision-making task. At every discrete time step $t$, the agent observes a state $s \in S$, and selects an action $a \in A$, where $S$ and $A$ are the state and action spaces, respectively. Depending on its action decision, the agent receives a reward $r$ from a reward function, and observes a next state $s' \in S$. In fully observable environments, the RL problem is represented by a finite Markov decision process, a tuple $(S, A, p, \gamma)$ where $p$ is the transition dynamics such that $s', r \sim p(s, a)$, and $\gamma \in [0, 1)$ is the constant discount factor. The objective in reinforcement learning is to find an optimal policy $\pi$ that maximizes the value defined as the expected cumulative discounted returns $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i+1}$, where $\gamma$ prioritizes the short term rewards. Policy of an agent $\pi$ is stochastic if it maps states to action probabilities, $\pi : S \rightarrow \mathcal{P}(A)$, or deterministic if it maps states to unique actions, $\pi : S \rightarrow A$. 

占有率 Springer
In continuous action domains, the maximum of the value $R_t$ over possible actions is intractable. Therefore, an agent’s policy is represented by a separate function that selects actions on the observed states. In deep reinforcement learning, such a function is approximated by actor networks. Algorithms that employ actor networks are referred to as actor-critic methods, and they enable the control of continuous systems through the employed actor network. In actor-critic algorithms, actor networks $\pi_\theta$ parameterized by $\theta$ are optimized by gradient ascent over the policy gradient $\nabla_\theta J(\theta)$ computed by a policy gradient algorithm.

### 3.2 On-policy temporal difference learning

On-policy learning requires learning from experiences collected by the currently followed policy. For a given policy $\pi$, the state-value function estimates the expected return in state $s$ while following the current policy $\pi$:

$$V^\pi(s) = \mathbb{E}_{s_i \sim \pi} [R_i | s_i].$$

In deep reinforcement learning, the state-value functions are modeled by differentiable function approximators $V_\psi(s)$ parameterized by $\psi$. The deep state-value network is learned by TD learning (Precup et al., 2001; Sutton, 1988), an update rule based on the Bellman equation (Bellman, 1957) which represents a fundamental relationship used to learn the state-value function by bootstrapping from the value estimate of the current state $s$ to the subsequent state $s'$:

$$V_\psi(s) = r + \gamma \mathbb{E}_{s'} [V_\psi(s')].$$

### 3.3 Off-policy temporal difference learning

In off-policy learning, agents collect transitions defined by the tuple $(s, a, r, s')$ and store them into the experience replay buffer (Lin, 1992). The collected experiences are then reused multiple times to perform gradient steps on the deep policies and value functions to improve data and sampling efficiency. There exists an action-value function $Q^\pi(s, a) = \mathbb{E}_{s_i \sim \pi, a_i \sim \pi} [R_i | s, a]$, associated with the policy $\pi$, also known as the critic or Q-function, which represents the expected return while following the policy after taking action $a$ in state $s$.

Similar to the state-value functions, the deep setting of off-policy reinforcement learning approximates action-value functions by deep neural networks $Q_\theta$ with parameters $\theta$. Then, the deep action-value functions is learned through TD learning by bootstrapping from the estimate of the current state-action pair $(s, a)$ to the subsequent state-action pair $(s', a')$, also known as Q-learning (Watkins & Dayan, 1992):

$$Q_\theta(s, a) = r + \gamma \mathbb{E}_{s' \sim \pi', a' \sim \pi'} [Q_{\theta'}(s', a')]; \quad a' \sim \pi_{\theta'}(s'),$$

where $\theta'$ and $\phi'$ are the parameters of the secondary frozen networks, $Q_{\theta'}(s, a)$ and $\pi_{\phi'}(s)$, to maintain stability and fixed objective over high-frequency updates. In practical applications of deep RL, the next action chosen by the behavioral or target policy network in the TD learning is perturbed by random action noise to realize the exploration in the next state, known as the target policy smoothing regularization (Fujimoto et al., 2018):
where \( N(0, \sigma) \) is a zero-mean Gaussian random variable with standard deviation \( \sigma \).

The weights of the target networks can be updated through the soft or hard update. In the soft update, parameters are updated with a small proportion \( \tau \) at each time step, \( \theta' \leftarrow \tau \theta + (1 - \tau)\theta' \), while hard update considers periodically updating the weights to exactly match the weights of the current or behavioral network.

4 Deep directed exploration motivated by the temporal difference error

Motivation is a term that describes the processes which impact arousal, strength, and action direction (Ryan & Deci, 2000). To be motivated is to feel compelled to do an action. Psychologists distinguish motivation into two classes, extrinsic motivation, taking action because of externally supplied rewards, and intrinsic motivation, doing something for its own sake as it is inherently enjoyable or exciting (Ryan & Deci, 2000).

There is a conformity between the animal motivation and RL systems in that both aim to maximize external rewards (Barto & Simsek, 2005). The remarkable resemblance between the critical components of RL and the activity of dopamine neurons (Schultz, 1998) play an essential role in animal motivational systems (Barto & Simsek, 2005). The core of computational models of the dopamine function lies in identifying the challenges that neural reward systems must overcome, including the critical task of selecting vital actions necessary for survival (McClure et al., 2003b). The TD learning algorithm (Precup et al., 2001; Sutton, 1988), used for planning actions to obtain future rewards, underlies the prediction error explanation of dopaminergic neuron responses (McClure et al., 2003b). The hypothesis that the rate of dopamine neuron spiking encodes a TD prediction error signal explains a large amount of neuronal recording data (Montague et al., 1996; Schultz, 1998; Schultz et al., 1997). Furthermore, this model also accounts for dopamine concentrations in the brain during electrical stimulation (Garris et al., 1999; Kilpatrick et al., 2000) and accurately interprets activity changes in animal brain reward structures measured using functional magnetic resonance imaging (Berns et al., 2001; McClure et al., 2003a; O’Doherty et al., 2003; Pagnoni et al., 2002).

In particular, dopamine serves two primary functions in TD learning. First, it serves as a learning signal for the state- or action-value to enable the prediction of future rewards. Second, dopamine release directs action selection toward situations that are expected to result in a reward (McClure et al., 2003b). Dopamine either indirectly impacts action selection by aiding in learning value predictions or directly influences action selection weights, i.e., policy weights. Positive error signals can increase the value attributed to a state or action, thereby increasing the likelihood of choosing it in the future. Dopamine release may also have more immediate effects on action selection. The prediction error signal indicates whether to continue with a particular course of action or switch to another based on whether the reward received is better or worse than expected (Dayan, 2002; Montague et al., 1996).

Consequently, following these discussions, we propose that dopamine neurons can be considered as an RL agent’s state- or action-value function. Furthermore, as dopamine neuron spiking encodes the TD prediction error, an RL agent in continuous action spaces can be motivated by consistently maximizing the prediction error of its value function. Constantly forcing
the agent to choose actions that result in large value error can determine whether the corresponding reward will be better or worse than expected, similar to dopamine release in the animal brain. Therefore, a more systematic encouragement of exploratory behaviors can be achieved, as shown in our extensive simulations. We summarize the TD error based intrinsic motivation for on- and off-policy actor-critic algorithms in Remark 1 and 2, respectively.

**Remark 1** (Intrinsic motivation for exploration in on-policy learning) For on-policy learning in continuous action spaces, a reinforcement learning agent can be motivated by a consistent maximization of the prediction error by its state-value function \( V(s) \).

**Remark 2** (Intrinsic motivation for exploration in off-policy learning) For off-policy learning in continuous action spaces, a reinforcement learning agent can be motivated by a consistent maximization of the prediction error by its action-value function \( Q(s, a) \).

### 4.1 On-policy intrinsically motivated exploration

To construct our directed and intrinsically motivated exploration framework, we first consider a separate, deep deterministic exploration policy \( \xi_\omega : S \rightarrow \mathcal{E} \), with parameters \( \omega \), that maps states to unique exploratory directions. Moreover, we assume that the state-value function estimates the value of a state by also considering the exploration directions that are in the subspace of the state space \( \mathcal{E} \in S \). Similarly, the policy also considers the exploration directions while selecting actions. Thus, the exploration policy can influence the distribution of the states received by the state-value function and policy under the parameters \( \omega \). Let \( V^*(s, \eta) \) be the optimal state value for a given state \( s \) and exploration direction \( \eta \) under the optimal policy \( \pi^*(\cdot|s, \eta) \), defined by the Bellman equation:

\[
V^*(s, \eta) = r + \gamma V^*(s', \eta') |_{\eta' = \lambda \xi_\omega (s')},
\]

where \( \eta' \) is the exploration direction in the next step and \( \lambda \) is a regularization term for the exploration policy not to detrimentally perturb the state distribution. By Remark 1, intrinsically motivated exploration can be considered as an adversarial game such that the state-value function tries to minimize the prediction error while the exploration policy aims to perturb the state distribution so that prediction error is maximized. Therefore, the objective for the joint optimization of the state-value and exploration networks is expressed by:

\[
\max_{\omega} \min_{\psi} \sum_{s' \sim p_x} \| V^*(s, \eta) - V^*_\psi (s, \eta) \|_{\eta = \lambda \xi_\omega (s')}^2.
\]

However, the knowledge of the optimal state-value function beforehand is not possible in practical applications of deep reinforcement learning. A common approach in on-policy learning is to regard empirical reversed sum of rewards \( \hat{R} \), i.e., rewards-to-go, as the fixed objective for the state-value function, defined by:

\[
\hat{R}_t = \sum_{i=i}^{T} R(s_i, a_i, s_{i+1})
\]
where $T$ is the length of the horizon for which rollout transitions are collected to train the agent. Using (7) and considering the exploration directions, the objective for the state-value network is expressed by:

$$J(\psi) = \| \hat{R} - V_\psi(s, \eta) \|_{\eta=\lambda \cdot \xi_\omega(s)}^2. \quad (8)$$

Then, exploration policy’s objective is in the inverse direction of the state-value function’s objective:

$$J(\omega) = -J(\psi),$$
$$= -\| \hat{R} - V_\phi(s, \lambda \cdot \xi_\omega(s)) \|^2,$$
$$:= -\Phi(s, \lambda \cdot \xi_\omega(s)). \quad (9)$$

Ultimately, the objective of the exploration policy is to perturb the state distribution so that the prediction error by the state-value network is maximized. This leads agents to state spaces where state value prediction is difficult and hence, allows the correction of unknown or less visited states’ values. Furthermore, the exploration network can be updated through the Deterministic Policy Gradient (DPG) Theorem (Silver et al., 2014) using the constructed objective (9):

$$\nabla_\omega J(\omega) = \mathbb{E}_{s \sim p_s} [\nabla_\eta \Phi(s, \eta)]_{\eta=\lambda \cdot \xi_\omega(s)} \nabla_\omega \xi_\omega(s)]. \quad (10)$$

Note that the existence of the explorer network’s gradient follows the Deterministic Policy Gradient Theorem. During the evaluation, however, no exploration is performed. Hence, the state-value and actor networks should not consider any direction that perturbs the observed states’ distribution. This can be achieved by simply replacing the exploration directions with zeros. We refer to the resulting on-policy variant of our algorithm as On-Policy Deep IntrinSCally MOtivated ExploRation (On-Policy DISCOVER), and provide a pseudocode in Algorithm 1.

**Algorithm 1 On-Policy Deep Directed Intrinsically Motivated Exploration (On-Policy DISCOVER)**

1. Initialize the explorer network $\xi_\omega$ with parameters $\omega$
2. for each rollout time step do
3. Observe state $s$
4. Obtain the exploration direction: $\eta = \lambda \cdot \xi_\omega(s)$
5. Select an action considering the exploration direction: $a = \pi_\phi(s, \eta)$
6. Receive reward $r$ and observe next state $s'$
7. Store the transition tuple $(s, a, \eta, r, s')$ into the rollout buffer
8. end for
9. for each policy update step do
10. Update $\omega$ by the deterministic policy gradient and mini-batch learning on the rollouts: $\nabla_\omega J(\omega) = N^{-1} \sum \nabla_\eta \Phi(s, \eta)]_{\eta=\lambda \cdot \xi_\omega(s)} \nabla_\omega \xi_\omega(s)$, where $N$ is the mini-batch size
11. Update policy and state-value networks by considering the sampled exploration directions $\eta$
12. end for
4.2 Off-policy intrinsically motivated exploration

For off-policy learning, now consider that the exploration directions are in a subspace of the action space $\mathcal{E}_\omega : \mathcal{S} \rightarrow \mathcal{E} \in \mathcal{A}$. We additionally initialize a secondary frozen target explorer network with parameters $\omega'$, to obtain a fixed objective and stability in the explorer network updates, similar to the Deep Q-learning algorithm (Mnih et al., 2015). The objective of the exploration policy is to maximize the TD error of the Q-networks in consistency with the observed states and actions selected by the policy. First, let the actions chosen by the policy be perturbed by the explorer networks:

$$\tilde{a} = a + \lambda \cdot \xi_\omega(s),$$  \hfill (11)

$$\tilde{a}' = \pi_{\phi'}(s') + \lambda \cdot \xi_{\omega'}(s'),$$  \hfill (12)

where $\lambda$ now regularizes the additive exploration direction for the actions. Note that the action in the next state may also be chosen by the behavioral policy such as in the Soft Actor-Critic algorithm (Haarnoja et al., 2018). Furthermore, to realize exploration in the next state, target policy smoothing regularization is obtained through the target explorer network in (12), which is shown to improve the performance of off-policy methods (Fujimoto et al., 2018).

As discussed in Remark 2, TD error should be kept maximum by considering the selected actions. Thus, parameters of the exploration policy can be updated through gradient ascent over the error by the Q-network. This constitutes the loss function for the exploration policy in the form of an implicit adversarial game:

$$J(\omega) = -\|y - Q_\psi(s, \tilde{a})\|^2,$$  \hfill (13)

where $\bar{\delta}$ is the off-policy TD error under the perturbed actions $\tilde{a}$. Then, the exploration policy is again updated through the DPG algorithm:

$$\nabla_\omega J(\omega) = \mathbb{E}_{s \sim p_s, a \sim \pi_\phi(s)} [\nabla_\eta \bar{\delta}(s, a + \eta)]_{\eta=\lambda \cdot \xi_\omega(s)} \nabla_\omega \xi_\omega(s).$$ \hfill (14)

Moreover, to apply the target policy smoothing regularization in the Q-network updates and maintain the exploration in the action-value estimates, we use the actions in the current and next state that are perturbed by the behavioral and target explorer networks, respectively, as expressed in (12). This allows the directed exploration to be maintained in the critic updates:

$$J(\theta) = \|y - Q_\psi(s, \tilde{a})\|^2.$$ \hfill (15)

This forms the off-policy variant of DISCOVER, which is summarized in Algorithm 2. In the next section, we explain how DISCOVER can be effectively combined with actor-critic algorithms to control continuous systems. Moreover, we perform a comprehensive theoretical comparison of the complexity of the introduced directed exploration scheme and undirected exploration strategies.
Algorithm 2 Off-Policy Deep Directed Intrinsically Motivated Exploration (Off-Policy DISCO VER)

1: Initialize the explorer network $\xi_\omega$ with parameters $\omega$
2: Initialize the target explorer network $\omega' \leftarrow \omega$
3: for each exploration time step do
4:  Choose an action $a$
5:  Perturb the selected action $\tilde{a} = a + \lambda \cdot \xi_\omega(s)$
6:  Observe reward $r$ and next state $s'$
7:  Store transition tuple $(s, a, \tilde{a}, r, s')$ into the replay buffer
8: end for
9: for each Q-network update step do
10:  Perturb the next action through the target explorer network to modify
11:  the fixed objective for Q-learning: $\tilde{a}' = a' + \lambda \cdot \xi_{\omega'}(s')$
12: end for
13: for each policy update step do
14:  Update $\omega$ by the deterministic policy gradient through the sampled
15:  batch of $N$ transitions: $\nabla_\omega J(\omega) = N^{-1} \sum_i \nabla_\eta \delta(s, a + \eta) |_{\eta = \lambda \cdot \xi_\omega(s)} \nabla_\omega \xi_\omega(s)$
16: end for
17: for each target policy update period do
18:  Update the target explorer network: $\omega' \leftarrow \tau \omega + (1 - \tau) \omega'$
19: end for

4.3 Actor-critic with deep directed intrinsically motivated exploration

Although the introduced exploration strategy is based on the prediction error in the TD
learning, the consistent maximization of the TD error unifies the intrinsic motivations
in the directed exploration as discussed in Remarks 3, 4, and 5 for the prediction error,
state novelty and information gain incentives, respectively.

Remark 3 (Prediction error) The TD error maximization objective leads the approximate
state- and action-value functions to learn the transitions with high TD error where pre-
dicting states or state-action tuples is difficult. This is performed in on-policy learning by
perturbing the distribution under the observed states or actions chosen by the policy for
off-policy learning.

Remark 4 (State novelty) If the agent repeatedly visits the same state space in state novelty,
the intrinsic reward corresponding to that subspace decreases. Hence, a state space may be
repeatedly visited in undirected methods due to randomness (Thrun, 1992). However, in
the deterministic TD error maximization objective, the agent is constantly forced to visit
the tuples with the highest TD error. Thus, the agent always seeks novel state spaces with-
out any possibility for a repetition in the action selection.

Remark 5 (Information gain) High TD error implies the uncertainty on environment
dynamics as the expected sum of rewards in the corresponding state space, and the under-
lying MDP is not well-known by the value function (Sutton et al., 2018). The explorer
network aims to reduce such uncertainty by learning the value of uncertain states or state-
action pairs through the additive exploration direction.

As discussed, the designed algorithm is formulated as a max-min optimization problem,
in which the value functions can become more robust in states where they experience high
uncertainty, and a safer value prediction is achieved. Notably, the potential performance improvement offered by our contributions is very similar to how Generative Adversarial Networks (GAN) (Goodfellow et al., 2014) operate. The introduced framework adversarially attacks the actions selected by the greedy policy to challenge the critic as if the generator in a GAN tries to fake the discriminator. Conversely, the critic is designated to learn the values of the perturbed states or actions to reduce uncertainty about them, just as the discriminator learns to classify fake images. Hence, the presented max-min optimization has the same motivation on which GANs are based, and the power and convergence of this approach have been previously shown by Goodfellow et al. (2014).

Furthermore, for off-policy learning, the intrinsically motivated exploration can also consider the agent’s history of learning process by training from the batch of transitions sampled from the experience replay buffer. However, as discussed, the separated updates of exploration strategy and policies suffer from off-policy learning (Singh et al., 2000). Therefore, the exploration policy should be evaluated and updated whenever the policy of the underlying actor-critic algorithm is updated. This yields that the exploration framework is synchronized to the agent’s policy update, with having the same update frequency and update rule of the target networks, i.e., soft or hard update. Furthermore, the exploration policy should be identically structured with the agent’s policy when combined with both on- and off-policy algorithms. The motivation of this imitation is that the exploration policy should be fairly as capable as the agent’s policy. If the agent’s policy can learn to control the environment through a policy gradient algorithm, the exploration policy can also learn to maximize the TD error with DPG under the same set of hyper-parameters and update structure, which we show through empirical studies in the next section. This requirement is highlighted in Remark 6.

Remark 6 When combined with actor-critic algorithms, the structure of the exploration policy should follow the exact architecture and hyper-parameter setting of the underlying algorithm’s policy, e.g., learning rate, size and depth of the actor network, update period, and utilization of target networks.

The large computational complexity remains as a detrimental drawback in the undirected methods (Thrun, 1992). We investigate the time complexity of the introduced exploration strategy in Theorem 1 under ergodic deterministic MDPs. Moreover, Lemma 2 remarks the time complexity of any exploration strategy under ergodic non-deterministic MDP setting. We then perform a time complexity comparison of our approach versus undirected strategies in Corollary 1, from which we derive that the time to find an optimal policy under the exploration motivated by maximum TD error is always less than or equal to the undirected exploration. In the following section, we show through an extensive set of simulations that DISCOVER exhibits remarkable results in on- and off-policy settings, outperforming the state-of-the-art exploration techniques. Furthermore, we demonstrate that only a single hyper-parameter choice is necessary for a wide range of tasks when the explorer network is identically structured and synchronized to the agent’s actor network, validating Remark 6.

Lemma 1 (Whitehead’s theorem on undirected exploration complexity in deterministic MDPs) Under deterministic Markov decision process conditions, the expected time to find the goal state using undirected exploration is bounded below by an expression exponential in the state dimension $n$. 
**Proof** See Whitehead (1991).

**Lemma 2** (Time complexity of the exploration under ergodic stochastic MDPs) For each \( n \in \{2, 3, \ldots \} \), there is an ergodic non-deterministic Markov decision process, at which the time complexity for reaching the goal state under exploration is exponential in \( n \), even if the optimal policy is given in advance.

**Proof** See Thrun (1992).

**Theorem 1** (Time complexity of the exploration motivated by the maximum temporal difference error under ergodic deterministic MDPs) With exploration directed by the maximum temporal difference error, the number of actions required to find an optimal policy in any finite ergodic deterministic Markov decision process with a goal state is bounded above by \( \mathcal{O}(d^2 m + m^2 n^2) \), given the number of unknown states or state-action pairs \( m \), and time complexity for each iteration in the temporal difference learning \( \mathcal{O}(d) \).

**Proof** Follows from the extension of Theorem 3 in Thrun (1992) to the exploration motivated by the maximum TD error. The TD learning update rule at iteration \( t \) following the current policy \( \pi \) is expressed for on- and off-policy learning, respectively, as:

\[
V_{t+1}^\pi(s) = r + \gamma V_t^\pi(s'), \tag{16}
\]

\[
Q_{t+1}^\pi(s, a) = r + \gamma \max_{a'} Q_t^\pi(s', a'). \tag{17}
\]

First, the equations imply that \( V^\pi(\cdot) \) and \( Q^\pi(\cdot) \) are monotonically increasing in the learning time. Then, consider an optimal path \( \langle s_1 = s_{\text{init}}, s_1, s_2, \ldots, s_k = s_{\text{goal}} \rangle \) from some initial state \( s_{\text{init}} \) to a goal state \( s_{\text{goal}} \). If \( V^\pi(s_i) \) is correct for some state \( s_i \) on this path \( (0 < i \leq k) \), then applying the TD learning in state \( s_{i-1} \) will assign the correct state-value to \( s_{i-1} \) (Thrun, 1992). This is also valid for Q-learning denoted by (17), specifically, if \( Q^\pi(s_i, a_i) \) is correct for some state-action tuple \( (s_i, a_i) \) on this path \( (0 < i \leq k) \), then applying the TD learning at state-action pair \( (s_{i-1}, a_{i-1}) \) will assign the correct action-value to \( (s_{i-1}, a_{i-1}) \) (Thrun, 1992).

Learning the environment will identify the correct value function and hence, an optimal policy if the goal state exists. The exploration strategy in interest will direct the agent to the unknown state space with the highest TD error and apply the TD learning update rule. Therefore, the value of a unique state or state-action pair will be corrected in each exploration run since always the one with the highest TD error is visited. If there are \( m \) unknown entities, i.e., state or state-action pair, then after \( m \) iterations, the optimal policy is found. However, learning the environment requires identifying the action model, i.e., the outcome of each action. This can be done with at most \( \mathcal{O}(d^2 mn^2) \) actions (Thrun, 1992). Furthermore, by Theorem 1 shown by Thrun (1992), the complexity of each exploration run is bounded above by \( \mathcal{O}(mn^2) \). Thus, the resulting worst-case complexity for an exploration directed by the maximum TD error under ergodic deterministic MDPs is then in \( \mathcal{O}(d^2 m + m^2 n^2) \).

**Corollary 1** In any finite ergodic Markov decision process, the expected time to find an optimal policy using exploration directed by the maximum temporal difference error is bounded above the expected time using undirected exploration.
The proof infers from Lemmas 1, 2, and Theorem 1. The polynomial or quadratic time is bounded above by the exponential time. Hence, by Lemma 1 and Theorem 1, the TD error based exploration is bounded above by undirected exploration for ergodic deterministic MDPs. From Lemma 2, we know that both maximum TD error based directed exploration and undirected exploration are bounded below by the time complexity exponential in the state space dimension for ergodic stochastic MDPs. Thus, exploration directed by maximum TD error has a time complexity that is either the same or less than undirected exploration’s time complexity for any MDP, in the worst-case scenario.

5 Experiments

5.1 Experimental details

We perform experiments to demonstrate the effectiveness of our method and compare it with the existing exploration strategies. The current approaches for exploring continuous action spaces in actor-critic methods rely on undirected strategies (Zhang & Van Hoof, 2021). Therefore, along with greedy action selection and standard Gaussian action noise (Williams, 1992), we also consider the available undirected frameworks: NoisyNet (Fortunato et al., 2018), PSNE (Plappert et al., 2018), and the state-of-the-art Deep Coherent Exploration (Coherent) (Zhang & Van Hoof, 2021). We evaluate each method in combination with on-policy actor-critic algorithms, Advantage Actor-Critic (A2C) (Mnih et al., 2016), Proximal Policy Optimization (PPO) (Schulman et al., 2017), and off-policy actor-critic algorithms, Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2016), Soft Actor-Critic (SAC) (Haarnoja et al., 2018) and Twin Delayed Deep Deterministic Policy Gradient (TD3) (Fujimoto et al., 2018). We also consider the directed strategy Meta-Policy Gradient (Xu et al., 2018) in our experiments with DDPG. We focus on challenging MuJoCo (Todorov et al., 2012) and Box2D (Parberry, 2013) continuous control tasks interfaced by OpenAI Gym (Brockman et al., 2016). Algorithm-specific adaptations of DISCOVER can be found in our code (see footnote 1).

Our implementation of NoisyNet is based on the code from authors’ GitHub repository.2 We refer to the authors’ implementation in OpenAI Baselines3 (Dhariwal et al., 2017) for PSNE, and original papers for Deep Coherent Exploration and SAC. In addition, authors’ GitHub repository4 is used to implement the fine-tuned TD3 and baseline DDPG. Finally, we implement the on-policy algorithms A2C and PPO through the well-known GitHub repository5 with tuned hyper-parameters for the continuous control tasks in OpenAI Gym. DDPG and SAC follow the tuned hyper-parameters provided by OpenAI Baselines3 Zoo6 (Raffin, 2020). For the action space noise, we use a fixed zero-mean Gaussian distribution with a standard deviation of 0.1. Competing algorithms closely follow the hyper-parameter settings outlined in the original papers. For the parameter-space noise algorithms, we initialize the parameter noise at 0.017 for on-policy and 0.034 for off-policy

---

2 https://github.com/Kaixin/NoisyNet-A3C.
3 https://github.com/openai/baselines.
4 https://github.com/sfujim/TD3.
5 https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail.
6 https://github.com/DLR-RM/rl-baselines3-zoo.
algorithms as they were found to produce the best results in. In addition, we use $\beta = 0.01$ for all environments in the Deep Coherent Exploration algorithm, and set the mean-squared error threshold in PSNE to 0.1.

As discussed in Remark 6, we initialize our exploration framework to exactly match the policy network’s structure and hyper-parameter setting for all baselines. This initialization includes the size and depth of the networks, optimizer, learning rate, frequency of the updates, and weight decay if it exists. In addition, we employ target explorer networks in off-policy algorithms for which the underlying actor-critic methods’ target network update rule, i.e., soft or hard update, and learning rate also apply to the target explorer network. This also yields the delayed policy updates in the TD3 algorithm to apply to the explorer network. We consider six values for the exploration regularization term $\lambda = \{0.0, 0.1, 0.3, 0.6, 0.9, 1.0\}$, where we use $\lambda = 0.1$ for On-Policy DISCOVER and $\lambda = 0.3$ for Off-Policy DISCOVER. Exact experimental setup and hyper-parameter settings are found in Appendix 1.

Each algorithm is run for 1 million time steps with evaluations every 1000 steps. Each evaluation reports the average cumulative return over 10 episodes in a distinct evaluation environment without exploration and updates. We report the results over 10 random seeds of the OpenAI Gym simulator, network initialization, and dependencies. The environment dynamics, state and action spaces, and reward functions are not pre-processed and modified for easy reproducibility and fair evaluation procedure.

5.2 Comparative evaluation and discussion

5.2.1 On-policy learning

Learning curves under the on-policy algorithms A2C and PPO are reported in Figs. 1 and 2, respectively. Additional results for the InvertedDoublePendulum-v2, InvertedPendulum-v2, Reacher-v2 environments are found in Appendix 2.1. Note that the default exploration used in these algorithms is the inherent entropy maximization of the actions produced by the stochastic policy. As the actions are sampled from the Gaussian policy, a maximized entropy yields diverse actions and thus, more efficient exploration. From an overall perspective, we observe that On-Policy DISCOVER consistently improves the baseline algorithms and either matches or outperforms the competing approaches in all tasks regarding the convergence rate and highest evaluation returns. In some of the environments, it is demonstrated that the performance of our method matches the baseline. These environments, e.g., InvertedDoublePendulum, InvertedPendulum, and Reacher, are regarded as trivial in that they do not require a comprehensive exploration or complex learning approaches (Henderson et al., 2018). However, we observe a crucial performance improvement in more challenging tasks and high dimensional environments such as Ant, HalfCheetah, Hopper, Humanoid, Swimmer, and Walker2d. Therefore, we infer that when the underlying algorithm fails at learning, the effectiveness of the employed exploration strategy becomes prominent.

For the competing approaches, although our implementation of NoisyNet and PSNE follow the authors’ implementation and tuned hyper-parameters, we notice that they usually underperform the baseline, while Deep Coherent Exploration exhibits a considerable performance. It was previously shown by Zhang and Van Hoof (2021) that NoisyNet and PSNE fail in the continuous setting as they were initially proposed for the standard Atari games (Bellemare et al., 2013) in which action spaces are discrete, as discussed in Sect. 2.
Furthermore, although the Coherent exploration was shown to exhibit better performance in the on-policy setting due to the synchronized policy and exploration strategy updates, our approach outperforms it by a notable margin. Finally, On-Policy DISCOVER exhibits an improved but suboptimal behavior in environments when the baseline does not converge such as A2C in Reacher, InvertedPendulum, InvertedDoublePendulum, and PPO in HalfCheetah, Humanoid. Naturally, when the off-policy method cannot solve the environment sufficiently, the performance improvement offered by DISCOVER does not cover optimal policies.

5.2.2 Off-policy learning

Evaluation results for the off-policy setting are reported in Figs. 3, 4, and 5 for the DDPG, SAC, and TD3 algorithms, respectively. Complete evaluation results are provided in Appendix 2.1. The results demonstrate that Off-Policy DISCOVER substantially improves
the baselines in the majority of the tasks. Specifically, in high-dimensional environments, Ant and Humanoid, which are considered challenging tasks (Henderson et al., 2018), DISCOVER significantly accelerates the learning while noise-based methods have a slight or no improvement. In addition, DDPG is known exhibit a random behavior in high-dimensional environments (Henderson et al., 2018) and Swimmer. Similarly, TD3 and SAC cannot surpass the local optima in Swimmer. We observe in these cases that the competing algorithms have a high dependence on the underlying algorithm, i.e., performance improvement is considerable only when the baseline performs well. Our method, in contrast, can break this correlation to an extent and converge to an improved policy.

We observe that DISCOVER has a faster convergence to higher evaluation rewards for the rest of the environments and off-policy algorithms. In comparison, the undirected methods either barely improve the baseline or closely follows the no exploration setting in terms of the learning speed and attained evaluation returns. Although the undirected algorithms are previously shown to have a considerable impact on the on-policy algorithms

![Fig. 2 Evaluation curves for the set of MuJoCo and Box2D continuous control tasks under the PPO algorithm. The shaded region represents half a standard deviation of the average evaluation return over 10 random seeds. A sliding window of size 5 smoothes curves for visual clarity](image-url)
such as A2C and PPO, they exhibit a poor improvement in the off-policy setting. In fact, for some of the environments, they underperform the Gaussian action noise. This poor performance is due to the asynchronous exploration and policy updates in the off-policy learning (Singh et al., 2000). The undirected methods only evaluate and update their deep exploration strategy when the exploration is performed. The decoupled updates of the agent’s policy and exploration strategy cannot lead agents to useful state spaces. Conversely, our method updates its exploration policy on par with the agent’s policy update in an off-policy fashion, which results in an apparent and significant improvement in the off-policy setting, verifying Remark 6.

Moreover, Meta-Policy Gradient notably improves the DDPG algorithm compared to the action and parameter-space noise techniques. This shows the effectiveness of the directed exploration over the undirected methods in off-policy learning. Nevertheless, the highest returns are attained by DISCOVER in our experiments with DDPG. This is due to the additional time duration spent on the teacher network’s data collection in the

Fig. 3 Evaluation curves for the set of MuJoCo and Box2D continuous control tasks under the DDPG algorithm. The shaded region represents half a standard deviation of the average evaluation return over 10 random seeds. A sliding window of size 5 smoothes curves for visual clarity.
Meta-Policy Gradient algorithm. In comparison, Fig. 3 shows that our method exhibits a more robust and better performance as it simultaneously learns to explore with the agent’s policy.

Although DISCOVER is shown to attain higher returns in all tasks, for stable environments where the number of time steps is fixed such as HalfCheetah, our method shows a slight advantage over the competing methods. This is due to the smaller exploration degree requirement in the stable environments as stated by Henderson et al. (2018). In contrast, the performance improvement in unstable environments is more considerable as undirected exploration strategies can cause sudden failures in off-policy learning (Henderson et al., 2018). Moreover, we still observe an improved but suboptimal behavior when the underlying off-policy algorithm does not converge to the optimal policy, mainly under the DDPG algorithm. As discussed, it is natural for an exploration strategy not to cover optimal policies when the baseline is incapable of sufficiently solving the environment. Nonetheless, our empirical studies reflect the psychological approach to the exploration-exploitation dilemma, that is, informative and directed guidance through the gradient of
the evaluation function prevents the boredom of the agents to encourage the exploration, which validates Remarks 1 and 2. Thus, we conclude that the introduced directed exploration scheme can overcome the limitations induced by the random walk behavior in the undirected approaches.

5.2.3 Delayed reward setting

We conduct experiments to evaluate the performance of DISCOVER in a delayed reward setting. The delayed setting used is the same as that described by Zheng et al. (2018). This setting accumulates rewards for $D$ time steps, where $D$ is the number of steps for which the reward is accumulated, before providing it to the agent. The agent receives the sum of rewards over $D$ steps at the end of the $D$th step or at the end of the episode, whichever comes earlier. We choose PPO and TD3 (with conventional Gaussian action noise) as baselines for on- and off-policy learning, respectively. To test the performance, we use the Walker2d environment, which is a more trivial task than most others, as shown in Figs. 2

![Evaluation curves for the set of MuJoCo and Box2D continuous control tasks under the TD3 algorithm. The shaded region represents half a standard deviation of the average evaluation return over 10 random seeds. A sliding window of size 5 smoothes curves for visual clarity](image)

---

**Fig. 5** Evaluation curves for the set of MuJoCo and Box2D continuous control tasks under the TD3 algorithm. The shaded region represents half a standard deviation of the average evaluation return over 10 random seeds. A sliding window of size 5 smoothes curves for visual clarity.
and 4. By testing in this environment, we aim to observe the performance drop due to delayed rewards more clearly. We limit the delay steps to $D = \{2, 5, 10\}$ as our focus in this study is mainly on learning from dense rewards rather than delayed ones, and larger values of $D$ would make the task more challenging and less interpretable. The results are reported in Figs. 6, 7, and 8.

As shown in Figs. 6 and 7, both PPO and TD3 show a significant performance drop in the delayed setting compared to the dense reward setting. This is due to the sparsity and delay introduced in the reward signal, which makes it more difficult for the agents to learn an optimal policy and explore the state-action space. The non-stationarity of the Walker2d environment also adds to this challenge. The agents must connect the delayed rewards to the actions they took $D$ time steps earlier, which is difficult and leads to poor policy performance. When $D = 5$, the algorithms fail to find a decent policy. When $D = 2$, they converge slowly, and when $D = 10$, they do not converge at all.

Although our algorithm is not specifically designed for the delayed reward setting, it still improves over PPO and TD3 in a structured manner, indicating its effectiveness in learning from sparse reward signals. We attribute this improvement to the fact that DISCOVER is biologically more plausible than standard RL algorithms such as PPO and TD3. DISCOVER’s psychological basis, which forces the critic to learn unknown or less-known states or state-action pairs, can provide an effective and directed exploration strategy even

---

**Fig. 6** Evaluation curves for the selected delayed reward settings under the PPO algorithm. The shaded region represents half a standard deviation of the average evaluation return over 10 random seeds. A sliding window of size 5 smoothes curves for visual clarity.

**Fig. 7** Evaluation curves for the selected delayed reward settings under the TD3 algorithm. The shaded region represents half a standard deviation of the average evaluation return over 10 random seeds. A sliding window of size 5 smoothes curves for visual clarity.
in delayed reward settings. As a real-world example, animals can still learn from delayed reinforcements even if they do not immediately observe the outcomes of their actions.

However, from Fig. 8, we observe that the improvement offered by DISCOVER starts to diminish as the number of delay steps $D$ increases. For instance, the final rewards DISCOVER achieves are insufficient when $D = 10$ and result in a suboptimal policy. In such cases, further reward manipulation, e.g., the addition of intermediate or intrinsic reward signals, may be directions for further research such as the study by Zheng et al. (2018), which integrates intrinsic motivation into the delayed reward settings. In summary, our results suggest that DISCOVER can be a promising direction for exploration in reinforcement learning, particularly in settings where exploration is critical to finding an optimal policy. However, further analysis and experimentation will be required to fully understand its strengths and weaknesses and its applicability in different environment settings such as partial state observability, and sparse or binary rewards.

5.3 Ablation studies

We perform ablation studies to analyze the effects of the components: exploration direction regularization $\lambda$ and delayed exploration policy updates, target explorer network, and target smoothing regularization in the off-policy TD learning. As delayed policy updates, target policy networks, and target smoothing regularization are employed in the TD3 algorithm, we conduct our ablations on TD3 for a comprehensive study. Therefore, the resulting performance of Off-Policy DISCOVER is analyzed when removing or altering each of these components. Moreover, we provide the results for On-Policy DISCOVER under the PPO algorithm for the considered $\lambda$ values. The effect of the rest of the DISCOVER components is not available in the on-policy setting since the policy updates are not delayed, and the target network is not employed in on-policy algorithms.

As the significance of each component may vary from task to task, we consider four environments with different characteristics in our ablation studies. We choose the low-dimensional environment Hopper, high-dimensional environment Humanoid, and
environments that vastly require on- and off-policy samples to be solved, Swimmer and HalfCheetah, respectively (Henderson et al., 2018). Ablation results are reported in Table 1. Unless otherwise stated, \( \lambda = 0.3 \) is used for Off-Policy DISCOVER. Learning curves for the ablation studies are given in Appendix 2.2.

Since larger values of \( \lambda \) increase the exploration, \( \lambda = 1 \) indicates a full exploration while \( \lambda = 0 \) represents the greedy action selection. From Table 1, we observe that after around \( \lambda = 0.3 \), the performance of Off-Policy DISCOVER dramatically decreases due to the highly perturbed action selections, which prevents agents from learning the environment. The exploration becomes insufficient for smaller \( \lambda \) and yields suboptimal policies. This is similar for the on-policy setting for which after around \( \lambda = 0.1 \), the underlying distribution of the observed state is highly perturbed, which degrades the learning efficiency. In addition, \( \lambda = 0.0 \) results in greedy action selection under which the agent is stuck at a suboptimal policy.

| Setting | HalfCheetah-v2 | Humanoid-v2 | Hopper-v2 | Swimmer-v2 |
|---------|----------------|-------------|-----------|------------|
| **On-policy** |               |             |           |            |
| \( \lambda = 0.0 \) | 3179.35       | 700.42      | 3126.34   | 127.85     |
| \( \lambda = 0.1 \) | **5224.88**   | **1010.48** | **3436.32** | **134.25** |
| \( \lambda = 0.3 \) | 2500.32       | 634.86      | 1890.07   | 122.57     |
| \( \lambda = 0.6 \) | 2621.43       | 626.17      | 2140.21   | 90.26      |
| \( \lambda = 0.9 \) | 1822.65       | 528.45      | 1607.60   | 56.78      |
| \( \lambda = 1.0 \) | 1933.92       | 555.24      | 1581.94   | 74.23      |
| **Off-Policy** |               |             |           |            |
| \( \lambda = 0.0 \) | 8853.83       | 3041.59     | 4993.90   | 105.19     |
| \( \lambda = 0.1 \) | 8762.55       | 2704.40     | **5080.31** | 107.76     |
| \( \lambda = 0.3 \) | **9939.39**   | **3350.27** | 4941.76   | **133.94** |
| \( \lambda = 0.6 \) | 7955.40       | 2651.23     | 4719.90   | 88.81      |
| \( \lambda = 0.9 \) | 4594.68       | 795.43      | 393.91    | 43.13      |
| \( \lambda = 1.0 \) | 2732.76       | 253.12      | 203.45    | 25.92      |
| w/o DPU | 8900.64       | 2794.50     | 4924.07   | 105.55     |
| w/o TN  | 8756.69       | 3248.61     | 4463.95   | 115.48     |
| w/o TSR | 8795.56       | 2953.44     | 4857.31   | 113.02     |

Bold values represent the maximum for each environment under on- or off-policy setting. The PPO and TD3 algorithms are used as baselines for On-Policy and Off-Policy DISCOVER, respectively.
policy networks and verifies Remark 6. Hence, when applied to an actor-critic algorithm, DISCOVER should follow the exact setting of the policy framework. This results in a single hyper-parameter optimized, the exploration regularization term $\lambda$.

### 5.4 Visualization of the state visitations

To study the behavior of our exploration policy, we investigate the collected set of transitions throughout the training of Off-Policy DISCOVER under the TD3 algorithm. We focus on off-policy learning for this study since it allows a step-wise progressive learning rather than the on-policy algorithms in which a single behavioral policy with frozen parameters collects the experiences for a long rollout duration. Hence, the transitions stored in the experience replay buffer can provide an interpretable visualization on the state space visitations, as shown by Xu et al. (2018).

We run the TD3 algorithm without exploration and with Off-Policy DISCOVER over 1 million time steps in the Swimmer environment. We use a replay buffer of size 1 million transitions where random sampling is performed. Therefore, each visited state is accessible at the end of the training under both exploration settings. We plot the state visitation probabilities in Fig. 9. We employ Kernel Density Estimation based on the samples in 2D embedding space to learn the state visitation probabilities. The 2D embeddings are performed through the joint projection of the data sets onto 2D embedding space by t-SNE (van der Maaten & Hinton, 2008). We choose the Swimmer environment as its state space dimension is relatively small compared to the other environments, preventing

![Fig. 9 State visitation density contours of the TD3 algorithm under greedy action selection and DISCOVER. Large temporal difference errors are represented by darker regions in the TD error contours](image-url)
possible artifacts in the low-dimensional visualization. Moreover, in Sect. 5.2, we observe that actor-critic methods usually perform poorly in the Swimmer environment, which can be considered challenging (Henderson et al., 2018). To further reduce the possibility of projection error, we first perform PCA to reduce the state dimension from 8 to 4 prior to t-SNE. The PCA operation results in a proportion of variance explained value of 0.984 and 0.966 for no exploration and Off-Policy DISCOVER data sets, respectively.

As shown in Fig. 9, we have three groups of visualization for each data set. Moreover, to give insights on the TD error maximization in the exploration, we colorize the states with the TD errors respective to their transitions. Thus, darker regions correspond to large absolute TD errors. We group the state space visualizations into the states visited in the earlier, intermediate, and later stages of the training. In the earlier stages of the data collection, the greedy action selection spans the two sides of the state space. The agent nearly does not see the middle space throughout the training, and visited states settle to a smaller region on both sides of the state space. Furthermore, the TD errors of the corresponding states are practically the same. This is an outcome of the greedy action selection in that the agent usually chooses the same actions and observes the same states, TD error of which are constantly reduced by Q-learning.

However, our approach diversifies the state visitation distributions ranging from different modes of regions, compared to the greedy action selection. From the earlier to the later stages of data collection, the exploration policy learns how to direct the agent to state-action pairs with high TD error, which we observe from the increasing similarity of the state visitation probabilities and TD error heat map. Hence, this comparison indicates that our exploration policy is progressively learning a stationary policy that maximizes the TD error and provides a robust exploration strategy to improve the underlying continuous control method.

Our empirical studies also demonstrate an interesting insight. In all stages of the training, DISCOVER consistently visits almost complementary distinct states in the distribution space compared to the greedy action selection. We deduce that our exploration policy also learns how the agent greedily selects the actions and systematically directs it to explore diverse state regions based on this selection. This enables a consistent and continuous improvement in the agent’s performance and designates our global and directed exploration strategy, which is notably different from noise-based exploration, which remains local. Finally, we obtain a considerable performance improvement by our exploration framework in this comparison, where the maximum return obtained by greedy and DISCOVER agents are 121.56 and 172.35, respectively.

6 Conclusion

This paper introduces a general and scalable deep directed exploration framework that adapts the psychological animal behavior to continuous control deep reinforcement learning. Motivated by the gradients of the value functions, our method directs agents to less known state spaces with high temporal difference error. Our work shows that the introduced deep deterministic exploration policy can extract useful information from the collected experiences for exploration and combine the intrinsic motivations in the current literature into a single exploration objective. Furthermore, by theoretical justifications, our directed exploration framework is guaranteed to have a faster convergence than undirected strategies in any ergodic Markov decision process.
When combined with actor-critic algorithms, we empirically show that our strategy leads the agents to diverse state regions that are orthogonal to the greedy actions and outperforms the competing exploration methods by a significant margin in most of the tested tasks. Moreover, an extensive set of ablation studies demonstrate that when our exploration framework adopts the same update rule, hyper-parameter setting, and structure of the baselines’ policy, it achieves maximum performance. Hence, our effective exploration strategy comes with the cost of training a single neural network and optimizing a single hyper-parameter which adjusts the exploration degree. Finally, the introduced exploration framework is readily applicable to various on- and off-policy actor-critic algorithms through a generic approach, and code to implement DISCOVER is publicly available at our repository (see footnote 1) to ensure reproducibility.

Appendix 1: detailed experimental setup

Software and environment

All networks are trained with PyTorch (version 1.8.1) (Paszke et al., 2019), using default values for all unmentioned hyper-parameters. Performances of all methods are evaluated in MuJoCo (mujoco-py version 1.50), and Box2D (version 2.3.10) physics engines interfaced by OpenAI Gym (version 0.17.3), using v3 environment for BipedalWalker and v2 for rest of the environments. The environment dynamics, state and action spaces, and reward functions are not pre-processed and modified for easy reproducibility and fair evaluation procedure with the baseline and competing algorithms. Each environment episode runs for a maximum of 1000 steps until a terminal condition is encountered. The multi-dimensional action space for all environments is within the range \((-1, 1)\) except for Humanoid, which uses the range of \((-0.4, 0.4)\).

Evaluation

All experiments are run for 1 million time steps with evaluations every 1000 time steps, where an evaluation of an agent records the average reward over 10 episodes without exploration noise and network updates. We use a new environment with a fixed seed (the training seed + a constant) for each evaluation to decrease the variation caused by different seeds. Therefore, each evaluation uses the same set of initial start states.

We report the average evaluation return of 10 random seeds for each environment, including the initialization of behavioral policies, simulators, network parameters, and dependencies. Unless stated otherwise, each agent is trained by one training iteration after each time step. Agents are trained by batches of transitions uniformly sampled from the experience replay. Learning curves are used to show performance, and they are given as an average of 10 trials with a shaded zone added to reflect a half standard deviation across the trials. The curves are smoothed uniformly over a sliding window of 5 evaluations for visual clarity.
Visualization of the state visitations

We visualize the states within the collected transitions for greedy action selection and DISCOVER under the TD3 algorithm while learning in the Swimmer environment over 1 million time steps. The results are reported over a single seed. We consider the last 975,000 transitions in the replay buffer as the first 25,000 is sampled from the environment’s action space. We first separately project the datasets onto a 4D state space through PCA to reduce the visual artifacts.

Later, we jointly embed the resulting datasets onto the 2D state space through the t-SNE implementation of the openTSNE library (Poličar et al., 2019). We use a perplexity value of 1396 and euclidean metric in measuring the distances between each state. The t-SNE is run over 1500 iterations. Default values in openTSNE is used for all unmentioned parameters. We split the datasets into three portions and separately visualize them, where each portion contains 325,000 samples. The PCA operation yields a proportion of variance explained by the value of 0.984 and 0.966 for the greedy and DISCOVER datasets.

Implementation

Our implementation of A2C and PPO is based on the code from the well-known repository (see footnote 5), following the tuned hyper-parameters for the considered continuous control tasks. For the implementation of TD3, we use the author’s GitHub repository (see footnote 4) for the fine-tuned version of the algorithm and the DDPG implementation. For SAC, we follow structure outlined in the original paper.

We implement NoisyNet by adapting the code from authors’ GitHub repository (see footnote 2) to the baseline actor-critic algorithms. Authors’ OpenAI Baselines implementation (see footnote 3) is used to implement PSNE. Similar to SAC, we refer to the original papers in implementing Deep Coherent Exploration and Meta-Policy Gradient as the authors did not provide a valid code repository.

Architecture and hyper-parameter setting

The on-policy methods, A2C and PPO follow the tuned hyper-parameters for the MuJoCo and Box2D tasks provided by the repository (see footnote 5). Our implementation of the off-policy actor-critic algorithms, DDPG, SAC, and TD3, closely follows

| Hyper-parameter                      | Value                           |
|--------------------------------------|---------------------------------|
| Actor regularization                | None                            |
| Optimizer                            | Adam (Kingma & Ba, 2015)        |
| Nonlinearity                         | ReLU                            |
| Gradient clipping                    | False                           |
| Number of hidden layers (all networks) | 2                              |
the set of hyper-parameters given in the respective papers. For DDPG and SAC, we use the fine-tuned environment-specific hyper-parameters provided by the OpenAI Baselines3 Zoo (see footnote 6). TD3 uses the fine-tuned parameters provided in the author’s GitHub repository (see footnote 4). Shared and environment and algorithm specific hyper-parameters for the off-policy methods are given in Tables 2, 3, and 4. Additionally Tables 5 and 6 reports the shared and algorithm specific tuned hyper-parameters for the on-policy baselines, respectively. Note that entropy coefficient used for A2C and PPO is 0, meaning that there is no entropy maximization and hence, inherent exploration. Nevertheless, the tuned 0 value corresponds to greedy action selection, which is found to perform best.
For the parameter-space noise algorithms, we initialize the parameter noise at 0.017 and 0.034, which gives the best results in practice for on- and off-policy algorithms, respectively, as suggested by Zhang and Van Hoof (2021). Furthermore, we use $\beta = 0.01$ for all environments in the Deep Coherent Exploration algorithm, and set the mean-squared error threshold in PSNE to 0.1.

The exploration framework of DISCOVER strictly follows the actor framework in the corresponding baseline algorithms. This includes the depth and size of the networks, learning rate, optimizer, nonlinearity, target, and behavioral policy update frequency,
target network learning rate, and the number of gradient steps in the updates. Moreover, we still use the exploration policy in the exploration time steps at the beginning of each training.

### Appendix 2: complete experimental results

#### Additional evaluation results

Additional evaluation results in the InvertedDoublePendulum, InvertedPendulum, and Reacher environments for A2C, PPO, DDPG, SAC, and TD3 are reported in Figs. 10, 11, 12, 13, and 14, respectively.

![Graph](image1)

Fig. 10 Additional evaluation curves for the set of MuJoCo continuous control tasks under the A2C algorithm. The shaded region represents half a standard deviation of the average evaluation return over 10 random seeds. A sliding window of size 5 smoothes curves for visual clarity.

![Graph](image2)

Fig. 11 Additional evaluation curves for the set of MuJoCo continuous control tasks under the PPO algorithm. The shaded region represents half a standard deviation of the average evaluation return over 10 random seeds. A sliding window of size 5 smoothes curves for visual clarity.
**Fig. 12** Additional evaluation curves for the set of MuJoCo continuous control tasks under the DDPG algorithm. The shaded region represents half a standard deviation of the average evaluation return over 10 random seeds. A sliding window of size 5 smooths curves for visual clarity.

**Fig. 13** Additional evaluation curves for the set of MuJoCo continuous control tasks under the SAC algorithm. The shaded region represents half a standard deviation of the average evaluation return over 10 random seeds. A sliding window of size 5 smooths curves for visual clarity.

**Fig. 14** Additional evaluation curves for the set of MuJoCo continuous control tasks under the TD3 algorithm. The shaded region represents half a standard deviation of the average evaluation return over 10 random seeds. A sliding window of size 5 smooths curves for visual clarity.
Learning curves for the ablation studies

Exploration direction regularization

From Figs. 15 and 16, we observe that the increasing degree of exploration detrimentally degrades the performance of the baseline algorithms. This is an expected result as highly perturbed actions and state distributions prevent agents from effectively learning from its mistakes. Inversely, insufficient exploration obtains a notable but suboptimal performance. In fact, greedy action selection, i.e., $\lambda = 0.0$, performs second best after DISCOVER yet, converges to a suboptimal policy.

The optimal exploration regularization value is found to be 0.1 and 0.3 for on- and off-policy settings. These values either enables a faster convergence to the optimal policy, e.g., Humanoid, or higher evaluation results, e.g., Swimmer, or both, e.g., HalfCheetah, Hopper. In addition, from Table 1, we find that the last 10 returns for $\lambda = 0.1$ is higher than $\lambda = 0.3$ in the Hopper-v2 environment under the off-policy setting. However, this is not visible in Fig. 16 due to the sliding window on the evaluation results. As $\lambda = 0.3$ setting has a faster convergence than $\lambda = 0.1$, we observe a better overall performance in the plots.

As explained, we perform our ablation studies in environments with different characteristics. Interestingly, we find that $\lambda = 0.3$ can generalize all the tested tasks. Hence, we infer $\lambda = 0.0$.

$\lambda = 0.0$ $\lambda = 0.1$ $\lambda = 0.3$
$\lambda = 0.6$ $\lambda = 0.9$ $\lambda = 1.0$

Fig. 15 Evaluation curves for the set of MuJoCo and Box2D continuous control tasks under PPO + DISCOVER when $\lambda = \{0.0, 0.1, 0.3, 0.6, 0.9, 1.0\}$. The shaded region represents half a standard deviation of the average evaluation return over 10 random seeds. A sliding window of size 5 smoothes curves for visual clarity

$\lambda = 0.0$ $\lambda = 0.1$ $\lambda = 0.3$
$\lambda = 0.6$ $\lambda = 0.9$ $\lambda = 1.0$

Fig. 16 Evaluation curves for the set of MuJoCo and Box2D continuous control tasks under TD3 + DISCOVER when $\lambda = \{0.0, 0.1, 0.3, 0.6, 0.9, 1.0\}$. The shaded region represents half a standard deviation of the average evaluation return over 10 random seeds. A sliding window of size 5 smoothes curves for visual clarity
that one can tune DISCOVER on a single physics dynamics and use the optimal value in different environments.

Ablation study of DISCOVER

Overall, Fig. 17 demonstrates that all settings exhibit a similar performance except for HalfCheetah as the complete algorithm converges faster to significantly higher evaluation results. This is because the HalfCheetah environment vastly requires off-policy samples to be solved, as highlighted by Henderson et al. (2018). Therefore, the complete algorithm can further benefit from off-policy learning, demonstrated in the HalfCheetah environment.

In the rest of the environments, the complete algorithm attains slightly faster convergence to higher returns, from which we conclude that the ensemble of each component is the most effective setting in improving the baseline’s policy. Thus, the exploration policy of DISCOVER should mimic the baseline’s policy framework to obtain the optimal performance.

Author contributions All authors contributed to the study conception and design. Formal analysis, investigation and literature search was performed by Baturay Saglam. The first draft of the manuscript was written by Baturay Saglam, and editing was done by Baturay Saglam and Suleyman Serdar Kozat. Revision and supervision were performed by Suleyman Serdar Kozat. All authors commented on previous versions of the manuscript.

Funding The authors did not receive funds, grants, or other support from any organization for the submitted work.

Availability of data and materials Simulators used in the experiments are freely and publicly available online: https://gym.openai.com/.

Code availability The source code for the research leading to these results are made publicly available at the corresponding author’s GitHub repository: https://github.com/baturaysaglam/DISCOVER.

Declarations

Conflict of interest All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.
References

Barto, A. G. (2013). Intrinsic motivation and reinforcement learning. In G. Baldassarre, & M. Miorilli (Eds.), Intrinsicly motivated learning in natural and artificial systems (pp. 17–47). Springer. https://doi.org/10.1007/978-3-642-32375-1_2

Barto, A. G., & Simsek, O. (2005). Intrinsic motivation for reinforcement learning systems. In The thirteenth yale workshop on adaptive and learning systems (pp. 113–118).

Bellmara, M. G., Naddaf, Y., Veness, J., & Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research, 47, 253–279. https://doi.org/10.1613/jair.3912

Bellman, R. (1957). Dynamic programming. Dover Publications.

Berns, G. S., McClure, S. M., Pagnoni, G., & Montague, P. R. (2001). Predictability modulates human brain response to reward. Journal of Neuroscience, 21(8), 2793–2798.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. CoRR. arXiv:1606.01540

Dayan, P. (2002). Motivated reinforcement learning. In T. Dietterich, S. Becker, & Z. Ghahramani (Eds.), Advances in neural information processing systems (Vol. 14). MIT Press. https://proceedings.neurips.cc/paper/2001/file/051928341be67dca0300e04104d9047-Paper.pdf

Dayan, P., & Sejnowski, T. J. (1996). Exploration bonuses and dual control. Machine Learning, 25(1), 5–22. https://doi.org/10.1007/BF00115298

Dhariwal, P., Hesse, C., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. CoRR. arXiv:1606.01540

Dhariwal, P., Hesse, C., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. CoRR. arXiv:1606.01540

Dhariwal, P., Hesse, C., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. CoRR. arXiv:1606.01540

Dhariwal, P., Hesse, C., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. CoRR. arXiv:1606.01540

Dhariwal, P., Hesse, C., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. CoRR. arXiv:1606.01540

Dhariwal, P., Hesse, C., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. CoRR. arXiv:1606.01540

Dhariwal, P., Hesse, C., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. CoRR. arXiv:1606.01540

Dhariwal, P., Hesse, C., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. CoRR. arXiv:1606.01540

Dhariwal, P., Hesse, C., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. CoRR. arXiv:1606.01540

Dhariwal, P., Hesse, C., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. CoRR. arXiv:1606.01540

Dhariwal, P., Hesse, C., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. CoRR. arXiv:1606.01540

Dhariwal, P., Hesse, C., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. CoRR. arXiv:1606.01540

Dhariwal, P., Hesse, C., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. CoRR. arXiv:1606.01540

Fujimoto, S., van Hoof, H., & Meger, D. (2018). Addressing function approximation error in actor-critic methods. In J. Dy, & A. Krause (Eds.), Proceedings of the 35th international conference on machine learning (Vol. 80, pp. 1587–1596). PMLR. https://proceedings.mlr.press/v80/fujimoto18a.html

Garris, P. A., Kilpatrick, M., Bunin, M. A., Michael, D., Walker, Q. D., & Wightman, R. M. (1999). Dissociation of dopamine release in the nucleus accumbens from intracranial self-stimulation. Nature, 398(6722), 67–69. https://doi.org/10.1038/18019

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, & K. Weinberger (Eds.), Advances in neural information processing systems (Vol. 27). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f16f8f06494e97b4afccf3-Paper.pdf

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In J. Dy, & A. Krause (Eds.), Proceedings of the 35th international conference on machine learning (Vol. 80, pp. 1861–1870). PMLR. https://proceedings.mlr.press/v80/haarnoja18b.html

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2018). Deep reinforcement learning that matters. In Proceedings of the AAAI conference on artificial intelligence (Vol. 321). https://ojs.aaai.org/index.php/AAAI/article/view/11694

Kearns, M., Mansour, Y., & Ng, A. Y. (2002). A sparse sampling algorithm for near-optimal planning in large Markov decision processes. Machine Learning, 49(2), 193–208. https://doi.org/10.1023/A:1017932429737

Kearns, M., & Singh, S. (2002). Near-optimal reinforcement learning in polynomial time. Machine Learning, 49(2), 209–232. https://doi.org/10.1023/A:1017984413808

Kilpatrick, M. R., Rooney, M. B., Michael, D. J., & Wightman, R. M. (2000). Extracellular dopamine dynamics in rat caudate–putamen during experimenter-delivered and intracranial self-stimulation. Neuroscience, 96(4), 697–706.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. Iclr (poster). arXiv:1412.6980

Koenig, S., & Simmons, R. G. (1996). The effect of representation and knowledge on goal-directed exploration. Machine Learning, 22(1), 227–250. https://doi.org/10.1007/BF00114729

Lee, K., Kim, G.-H., Ortega, P., Lee, D. D., & Kim, K.-E. (2019). Bayesian optimistic Kullback–Leibler exploration. Machine Learning, 108(5), 765–783. https://doi.org/10.1007/s10994-018-5767-4

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2016). Continuous control with deep reinforcement learning. Iclr (poster). arXiv:1509.02971

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. Machine Learning, 8(3), 293–321. https://doi.org/10.1023/A:1022628806385

 Springer
McClure, S. M., Berns, G. S., & Montague, P. R. (2003). Temporal prediction errors in a passive learning task activate human striatum. *Neuron, 38*(2), 339–346.

McClure, S. M., Daw, N. D., & Read Montague, P. (2003). A computational substrate for incentive salience. *Trends in Neurosciences, 26*(8), 423–428. https://doi.org/10.1016/j.tins.2003.07.007

Meuleau, N., & Borghine, P. (1999). Exploration of multi-state environments: Local measures and back-propagation of uncertainty. *Machine Learning, 35*(2), 117–154. https://doi.org/10.1023/A:1007541107674

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In M. F. Balcan, & K. Q. Weinberger (Eds.), *Proceedings of the 33rd international conference on machine learning* (Vol. 48, pp. 1928–1937). USAPMLR. https://proceedings.mlr.press/v48/mnih16.html

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellmeme, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529–533. https://doi.org/10.1038/nature14236

Montague, P. R., Dayan, P., & Sejnowski, T. J. (1996). A framework for mesencephalic dopamine systems based on predictive Hebbian learning. *Journal of Neuroscience, 16*(5), 1936–1947.

Moore, A. W., & Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning, 13*(1), 103–130. https://doi.org/10.1007/BF00993104

Nouri, A., & Littman, M. L. (2010). Dimension reduction and its application to model-based exploration in continuous spaces. *Machine Learning, 81*(1), 85–98. https://doi.org/10.1007/s10994-010-5202-y

O’Doherty, J. P., Dayan, P., Friston, K., Critchley, H., & Dolan, R. J. (2003). Temporal difference models and reward-related learning in the human brain. *Neuron, 38*(2), 329–337.

Pagnoni, G., Zink, C. F., Montague, P. R., & Berns, G. S. (2002). Activity in human ventral striatum locked to errors of reward prediction. *Nature Neuroscience, 5*(2), 97–98.

Parberry, I. (2013). *Introduction to game physics with box2d* (1st ed.). USACRC Press, Inc.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., & Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems* (Vol. 32, pp. 8024–8035). Curran Associates, Inc. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R. Y., Asfour, T., Abbeel, P., & Andrychowicz, M. (2018). Parameter space noise for exploration. In *International conference on learning representations* (openreview.net/forum?id=ByBA1eAZ)

Poličar, P. G., Stražar, M., & Zupan, B. (2019). opentsne: A modular python library for t-SNE dimensionality reduction and embedding. bioRxiv. Retrieved from https://www.biorxiv.org/content/early/2019/08/13/731877

Precup, D., Sutton, R., & Dasgupta, S. (2001). Off-policy temporal-difference learning with function approximation. In *Proceedings of the 18th international conference on machine learning*.

Raffin, A. (2020). RI baselines3 zoo. GitHub. https://github.com/DLR-RM/rl-baselines3-zoo

Ryan, R. M., & Deci, E. L. (2000). Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary Educational Psychology, 25*(1), 54–67. https://www.sciencedirect.com/science/article/pii/S0361476X99910202.

Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2016). High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the international conference on learning representations* (iclr).

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv:1707.06347

Schultz, W. (1998). Predictive reward signal of dopamine neurons. *Journal of Neurophysiology, 80*(1), 1–27.

Schultz, W., Dayan, P., & Montague, P. R. (1997). A neural substrate of prediction and reward. *Science, 275*(5306), 1593–1599. https://doi.org/10.1126/science.275.5306.1593

Silver, D., Lever, G., Heess, N., Degriss, T., Wierstra, D., & Riedmiller, M. (2014). Deterministic policy gradient algorithms. In E. P. Xing, & T. Jebara (Eds.), *Proceedings of the 31st international conference on machine learning* (Vol. 32, pp. 387–395). PMLR. https://proceedings.mlr.press/v32/silver14.html

Singh, S., Jaakkola, T., Littman, M. L., & Szepesvári, C. (2000). Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning, 38*(3), 287–308. https://doi.org/10.1023/A:1007678930559

Singh, S. P. (1992). Transfer of learning by composing solutions of elemental sequential tasks. *Journal of Artificial Intelligence Research, 4*, 1–27.

Singh, S., Jaakkola, T., Littman, M. L., & Szepesvári, C. (2000). Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning, 38*(3), 287–308. https://doi.org/10.1023/A:1007678930559

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning, 3*(1), 9–44. https://doi.org/10.1007/BF00115009
Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT Press.

Thrun, S. B. (1992). Efficient exploration in reinforcement learning (Technical Report No. CMU-CS-92-102). USA Carnegie Mellon University.

Todorov, E., Erez, T., & Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems* (pp. 5026–5033). https://doi.org/10.1109/IROS.2012.6386109

Uhlenbeck, G. E., & Ornstein, L. S. (1930). On the theory of the Brownian motion. *Physical Review*, 36, 823–841. https://doi.org/10.1103/PhysRev.36.823

van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(66), 2579–2605.

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3), 279–292. https://doi.org/10.1007/BF00992698

Whitehead, S. D. (1991). A complexity analysis of cooperative mechanisms in reinforcement learning. In *Proceedings of the ninth national conference on artificial intelligence* (Vol. 2, pp. 607–613). AAAI Press.

Whitehead, S. D., & Ballard, D. H. (1991). Learning to perceive and act by trial and error. *Machine Learning*, 7(1), 45–83. https://doi.org/10.1023/A:1022619109594

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3), 229–256. https://doi.org/10.1007/BF00992696

Xu, T., Liu, Q., Zhao, L., & Peng, J. (2018). Learning to explore via meta-policy gradient. In J. Dy, & A. Krause (Eds.), *Proceedings of the 35th international conference on machine learning* (Vol. 80, pp. 5463–5472). https://proceedings.mlr.press/v80/xu18d.html

Zhang, Y., & Van Hoof, H. (2021). Deep coherent exploration for continuous control. In M. Meila, & T. Zhang (Eds.), *Proceedings of the 38th international conference on machine learning*. PMLR (Vol. 139, pp. 12567–12577). https://proceedings.mlr.press/v139/zhang21t.html

Zheng, Z., Oh, J., & Singh, S. (2018). On learning intrinsic rewards for policy gradient methods. S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), Advances in neural information processing systems (Vol. 31). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2018/file/51de85ddd068f0b7c787691d356176df9-Paper.pdf

**Publisher’s Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.