NOVEL DEEP REINFORCEMENT LEARNING ALGORITHMS FOR CONTINUOUS CONTROL

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF

MASTER OF SCIENCE

IN

ELECTRICAL AND ELECTRONICS ENGINEERING

By

Baturay Sağlam

June 2023

NOVEL DEEP REINFORCEMENT LEARNING ALGORITHMS FOR CONTINUOUS CONTROL

By Baturay Sağlam

June 2023

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Süleyman Serdar Kozat(Advisor)

Aykut Koç

Çağatay Candan

Approved for the Graduate School of Engineering and Science:

Orhan Arıkan
Director of the Graduate School

ABSTRACT

NOVEL DEEP REINFORCEMENT LEARNING ALGORITHMS FOR CONTINUOUS CONTROL

Baturay Sağlam
M.S. in Electrical and Electronics Engineering
Advisor: Süleyman Serdar Kozat
June 2023

Continuous control deep reinforcement learning (RL) algorithms are capable of learning complex and high-dimensional policies directly from raw sensory inputs. However, they often face challenges related to sample efficiency and exploration, which limit their practicality for real-world applications. In light of this, we introduce two novel techniques that enhance the performance of continuous control deep RL algorithms by refining their experience replay and exploration mechanisms. The first technique introduces a novel framework for sampling experiences in actor-critic methods. Specifically designed to stabilize and prevent divergence caused by Prioritized Experience Replay (PER), our framework effectively trains both actor and critic networks by striking a balance between temporal-difference error and policy gradient. Through both theoretical analysis and empirical investigations, we demonstrate that our framework is effective in improving the performance of continuous control deep RL algorithms. The second technique encompasses a directed exploration strategy that relies on intrinsic motivation. Drawing inspiration from established theories on animal motivational systems and adapting them to the actor-critic setting, our strategy showcases its effectiveness by generating exploratory behaviors that are both informative and diverse. It achieves this by maximizing the error of the value function and unifying the existing intrinsic exploration objectives in the literature. We evaluate the presented methods on various continuous control benchmarks and demonstrate that they outperform state-of-the-art methods while achieving new levels of performance in deep RL.

Keywords: deep reinforcement learning, continuous control, off-policy learning, exploitation-exploration.

ÖZET

SÜREKLİ KONTROL İÇİN YENİ DERİN PEKİŞTİRMELİ ÖĞRENME ALGORİTMALARI

Baturay Sağlam
Elektrik ve Elektronik Mühendisliği, Yüksek Lisans
Tez Danışmanı: Süleyman Serdar Kozat
Haziran 2023

Sürekli kontrol altında derin pekiştirmeli öğrenme algoritmaları, ham duyusal girdilerden doğrudan karmaşık ve yüksek boyutlu politikalar öğrenebilme kapasitesine sahiptir. Ancak, genellikle örnekleme verimliliği ve keşif ile ilgili zorluklarla karşılaşırlar, bu da gerçek dünya görevleri için uygulanabilinirliklerini sınırlar. Bu bağlamda, sürekli kontrol altında derin pekiştirmeli öğrenme algoritmalarının performansını geliştiren iki yeni teknik sunuyoruz. İlk teknik, aktör-eleştirmen metotlarında deneyleri örneklemek için yeni bir yaklaşım sunmaktadır. Öncelikli Deneyim Tekrar algoritması tarafından neden olunan kararsızlık ve ayrışmayı önlemek ve stabilize etmek için özel olarak tasarlanan tekniğimiz, zamansal-fark hatası ve politika gradyanı arasında denge sağlayarak hem aktör hem de eleştirmen ağlarını etkili bir şekilde eğitebilmektedir. Teorik analizler ve deneysel çalışmalar, yöntemimizin sürekli kontrol altında derin pekiştirmeli öğrenme algoritmalarının performansını iyileştirmede etkili olduğunu göstermektedir. İkinci teknik, içsel motivasyona dayalı yönlendirilmiş bir keşif stratejisini içermektedir. Hayvan motivasyon sistemleri üzerine kurulu kuramlardan esinlenerek ve bunları sürekli kontrol ortamına adapte ederek, stratejimiz, bilgilendirici ve çeşitlilik gösteren keşif davranışları oluşturmada etkinliğini sergilemektedir. Bunun, değer fonksiyonunun hatasını maksimize ederek ve mevcut literatürde bulunan içsel keşif hedeflerini birleştirerek gerçekleştiğini gösteriyoruz. Sunulan yöntemleri çeşitli sürekli kontrol testlerinde değerlendiriyor ve derin pekiştirmeli öğrenmede yeni performans seviyelerine ulaşarak mevcut en iyi yöntemleri geride bıraktığımızı gösteriyoruz.

Anahtar sözcükler: derin pekiştirmeli öğrenme, sürekli kontrol, politika-dışı öğrenme, sömürü-keşif.

Acknowledgement

I would like to express my sincere gratitude to all the people who have supported me throughout my master's journey. First and foremost, I thank my mother and father for their unconditional love and encouragement. They always believed in me and gave me what they had. They are the source of my strength, diligence, and motivation.

I am deeply indebted to my master's supervisor, Prof. Süleyman Serdar Kozat, for his invaluable guidance and mentorship. He not only taught me how to think critically as a researcher, but enriched my vision to think realistically and helped me to find a balance between realism and idealism in life. He has been a great role model for me both as a researcher and as a person. I would not be at Bilkent for my master's and would not be pursuing my Ph.D. without his support.

I also thank my best friends, Onat Dalmaz and Kaan Gonc, for being with me throughout my higher education, both at college and master's. We have overcome various challenges both in our academic life and social life. They have been with me since 2016 and I share my success with them. From parties to lectures, we always supported each other. Onat, in particular, has been a constant source of inspiration and motivation for me. He has always pushed me to do my best and to challenge myself. He has also been a great partner in crime, always ready to join me in any adventure or mischief. Even if Kaan left his projects to ten minutes before the deadline, we managed to obtain the best results in all of our courses and projects. Lastly, I appreciate their collaboration in our recent work, *User Feedback-based Online Learning for Intent Classification*, which has finally come alive after two years of discussion and two weeks of experimentation. In the end, they have been more than friends to me; they have been my research collaborative brothers.

I would also like to thank my wonderful friends Eda Dede, Begüm Atsü, Hande Davul, and especially Ecem Zengin. They have been among the rare few who believed I would succeed. They have been with me to shape my state of mind and fuel my motivation. They have always cheered me up, listened to me attentively, and encouraged me to follow my dreams passionately. Ecem, in particular, has been the most precious person in my life. She has been my soulmate and my partner in everything. She has supported me in every step of my journey, with her unconditional love, profound wisdom, and endless patience. She has made me a better person and a happier one.

I would like to acknowledge my research collaborators with whom we worked at Prof. Kozat's team: Doğan Can Çiçek and Furkan Burak Mutlu. They have contributed to my research with their insights, feedback, and technical skills. I have learned a lot from them and enjoyed working with them. I would not be able to study my articles if it was not for their ideas from being a spark all the way to a lava.

Last but not least, I thank Turk Telekom for providing me the 5G and Beyond Graduate Scholarship Programme. This scholarship has been a great opportunity for me to pursue my master's degree at Bilkent University and to conduct cutting-edge research on wireless communications and deep reinforcement learning. I also thank Dr. Selami Çiftçi for taking care of me, helping me even with the smallest things, and regularly meeting me and believing in me. He has been a great mentor, although we have met only for a few months. He has given me valuable advice, feedback, and support throughout my scholarship period.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Preliminaries

Reinforcement learning (RL) is a branch of machine learning that focuses on how agents can learn to interact with their environment and maximize a reward signal. RL has shown promising results in various domains such as decision-making in wireless communication systems [1]. However, many real-world problems involve continuous and high-dimensional action spaces, where the agent can choose any value within a range for each action dimension. For instance, controlling a robot arm or a car requires specifying the torque or the steering angle continuously. These problems pose significant challenges for RL algorithms, as they have to deal with the curse of dimensionality and the exploration-exploitation trade-off. In this thesis, we propose two novel techniques that improve the performance of continuous control deep RL algorithms by refining *(i)* the use of previously collected data *(ii)* and the exploration strategy of the agent.

Off-policy deep RL enables agents to learn from their previous experiences by storing them in a buffer, the experience replay memory [2], and reusing them multiple times to update their policies. Although uniform sampling from the buffer was initially proposed for the experience replay, various sampling methods have been developed to improve the data efficiency by assigning priority scores to the experiences [3, 4, 5, 6, 7].

The idea of priority-based non-uniform sampling in deep RL originates from a technique known as Prioritized Experience Replay (PER) [3], which samples experiences with high temporal-difference (TD) error more frequently, allowing for faster learning and reward propagation by focusing on the most crucial data. While PER is intuitively motivated for learning in discrete action spaces, many empirical studies have shown that it significantly degrades the performance of continuous control algorithms, resulting in suboptimal or random behavior [8, 4, 5, 7]. However, such poor performance lacks a solid theoretical explanation. Therefore, our first technique aims to investigate and provide insights into why continuous control deep RL algorithms are ineffective when combined with PER. Based on this analysis, we also propose novel modifications to PER to enhance its empirical performance, as the first technique introduced in this study.

The trade-off between exploration and exploitation is a longstanding and fundamental challenge in conventional RL and modern deep RL. The main goal of exploration is to ensure that agents collect a diverse set of experiences to avoid adopting a premature behavior. If the exploration is insufficient (i.e., exploitation), agents may miss actions that yield high rewards and converge to a local optimum. In contrast, if the exploration is excessive, agents may waste time and resources by trying many suboptimal actions without effectively using the collected experiences. However, designing an effective and efficient exploration strategy 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 time and resources required for exploration [9].

Although the existing exploration strategies have demonstrated good performance, they have several limitations especially when applied to continuous control deep RL algorithms [10]. First, the *undirected* exploration techniques that rely on learning a probability distribution for an exploration noise conditioned on the agent's observations introduce excessive computational cost and do not work well with off-policy methods as they evaluate the exploration strategy only when the exploration is performed. This causes them to fail in the off-policy setting where the exploration is separated from learning. Second, the *directed* exploration techniques, in which the exploration strategy is learned in a rule-based manner, are mostly designed for hard exploration tasks where rewards are delayed, deceptive, or sparse [11]. In contrast to the directed exploration strategies in hard exploration tasks, intrinsic motivation has been effectively used for exploration in RL. In particular, the agents are guided by an intrinsic motivation based on certain constraints on the agent's observation, inspired by the findings in animal psychology literature. However, these strategies also suffer from several limitations, such as the lack of generalizability to different domains as they only focus on low- or high-dimensional environments, and the scarcity of studies for continuous action spaces.

To this end, our second technique revolves around a novel approach to the exploration-exploitation dilemma by adapting the existing theories on animal motivational systems to deep RL for continuous control. We introduce a scalable directed high-frequency perturbation algorithm that unifies the intrinsic motivations defined in the RL literature. Specifically, we learn an exploration policy that forces the agent to make mistakes and learn the outcomes of unknown or less known actions, making it robust to adapt to its environment rapidly.

We evaluate the performance of the introduced algorithms on the MuJoCo [12] and Box2D [13] continuous control benchmarks interfaced by OpenAI Gym [14]. An extensive set of empirical studies shows that the introduced frameworks significantly outperform the prior studies by a wide margin and achieve notable gains over the state-of-the-art in the majority of the domains tested. Further, the ablation studies verify that each proposed modification is essential to maintain the overall performance in improving the continuous control deep RL.

1.2 Contributions

The first technique improves off-policy learning by theoretically analyzing the drawbacks of the PER algorithm and proposing a set of modifications to alleviate them. The second technique develops a new method for balancing exploration and exploitation that enhances the agent's performance and robustness in complex environments.

1.2.1 Improving the Sampling Efficiency in Off-Policy Learning

1. The PER algorithm [3] samples experiences according to their TD error, which measures the discrepancy between the value function and the actual reward. We show that this can lead to poor policy updates in the actor-critic setting if the value function is uncertain about the sampled experiences. A simple analogy is that a student cannot learn well from a teacher who is unsure about the subject.

2. To address this problem, we suggest that the policy should be updated with low TD error experiences, while the value function should be updated with high TD error experiences. This way, the policy learns from reliable feedback, while the value function reduces its uncertainty.

3. However, this scheme violates the actor-critic theory [15] for continuous control, which requires that the value function and the policy parameters are interdependent. This means that both components should be updated with the same experiences for a fraction of the batch size. We design to sample this fraction uniformly from the replay buffer.

4. Moreover, we argue that sampling strategies alone are not sufficient to prevent outliers and biased transitions from affecting the learning process [8]. We demonstrate that correcting the priority weights of the transitions and the loss of the value function can further improve the performance and robustness of the algorithm.

5. The holistic approach accounting these findings is shown to bring considerable improvements in continuous control tasks in terms of final performance and sample efficiency.

1.2.2 Balancing Exploration and Exploitation

- We draw inspiration from the theories of animal motivation and design an intrinsically motivated exploration strategy for continuous action spaces. We devise a scalable algorithm that learns an exploration policy that guides the agent to less-visited states with high value uncertainty.
- By maximizing the TD error, the proposed exploration strategy encourages the agent to explore unseen or less-known states or actions and to learn their values more accurately. In contrast to the prior works, this motivation also effectively leverages the off-policy learning structure by extracting useful information from the past experiences.
- We provide theoretical analysis and empirical evidence that our technique enjoys faster convergence than the previous undirected deep exploration methods.
- The introduced intrinsically motivated exploration scheme also joins the first technique by providing substantial gains over the state-of-the-art.

1.3 Outline

In the remainder of this thesis, the organization is as follows. Chapter 2 provides the fundamental technical preliminaries of continuous control deep RL. Chapter 3 analyzes the limitations of the PER algorithm and presents the framework for alleviating them, as the first technique. Chapter 4 introduces the second technique of the intrinsically motivated exploration algorithm and provides a theoretical analysis of its fast convergence property. Finally, Chapter 5 portrays the conclusive remarks.

Chapter 2

Background

In this chapter, we start by defining the RL framework that we use in this study. Next, we present the temporal-difference (TD) learning algorithm [16], which is the basis of PER and the exploration strategy we introduce. Lastly, we analyze how the PER algorithm works.

2.1 Deep 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 $s \in \mathcal{S}$ and takes an action $a \in \mathcal{A}$, where $\mathcal{S}$ and $\mathcal{A}$ are state and action spaces, respectively. Depending on its action selection, the agent observes a next state $s' \in \mathcal{S}$ and receives a reward r. The RL problem is often formulated as a fully observable Markov decision process (MDP) represented by a tuple $(\mathcal{S}, \mathcal{A}, P, \gamma)$, 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 π that maximizes the *value* defined as the expected sum of discounted rewards $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i+1}$, where γ prioritizes the short term rewards. The policy can be either deterministic $a = \pi_\phi(s)$ if it maps states to unique actions or stochastic $a \sim \pi_\phi(\cdot|s)$ if it maps states to action probabilities. Note that we denote the distribution of the encountered states and rewards lead by the policy π using $s, r \sim P_\pi$.

In continuous control (i.e., continuous action domains), the maximum of the value R_t is intractable due to the infinite number of possible actions. To overcome this issue, deep RL algorithms use a neural network called a policy or actor network π_ϕ, with parameters ϕ, to map states to actions. These algorithms are known as *actor-critic* methods, and they can control continuous systems by using the actor network to select actions. Actor-critic methods optimize the actor network by gradient ascent over the parameters through computing the policy gradient $\nabla_\phi J(\phi)$ using a policy gradient algorithm.

2.2 Temporal-Difference Learning

Here, we separately examine TD learning for on- and off-policy learning as both constitute the introduced exploration technique differently.

2.2.1 On-Policy Learning

On-policy learning considers agents that learn from the experiences that are generated by the policy that it is currently following. For a given policy π, the state-value function gives the expected return in state s if it follows the policy π:

$$V^\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s]. \tag{2.1}$$

The state-value function is learned by the Bellman equation [17]:

$$V^\pi(s) = \mathbb{E}_{r,s' \sim P_\pi}[r + \gamma V^\pi(s')]. \tag{2.2}$$

In deep RL, the state-value functions are represented by deep function approximators $V_\psi(s)$ with parameters ψ. The state-value network is trained using learning by bootstrapping from the current state s to the next state s', which is an update rule based on the Bellman equation. The Bellman equation establishes a recursive relationship between the value of a state s and the value of the next state s', as expressed in (2.2). Therefore, on-policy TD learning minimizes the loss: $|r + \gamma \mathbb{E}_{s' \sim P_\pi}[V_\psi(s')] - V_\psi(s)|^2$.

2.2.2 Off-Policy Learning

In off-policy learning, agents store the transitions that they experience in the form of tuples (s, a, r, s') in the experience replay buffer [2]. The agents can then use the stored experiences multiple times to update their policies and value networks, which improves their data and sampling efficiency. Off-policy learning considers the value functions with respect to the actions selected by the policy instead of only the observed states. Thus, there exists an action-value function $Q^\pi(s, a) = \mathbb{E}_{s \sim P_\pi, a \sim \pi}[R|s, a]$, associated with the policy π, also known as the critic or Q-function, which represents the expected return R while following the policy π after performing the action a in state s:

$$Q^\pi(s, a) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s, a_0 = a]. \tag{2.3}$$

The action-value function is also determined through the Bellman equation [17]:

$$Q^\pi(s, a) = \mathbb{E}_{r, s' \sim P_\pi, a' \sim \pi}[r + \gamma Q^\pi(s', a')], \tag{2.4}$$

where a' is the action selected by the policy on the observed next state s'.

In the deep setting of off-policy RL, the critic is approximated by deep neural networks Q_θ with parameters θ. Given a transition tuple $\tau = (s, a, r, s')$, off-policy TD learning is performed by minimizing the loss $\delta_\theta(\tau)$ based on the TD error, that is, the action-value prediction error of the Q-network Q_θ. This is also known as Q-learning [18]:

$$y(\tau) = r + \gamma Q_{\theta'}(s', a'), \tag{2.5}$$

$$\delta_\theta(\tau) = y(\tau) - Q_\theta(\tau), \tag{2.6}$$

where the target $y(\tau)$ in (2.5) uses a separate target network with parameters θ' to stabilize the learning process and maintain a fixed objective for the optimal Q-function [19]. The next action a' in (2.5) can be selected by either the behavioral actor network π_ϕ (i.e., the policy that the agent follows for action selection) or the target actor network $\pi_{\phi'}$, depending on the actor-critic algorithm. The target actor network with parameters ϕ' is used to stabilize the updates, in a similar way to the target Q-network. Furthermore, the target parameters are updated either softly or hard by copying a small proportion ζ of the parameters from the behavioral network at each step or by matching it every fixed interval, respectively. Note that the TD error in the latter equation can also be expressed as $\delta_\theta(\tau) = Q_\theta(\tau) - y(\tau)$, yet this does not affect our analysis. Finally, we especially use a separate notation denote the off-policy TD error since we will frequently refer to the off-policy TD error in our discussion of PER and the first technique introduced.

A sampling technique is used to select a batch of transitions $\tau \in \mathcal{B}$ for training the policy and Q-network in the off-policy setting. In each update step, the loss of the Q-network is computed as the average over the batch of sampled transitions $\mathcal{B}$: $\frac{1}{|\mathcal{B}|}\sum_{\tau \in \mathcal{B}} \delta_\theta(\tau)$, where $|\mathcal{B}|$ is the number of transitions in $\mathcal{B}$. Note that the batch size does not affect our analysis on the prioritization because the expected gradient remains unchanged [8, 20, 21].

Lastly, the next action chosen by the behavioral or target policy network in TD learning is usually perturbed by random action noise to realize the exploration in the next state. This technique is called target policy smoothing regularization [22]:

$$Q_\theta(s, a) = r + \gamma \mathbb{E}_{s' \sim P_\pi, a' \sim \pi}[Q_{\theta'}(s', \tilde{a}')], \quad \tilde{a}' \sim a' + \mathcal{N}(0, \sigma), \tag{2.7}$$

where $\mathcal{N}(0, \sigma)$ is a Gaussian random variable with zero mean and standard deviation σ.

2.3 Prioritized Experience Replay

Prioritized Experience Replay is a technique for replay buffers that samples transitions directly proportional to their absolute TD error. The main idea of PER is that learning from the highest error samples will lead to the most significant performance improvement. PER makes two changes to the standard uniform sampling. First, it uses a stochastic prioritization scheme. The reason is that TD errors are only updated for replayed transitions. Hence, the transitions with initially high TD errors are replayed more often, resulting in a greedy prioritization. Moreover, the noisy Q-value estimates increase the variance due to the greedy sampling. Therefore, sampling transitions directly proportional to their TD errors can cause overfitting. To avoid this, each transition τ_i is assigned a probability value that is proportional to its absolute TD error, and raised to the power of a hyperparameter ε to smooth out the extremes:

$$p(\tau_i) = \frac{|\delta_\theta(\tau_i)|^\varepsilon + \mu}{\sum_j (|\delta_\theta(\tau_j)|^\varepsilon + \mu)}, \tag{2.8}$$

where a small constant μ is added to prevent transitions from having zero probabilities. Otherwise, they would never be sampled again. This is required because the latest value of a transition's TD error is estimated by the TD error when it was last sampled. In addition, favoring large TD error transitions with the stochastic prioritization shifts the distribution of s' to $\mathbb{E}_{s' \sim P_\pi, a' \sim \pi}[Q(s', a')]$. This can be corrected through importance sampling with ratios $w(\tau_i)$:

$$\hat{w}(\tau_i) = \left(\frac{1}{|R|} \cdot \frac{1}{p(\tau_i)} \right)^{\upsilon}, \tag{2.9}$$

$$w(\tau_i) = \frac{\hat{w}(\tau_i)}{\max_j \hat{w}(\tau_j)}, \tag{2.10}$$

$$\mathcal{L}_{\text{PER}}(\delta_\theta(\tau_i)) = w(\tau_i)\mathcal{L}(\delta_\theta(\tau_i)), \tag{2.11}$$

where $\mathcal{L}$ denotes the TD error under prioritization, $|R|$ is the number of transitions in the replay buffer, and the hyperparameter υ is used to reduce the high variance caused by the importance sampling weights. The latter equation corrects the distribution shift by using a ratio between uniform sampling with probability $\frac{1}{|R|}$ and the ratio defined in (2.8). This method reduces the influence of high priorities on the distribution. Finally, the υ value is annealed from a predefined initial value υ_0 to 1, to eliminate the bias introduced by the distributional shift.

Chapter 3

Actor Prioritized Experience Replay

The content of this chapter reflects the study described in the following publication:

- B. Saglam, F. B. Mutlu, D. C. Cicek, and S. S. Kozat, "Actor prioritized experience replay," in *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.

In this chapter, we first analyze the causes of the poor empirical performance of the PER algorithm. Based on our theoretical insights, we then introduce a set of modifications for adapting it to the actor-critic methods for continuous control. We consolidate the proposed setup under the *Loss-Adjusted Approximate Actor Prioritized Experience Replay* (LA3P) algorithm. Finally, we review the literature and compare LA3P with the prior methods that aim to correct PER.

3.1 Prioritized Sampling in Actor-Critic Algorithms

First, we establish the theoretical foundations for why the prioritized sampling deteriorates the performance of the actor-critic methods. We show in Assumption 1 that some transitions may have the property that their TD error magnitude can amplify the Q-value estimation error magnitude. Based on our theoretical results, we demonstrate in Theorem 1 that optimizing a policy with transitions that have large TD error will cause the gradient of the actor network computed under the Q-network's estimates to diverge from the true gradient computed under the optimal Q-function. In addition to our theoretical analysis, we also address the problems addressed by a previous study that explains another aspect for the degraded performance of PER.

Assumption 1. *Consider the temporal-difference error δ_θ associated with the critic network Q_θ in off-policy actor-critic algorithms. Then, there exists a transition tuple $\tau_i = (s_i, a_i, r_i, s_{i+1})$ with $\delta_\theta(\tau_i) \neq 0$ such that if the absolute temporal-difference error on τ_i increases, the absolute Q-value estimation error on at least τ_i or τ_{i+1} will also increase.*

Discussion of Assumption 1. We assume that the target networks do not influence the estimation, as their purpose is to provide stability and a fixed goal over the updates [19]. We start by expressing the temporal-difference error $\delta_\theta(\tau_i)$ as a function of the bootstrapped value estimation:

$$\delta_\theta(\tau_i) = r_i + \gamma Q_\theta(s_{i+1}, a_{i+1}) - Q_\theta(s_i, a_i), \tag{3.1}$$

where $a_{i+1} \sim \pi_\phi(\cdot|s_{i+1})$. We know that the optimal action-value function Q^π under the policy π produces no TD error:

$$\delta^\pi(\tau_i) = r_i + \gamma Q^\pi(s_{i+1}, a_{i+1}) - Q^\pi(s_i, a_i) = 0. \tag{3.2}$$

Subtracting (3.2) from (3.1) yields:

$$\delta_\theta(\tau_i) = \gamma \underbrace{\left(Q_\theta(s_{i+1}, a_{i+1}) - Q^\pi(s_{i+1}, a_{i+1})\right)}_{:=\epsilon_{\tau_{i+1}}} - \underbrace{\left(Q_\theta(s_i, a_i) - Q^\pi(s_i, a_i)\right)}_{:=\epsilon_{\tau_i}} \neq 0. \tag{3.3}$$

Clearly, ϵ_{τ_i} and $\epsilon_{\tau_{i+1}}$ denote the estimation error at the current and next steps, respectively. We then examine the following cases for different signs of these variables when the absolute TD error grows, and demonstrate that if the absolute value of TD error increases, then the absolute value of either ϵ_{τ_i} or $\epsilon_{\tau_{i+1}}$ also increases. Note that $\gamma \geq 0$ is ignored as it is constant.

Case 1: $|\delta_\theta(\tau_i)|$ increases when $\delta_\theta(\tau_i), \epsilon_{\tau_i}, \epsilon_{\tau_{i+1}} > 0$. This can happen if $\epsilon_{\tau_{i+1}}$ becomes a greater positive number as well, which implies that $|\epsilon_{\tau_{i+1}}|$ eventually increases.

Case 2: $|\delta_\theta(\tau_i)|$ increases when $\delta_\theta(\tau_i), \epsilon_{\tau_{i+1}} > 0$ and $\epsilon_{\tau_i} < 0$. This can be achieved as long as either $\epsilon_{\tau_{i+1}}$ increases or ϵ_{τ_i} becomes a smaller negative number. Eventually, at least one of $|\epsilon_{\tau_{i+1}}|$ and $|\epsilon_{\tau_i}|$ will increase.

Case 3: $|\delta_\theta(\tau_i)|$ increases when $\delta_\theta(\tau_i), \epsilon_{\tau_i} > 0$ and $\epsilon_{\tau_{i+1}} < 0$. This case is impossible to occur as $\delta_\theta(\tau_i)$ is positive initially.

Case 4: $|\delta_\theta(\tau_i)|$ increases when $\delta_\theta(\tau_i) > 0$ and $\epsilon_{\tau_i}, \epsilon_{\tau_{i+1}} < 0$. If both estimation error terms are negative and $\delta_\theta(\tau_i)$ is positive initially, it means that the magnitude of ϵ_{τ_i} is larger than that of $\epsilon_{\tau_{i+1}}$. If $\delta_\theta(\tau_i)$ increases to a greater positive number, ϵ_{τ_i} may become a smaller negative number. Therefore, $|\epsilon_{\tau_i}|$ will increase as well.

The remaining cases: $|\delta_\theta(\tau_i)|$ increases when $\delta_\theta(\tau_i) < 0$. For the remaining cases where $\delta_\theta(\tau_i)$ becomes a smaller negative number, we will reach the same conclusion as in the first four cases if we multiply both sides with the minus sign. Therefore, the absolute value of at least ϵ_{τ_i} or $\epsilon_{\tau_{i+1}}$ can still increase.

We should emphasize that this behavior may not always result in the deduction shown. However, we infer that there is always a possibility for it to occur under the existence of a TD error with a growing magnitude, which is the main proxy for the prioritized sampling scheme in PER.

□

Our assumption is realistic and probable to happen in practice, as it is grounded on the basic principles of off-policy actor-critic reinforcement learning. Specifically, the Q-value estimation error is affected by the discrepancy between the behavior policy and the target policy. If the TD error is large, it may suggest that the discrepancy between the policies is substantial, making it more probable that the Q-value estimation error will also be large [23, 24, 25]. Therefore, our assumption not only captures the practical difficulties that an agent may encounter but is also supported by the RL literature. For example, [3] and [26] discuss the relationship between TD error and Q-value estimation error, stating that a large absolute TD error indicates poor prediction and a need for stronger weight updates in the direction of the error. We then use our latter result to explain why policy optimization cannot be efficiently performed using transitions with large TD errors, following the assumption made. Specifically, Theorem 1 emphasizes the importance of transitions with minimal TD errors during policy optimization, as transitions with large errors can lead to suboptimal policies and cause the approximate policy gradient to deviate from its true value.

Theorem 1. *Let τ_i be a transition such that Assumption 1 is satisfied. Then, if the absolute temporal-difference error at time step i increases, the computed policy gradient will diverge from the true policy gradient for at least the current step i or the subsequent step $i+1$.*

Proof. We use Sutton's policy gradient theorem in the deep function approximation setting [27] to prove the result. First, we formally write the standard policy iteration with function approximation in terms of the policy parameters ϕ:

$$\phi \leftarrow \phi + \eta \nabla \phi, \tag{3.4}$$

where $\nabla \phi$ is the computed policy gradient, and η is the learning rate. The policy gradient with function approximation can be expressed in a general form as follows:

$$\nabla \phi \coloneqq \int_{\mathcal{S}} d^{\pi}(s) \int_{\mathcal{A}} \frac{\partial \pi(s,a)}{\partial \phi} Q_{\theta}(s,a), \tag{3.5}$$

where Q_θ is the function approximation of Q^π, and $d^\pi(s)$ is a discounted weighting of states encountered, starting from s_0 and then following π: $d^\pi(s) =$

$\sum_{t=0}^{\infty} \gamma^t p(s_t = s | s_0, \pi)$. The gradient in the latter equation is calculated by averaging over the continuous state and action spaces. However, this is often too expensive to compute. Instead, the gradient can be approximated over a subset of the state and action space, which corresponds to the sampled minibatch of transitions in the off-policy setting. We assume one-step gradient computation and ignore the gradient over the sampled minibatch of transitions for simplicity. For the state-action pair $(s_i, a_i) \in \tau_i$, the policy gradient $\nabla\phi(\tau_i)$ is proportional to $d^{\pi}(s_i)$ times the gradient of $\pi(s_i, a_i)$ with respect to ϕ and scaled by the estimated Q-value of (s_i, a_i). Let $k \coloneqq d^{\pi}(s_i)\frac{\partial \pi(s_i, a_i)}{\partial \phi}$, and we obtain:

$$\nabla\phi(\tau_i) = kQ_{\theta}(s_i, a_i), \tag{3.6}$$

$$\nabla\phi_{\text{true}}(\tau_i) = kQ^{\pi}(s_i, a_i), \tag{3.7}$$

for the calculated and true policy gradients, respectively. The same k applies to both the computed policy gradient and the true one since it does not depend on the Q-value estimates. Moreover, since k is constant over the range of values of the Q-value estimates, we can consider it as a constant for the proportionality between the policy gradients and Q-value estimates. As stated in Assumption 1, we have $Q_{\theta}(s_i, a_i) = Q^{\pi}(s_i, a_i) + \epsilon_{\tau_i}$. Using this, we can write:

$$\begin{aligned} \nabla\phi(\tau_i) - \nabla\phi_{\text{true}}(\tau_i) &= k(Q_{\theta}(s_i, a_i) - Q^{\pi}(s_i, a_i)), & (3.8) \\ &= k(Q^{\pi}(s_i, a_i) + \epsilon_{\tau_i} - Q^{\pi}(s_i, a_i)), & (3.9) \\ &= k\epsilon_{\tau_i}. & (3.10) \end{aligned}$$

Taking the absolute value of the latter equation provides:

$$|\nabla\phi(\tau_i) - \nabla\phi_{\text{true}}(\tau_i)| = |k\epsilon_{\tau_i}|, = |k||\epsilon_{\tau_i}| = k|\epsilon_{\tau_i}|. \tag{3.11}$$

Even though the k term is a variable, not a constant, we can still move it out of the absolute operator since it is a nonnegative number, being the product of probabilities. Hence, the above equation implies that the absolute difference between the approximate and true policy gradient is proportional to the absolute Q-value estimation error:

$$|\nabla\phi(\tau_i) - \nabla\phi_{\text{true}}(\tau_i)| \propto |\epsilon_{\tau_i}|, \tag{3.12}$$

where we do not use k as a constant of proportionality since it does not change over the Q-value estimates. Given that Assumption 1 holds, an increase in the absolute TD error in the current step results in a corresponding increase in the absolute estimation error in either the current or the subsequent step. (3.12) also indicates that an increasing in the absolute estimation error leads to a larger difference between the computed and actual policy gradient in the corresponding step. Therefore, we deduce that an increase in the absolute TD error causes the computed policy gradient to diverge from the actual policy gradient, at least in the current or the subsequent step. □

Following the proof of Theorem 1, Corollary 1 formally states our finding.

Corollary 1. *An increase in the absolute temporal-difference error can lead to a less accurate approximate policy gradient, causing it to deviate from the true policy gradient corresponding to the optimal Q-function, at least for the current or subsequent transition.*

This is a key component in the degragated performance of the prioritized sampling when an actor network is used. We now discuss a recent finding that provides an insight for the suboptimal empirical performance of PER and supports our investigation. As we mentioned before, prioritizing transitions with high TD error by stochastic sampling causes a shift in the distribution of s' to $\mathbb{E}_{s' \sim P_\pi, a' \sim \pi}[Q(s', a')]$. Therefore, this introduced bias is corrected by importance sampling expressed in (2.11). However, [8] claimed that PER does not completely remove the bias and may favor outliers when combined with MSE loss in the Q-network updates. Specifically, when MSE is used together with PER, optimizing it using a loss of $\frac{1}{\rho}|\delta_\theta(\tau)|^\rho$, where $\rho + \varepsilon - \varepsilon\upsilon \neq 2$, results in bias in the Q-network target and consequently, biased Q-network updates. Remark 1 highlights this finding, which was recognized by [8] as one of the factors that contribute to the suboptimal performance of PER when used with standard actor-critic algorithms in continuous action domains.

Remark 1. *The PER objective is biased if $\rho + \varepsilon - \varepsilon\upsilon \neq 2$ under the loss function $\frac{1}{\rho}|\delta_\theta(\tau)|^\rho$ [8].*

According to Remark 1, [8] concluded that prioritized sampling is prone to bias when used with MSE because the condition $\rho + \varepsilon - \varepsilon \upsilon \neq 2$ is never exactly satisfied, i.e., if $\upsilon < 1$, then $2+\varepsilon-\varepsilon\upsilon > 2$ for $\varepsilon \in (0, 1]$ for $\varepsilon \in (0, 1]$. Furthermore, the introduced bias is not always constant. Fortunately, an L1 loss can satisfy this property such that $1 + \varepsilon - \varepsilon\upsilon \in [1, 2]$ for $\varepsilon \in (0, 1]$ and $\upsilon \in [0, 1]$ since $\rho = 1$. However, the L1 loss may not be ideal in practice since each update involves a constant step size, which may cause the objective to be overshot if the learning rate is too high. To remedy this, [8] chose the commonly used Huber loss with $\kappa = 1$ in the Q-network updates instead of the MSE loss:

$$\mathcal{L}_{\text{Huber}}(\delta_\theta(\tau_i)) = \begin{cases} 0.5\delta_\theta(\tau_i)^2 & \text{if } |\delta_\theta(\tau_i)| \leq \kappa, \\ |\delta_\theta(\tau_i)| & \text{otherwise.} \end{cases} \tag{3.13}$$

The Huber loss switches from L1 to MSE and adjusts the gradient accordingly as $\delta_\theta(\tau_i)$ approaches zero when the error is lower than threshold 1. Also, MSE is used when $|\delta_\theta(\tau_i)| < 1$. Therefore, to avoid the bias caused by MSE and prioritization, samples with an error below 1 should be sampled uniformly. The Loss-Adjusted Prioritized Experience Replay (LAP) [8] algorithm achieves this by clipping samples with low priority to at least 1 using the prioritization scheme: $p(\tau_i) = \max(|\delta_\theta(\tau_i)|^\varepsilon, 1)$. Further, [8] addressed the problem described in Remark 1 by combining a modified stochastic prioritization scheme with the Huber loss:

$$p(\tau_i) = \frac{\max(|\delta_\theta(\tau_i)|^\varepsilon, 1)}{\sum_j \max(|\delta_\theta(\tau_j)|^\varepsilon, 1)}. \tag{3.14}$$

Our theoretical findings for the suboptimal performance of PER, which we highlight in Remark 2, are consistent with the results reported by [8]. We point out that our focus is on the off-policy actor-critic algorithms where a separate actor network chooses actions to maximize the Q-network's action-value estimate. Contrarily, Remark 2 covers both discrete control and off-policy actor-critic deep RL. There might be other reasons for the suboptimal performance of PER such as inaccurate Q-value estimates. However, those reasons are not specific to PER and are caused by the variants of the DQN algorithm. Thus, to our knowledge, Corollary 1 and Remark 2 provide the foundation for the algorithmic limitations of PER in off-policy actor-critic.

Remark 2. *To further eliminate the bias favoring the outlier transitions in the Prioritized Experience Replay algorithm, the Huber loss with* $\kappa = 1$ *should be used in conjunction with the prioritization scheme expressed in* (3.14) *[8].*

3.2 Adaptation of Prioritized Experience Replay to Actor-Critic Algorithms

We now introduce a set of novel modifications to PER for addressing the problems identified in the prioritized sampling when used with actor-critic algorithms.

3.2.1 Inverse Sampling for the Actor Network

We begin with Corollary 1, which implies that training the actor network with a transition that has a large TD error can make the approximate policy gradient deviate from the true gradient, at least for the current or subsequent transitions. However, we should emphasize that the performance may not necessarily deteriorate under the TD error-based prioritization scheme if the policy gradient deviates only for next transitions that do not have a large TD error. However, it is unlikely to occur, as the replay buffer usually contains only a few transitions with large TD errors in the initial optimization steps. These transitions are typically fewer than the batch size, so they are still likely to be sampled.

Observation 1. *The PER algorithm may not affect the performance of actor-critic methods if the actor network is not optimized with the transitions that are subsequent to the sampled transitions. However, this is a rare case in the standard off-policy actor-critic algorithms.*

We suggest optimizing the actor network with transitions that have small TD errors to address the problem identified in Corollary 1. To this end, we use inverse sampling from the prioritized replay buffer, which means sampling transitions with low TD errors for the actor network using the sampling scheme of PER. To apply this method, we examine the data structure of PER. A common and efficient implementation of PER, which corresponds to *proportional prioritization*, is based on a "sum-tree" data structure. This sum-tree data structure is very similar to the array representation of a binary heap, with the main difference being that a parent node's value is equal to the sum of its children instead of the standard heap property. The internal nodes of the sum-tree data structure are intermediate sums, with the parent node carrying the sum over all priorities p_{total}. Concurrently, the leaf nodes store the priorities of the transitions. This allows for $\mathcal{O}(\log|R|)$ updates and sampling while computing the cumulative total of priorities. For sampling a minibatch of size N, the range $[0, p_{\text{total}}]$ is evenly divided into N ranges. A value is uniformly sampled within each range. The sum-tree data structure is then queried for the transitions that match each sampled value.

An intuitive approach to sample transitions with a probability inversely proportional to the TD error is to build a new sum-tree that contains the global inverse of the priorities. While priorities in vanilla PER are updated for every training step through the previously defined sum tree data structure, inverse sampling for actor updates requires building a new sum tree before training. In every update step, the priorities for the actor network are computed using:

$$I \sim \tilde{p}(\tau_i) = \frac{p_{\max}}{p(\tau_i)} = \max_i \left(\frac{\max(|\delta_\theta(\tau_i)|^\varepsilon, 1)}{\sum_j \max(|\delta_\theta(\tau_j)|^\varepsilon, 1)} \right) \times \frac{\sum_j \max(|\delta_\theta(\tau_j)|^\varepsilon, 1)}{\max(|\delta_\theta(\tau_i)|^\varepsilon, 1)}, \quad (3.15)$$

where $p(\tau_i)$ is the priority of the i^{th} transition and $p_{\max}$ is the maximum of the previously determined priorities of the stored transitions. As we emphasize in Remark 2, the use of MSE with PER may still lead to different biases that could potentially favor outlier transitions. To address this issue, we use the prioritization scheme proposed by the LAP algorithm, given by (3.14), in (3.15). Observe that (3.15) does not change proportional prioritization. The relative proportions (i.e., the largest over the smallest) do not change as we take the inverse by multiplication.

This is the main component of our approach. To reduce the effect of outlier bias also in the Q-network updates, we again use the Huber loss function with a tuning parameter of $\kappa = 1$, as defined in equation (3.13), similar to the LAP algorithm. Therefore, during each optimization step, both the Q-network and priorities are updated using the following expressions, respectively:

$$I \sim p(\tau_i) = \frac{\max(|\delta_\theta(\tau_i)|^\varepsilon, 1)}{\sum_j \max(|\delta_\theta(\tau_j)|^\varepsilon, 1)}, \tag{3.16}$$

$$\theta \leftarrow \theta - \eta \cdot \frac{1}{|I|} \sum_{i \in I} \nabla_\theta \mathcal{L}_{\text{Huber}}(\delta_\theta(\tau_i)), \tag{3.17}$$

$$p(\tau_i) \leftarrow \max(|\delta_\theta(\tau_i)|^\varepsilon, 1) \text{ for } i \in I, \tag{3.18}$$

where I denotes the indices of the prioritized transitions that are sampled to form the minibatch. Note that the clipping reduces the chance of dead transitions when $p(\tau_i) \approx 0$, which eliminates the need for the μ parameter. It is important to note that the Huber loss function cannot be used in the computation of the policy gradient for several reasons. First, the priorities are determined by the loss function of the Q-network, which is the TD error, and the use of MSE loss with TD error-based prioritized sampling is the main cause of the mentioned outlier bias. Also, the policy loss and gradient are computed using a class of policy gradient methods that cannot be replaced by the Huber loss. Therefore, the outlier bias does not affect the policy gradient, and there is no need to use the Huber loss in the policy network.

3.2.2 Optimizing the Actor and Critic with a Shared Set of Transitions

In some cases, optimizing the actor and critic networks with completely different transitions can potentially violate the actor-critic theory. Typically, the features used by the critic network are dependent on the actor parameters and policy gradient since the actor determines the actions that ultimately lead to the observed state space [15]. An important corollary to this observation is that if the critic is updated using a set of features that exist in a state-action space where the actor is never optimized, this may lead to significant instability. This is because the transitions used by the critic are processed through the actor, and if the actor never sees these transitions, the updates of the critic may not accurately reflect the actual performance of the actor [15]. Therefore, the action evaluations of the critic might become questionable.

Intuitively, this situation can occur when inverse the prioritized and prioritized sampling are used for the actor and critic networks, respectively, since they may never be optimized with the same transitions. The reason for this is that the TD error of the critic's samples may not decrease to the extent that the actor is unable to observe them. We pointed out that there is not always a direct correlation between TD and estimation errors. Hence, some of the transitions might initially have low TD errors. If the actor is optimized with respect to these low TD error transitions throughout the learning and the Q-network only focuses on the remaining large TD error transitions, samples used in the actor and critic training may not be the same. Although this remains unlikely, we nevertheless prevent it by updating the actor and critic networks through a set of shared transitions, being a fraction of the sampled minibatch of transitions. However, we could not know the value of such a fraction, and we will introduce it as a hyperparameter later. We emphasize these deductions in Observation 2 and regulate our approach accordingly.

Observation 2. *If the transitions for the actor and critic are sampled through inverse prioritized and prioritized sampling, respectively, they may never see the same transitions. This, in turn, violates the actor-critic theory [15] and can lead to unstable learning. Hence, the actor and critic should be optimized with the same set of transitions, at least for a fraction of the sampled minibatch of transitions, in every update step.*

We examine the following sampling alternatives in terms of the TD error for choosing the set of shared transitions. We can also explore auxiliary variables driven by the components in model-based RL to compute the scores of the experiences, similar to the work of [5]. However, learning additional features introduces additional computational overhead. This aspect, however, is beyond the scope of this study, as our main focus is only on prioritization regarding the TD error and corrections to vanilla PER in actor-critic methods.

- **Transitions with large TD error:** The actor network cannot be optimized with experiences that have large TD error since the policy gradient significantly deviates from the true gradient, as discussed in Corollary 1.
- **Transitions with small TD error:** Learning from the experiences with small TD error can be beneficial, yet they decrease the sampling efficiency and waste resources since the Q-network has little to learn from small TD error transitions in the sense of prioritized sampling.
- **Uniformly sampled transitions:** The previous two alternatives imply that uniform sampling for the set of shared transitions can be a suitable choice. Although large TD error transitions might be included in the uniformly sampled minibatch, their effects are reduced due to averaging in the minibatch learning.

We conclude that uniform sampling for a fraction of the transitions in the sampled minibatch is a suitable approach for addressing the issue highlighted in Observation 2. While we could also consider using transitions with an average magnitude of TD errors, the use of uniform sampling already encompasses transitions with a mean TD error in the expectation. Furthermore, relying on random sampling allows for the inclusion of transitions that cannot be sampled by prioritized and inverse prioritized sampling. However, other distributions, such as those that encourage mean tendency/variance reduction, could be promising directions for future research.

As discussed in Remark 2, the combination of Huber loss ($\kappa = 1$) with the prioritized sampling can eliminate outlier bias in the LAP algorithm. The authors also introduced the mirrored loss function of LAP, with an equivalent expected gradient, for uniform sampling from the experience replay buffer. To observe the same benefits of LAP also in the uniform sampling counterpart, its mirrored loss function, Prioritized Approximate Loss (PAL), should be used instead of MSE.

Similar to the case of the LAP function, the PAL loss is also not used in the policy network's updates. The PAL function is expressed as:

$$\varrho = \frac{\sum_j \max(|\delta_\theta(\tau_j)|^\varepsilon, 1)}{N}, \tag{3.19}$$

$$\mathcal{L}_{\mathrm{PAL}}(\delta_\theta(\tau_i)) = \frac{1}{\varrho}\begin{cases} 0.5\delta_\theta(\tau_i)^2 & \text{if } |\delta_\theta(\tau_i)| \leq 1, \\ \frac{|\delta_\theta(\tau_i)|^{1+\varepsilon}}{1+\varepsilon} & \text{otherwise.} \end{cases} \tag{3.20}$$

Remark 3. *To eliminate the outlier bias in the uniform sampling counterpart, the Prioritized Approximate Loss (PAL) function should be used, which has the same expected gradient as the Huber loss when combined with PER [8].*

With the latter component included, this forms our PER correction algorithm, **L**oss-**A**djusted **A**pproximate **A**ctor **P**rioritized Experience Replay (LA3P). In summary, our approach involves uniformly sampling a minibatch of transitions, with a size of $\lambda \cdot N$ in each update step. Here, $\lambda \in [0, 1)$ represents the fraction of the transitions that are uniformly sampled and serves as the only introduced hyperparameter in our approach. Using the uniformly sampled batch, the critic and actor networks are optimized consecutively. The critic is updated based on the PAL function expressed in (3.19), and the priorities are updated immediately after. Following this, a total of $(1 - \lambda) \cdot N$ transitions are sampled through prioritized and inverse prioritized sampling for the critic and actor networks each. The critic is then optimized using the Huber loss ($\kappa = 1$) expressed in (3.13), while a policy gradient technique optimizes the actor. Lastly, the priorities are updated again. Overall, both the actor and critic networks are optimized with N transitions per update step, similar to standard off-policy actor-critic algorithms. Note that we update the priorities also in the uniform counterpart since the score of the transitions should be up-to-date whenever possible. In addition, the order of the prioritized and uniform updates does not change the expected gradient. Hence, it is not important which of them is used first. Nevertheless, in our implementation and experiments, we precisely use the structure outlined in Algorithm 1, denoted through a generic application to off-policy actor-critic methods.

Algorithm 1 Off-Policy Actor-Critic with Loss-Adjusted Approximate Actor Prioritized Experience Replay (LA3P)

Input: Minibatch size N, exponents ε and υ, uniform sampling fraction λ, target step-size ζ, and actor and critic step-sizes η_π and η_Q
Initialize actor π_ϕ and critic Q_θ networks, with random parameters ϕ and θ
Initialize target networks $\phi' \leftarrow \phi$, $\theta' \leftarrow \theta$, if required
Initialize $p_{\text{init}} = 1$ and the experience replay buffer $\mathcal{R} = \emptyset$
for $t = 1$ **to** T **do**
  Select action a_t and observe reward r_t and new state s_{t+1}
  Store the transition tuple $\tau_t = (s_t, a_t, r_t, s_{t+1})$ in $\mathcal{R}$ with initial priority $p_t = p_{\text{init}}$
  for each update step **do**
    Uniformly sample a minibatch of transitions:
    $I \sim p(\tau_i) = \frac{1}{|R|}$ where $|I| = \lambda \cdot N$
    Optimize the critic network: $\theta \leftarrow \theta - \eta_Q \cdot \frac{1}{|I|} \sum_{i \in I} \nabla_\theta \mathcal{L}_{\text{PAL}}(\delta_\theta(\tau_i))$
    Compute the policy gradient $\nabla\phi(\tau_i)$ τ_i where $i \in I$
    Optimize the actor network: $\phi \leftarrow \phi + \eta_\pi \cdot \frac{1}{|I|} \sum_{i \in I} \nabla\phi(\tau_i)$
    Update the priorities of the uniformly sampled transitions:
    $p(\tau_i) \leftarrow \max(|\delta_\theta(\tau_i)|^\varepsilon, 1)$ for $i \in I$
    Update target networks if required: $\theta' \leftarrow \zeta\theta + (1-\zeta)\theta'$, $\phi' \leftarrow \zeta\phi + (1-\zeta)\phi'$
    Sample a minibatch of transitions through prioritized sampling:
    $I \sim p(\tau_i) = \frac{\max(|\delta_\theta(\tau_i)|^\varepsilon, 1)}{\sum_j \max(|\delta_\theta(\tau_j)|^\varepsilon, 1)}$ where $|I| = (1-\lambda) \cdot N$
    Optimize the critic network: $\theta \leftarrow \theta - \eta_Q \cdot \frac{1}{|I|} \sum_{i \in I} \nabla_\theta \mathcal{L}_{\text{Huber}}(\delta_\theta(\tau_i))$
    Update the priorities of the prioritized transitions:
    $p(\tau_i) \leftarrow \max(|\delta_\theta(\tau_i)|^\varepsilon, 1)$ for $i \in I$
    Sample a minibatch of transitions through inverse prioritized sampling:
    $I \sim \tilde{p}(\tau_i) = \frac{p_{\max}}{p(\tau_i)} = \max_i \left(\frac{\max(|\delta_\theta(\tau_i)|^\varepsilon, 1)}{\sum_j \max(|\delta_\theta(\tau_j)|^\varepsilon, 1)} \right) \times \frac{\sum_j \max(|\delta_\theta(\tau_j)|^\varepsilon, 1)}{\max(|\delta_\theta(\tau_i)|^\varepsilon, 1)}$
    Compute the policy gradient $\nabla\phi(\tau_i)$ for τ_i where $i \in I$
    Optimize the actor network: $\phi \leftarrow \phi + \eta_\pi \cdot \frac{1}{|I|} \sum_{i \in I} \nabla\phi(\tau_i)$
    Update target networks if required: $\theta' \leftarrow \zeta\theta + (1-\zeta)\theta'$, $\phi' \leftarrow \zeta\phi + (1-\zeta)\phi'$
  end for
end for

3.2.3 Complexity Analysis

The LA3P framework adds an extra sum tree, the LAP, and PAL functions to the basic PER. These modifications only affect the sampled batches of transitions, so their computational complexity is overshadowed by the extra sum tree, which operates on the entire replay buffer. Furthermore, the extra sum tree needs a priority update. The complexity of setting the node priorities is the same as in PER. Also, LA3P inverts the priorities by multiplication, which takes $\mathcal{O}(|R|)$ time. Hence, LA3P runs in $\mathcal{O}(\log|R|) + \mathcal{O}(|R|)$. Since $\mathcal{O}(|R|)$ is greater than $\mathcal{O}(\log|R|)$, we deduce that LA3P has a worst-case runtime of $\mathcal{O}(|R|)$.

The additive sum tree in our approach requires taking the inverse of the priorities by multiplication, which greatly increases the computational complexity of the basic PER. This may make our approach seem impractical. However, this problem can be solved by the Single Instruction, Multiple Data (SIMD) structure that modern CPUs implement. Specifically, SIMD instructions can do the same operation, such as the array division needed in our case, on all cores at the same time. Fortunately, this implementation detail is not the user's responsibility and can be done automatically by the CPU. Therefore, the extra computational efficiency that the SIMD instructions introduce will considerably lower the computational load of the LA3P framework.

3.3 Experiments

With all the mentioned concepts combined, we explore how much our prioritization framework can enhance the performance of off-policy actor-critic methods. To do this, we first survey the existing studies in the literature, which we compare our method with. Next, we present the experimental details and report the results. Lastly, we discuss the results obtained.

3.3.1 Competing Methods

We compare our proposed method with four baseline algorithms. Naturally, the first one is the PER algorithm, which we modify in several ways to design LA3P. The second one is LAP, which we use as a part of our framework and extend it further. Although PAL combined with uniform sampling could also be used for comparison, [8] have shown that it has the same expected gradient as LAP, implying similar empirical performance. The third one is the uniform sampling from the buffer, where each sample has an equal chance of being selected. This is a common baseline for deep RL methods that do not propose a novel experience replay mechanism. Therefore, comparing with uniform sampling will show us the full improvement that the inverse prioritized sampling can provide. The last technique is the Model-Augmented PER (MaPER) algorithm [5], which is also one of the few studies that addressed the limitations of PER. Specifically, the authors argue that heavily relying on the TD error (or Q-network's error) to sample from the replay buffer may be ineffective due to the possible under- or overestimation of the Q-values caused by the deep function approximation and bootstrapping. To overcome this, the authors propose to learn auxiliary features from the model-based RL components to score the experiences. This is done by using the critic network to enhance the curriculum learning effect for predicting Q-values with minimal memory and computational cost compared to vanilla PER.

3.3.2 Experimental Details

We assess the effectiveness of LA3P on the standard set of MuJoCo [12] and Box2D [13] continuous control tasks interfaced by OpenAI Gym [14]. We test the effectiveness of our method by combining it with the state-of-the-art off-policy actor-critic algorithms, namely Soft Actor-Critic (SAC) [28] and Twin Delayed Deep Deterministic Policy Gradient (TD3) [22].

Our implementation of the state-of-the-art algorithms closely follows the hyperparameter setting and architecture described in the original papers. Specifically, we implement TD3 using the code from the author's GitHub repository[1], which contains the fine-tuned version of the algorithm. The implementation of SAC is precisely based on the original paper. Different from the paper, we include entropy tuning, as demonstrated by [29] to improve the algorithm's overall performance. Furthermore, we add 25000 exploration time steps before the training to increase the data efficiency.

We implement the algorithm and our framework using the LAP and PAL code in the author's GitHub repository[2], which needs a few lines on top of the standard PER implementation. Furthermore, we use the same repository for the PER implementation, which is based on proportional prioritization through sum trees. To implement LA3P, we combine uniform sampling with PAL and PER with LAP. For all experience replay sampling algorithms, except for uniform, we set $\upsilon = 0.4$, in line with the original papers. We also set $\varepsilon = 0.6$ and $\varepsilon = 0.4$ for PER and LAP, respectively. Since we use LAP and PAL functions in our framework, we set $\varepsilon = 0.4$ for LA3P. The implementation of MaPER is obtained from the code available on the submission website[3]. More details on the exact hyperparameter settings, architecture, and implementation can be found in Appendix A.1 and A.2.

Each method is trained for a million steps over ten random seeds of network initialization, simulators, and dependencies, except for the Ant, HalfCheetah, Humanoid, and Swimmer environments. We observed that the algorithms could improve from further training in these environments, and thus, we train them for 2 million steps. Note that we use a replay buffer size equal to the number of training steps in all experiments. Every 1000 steps, each method is evaluated in a separate evaluation environment (training seed + constant) for ten episodes, where no exploration and learning are performed. To construct the reported learning curves, we compute the average performance over ten evaluation episodes at each

[1] `https://github.com/sfujim/TD3`

[2] `https://github.com/sfujim/LAP-PAL`

[3] `https://openreview.net/forum?id=WuEiafqdy9H`

evaluation period. For easy reproducibility and fair evaluation, we did not change the environment dynamics and reward functions of the simulators. Computing infrastructure (i.e., hardware and software) used to produce the reported results are summarized in our repository[4]. Detailed experimental setup is provided in Appendix A.

For all tasks, we use uniform fraction value of $\lambda = 0.5$. Initially, we tested $\lambda = \{0.1, 0.3, 0.5, 0.7, 0.9\}$ on Ant, HalfCheetah, Humanoid, and Walker2d and found that $\lambda = 0.5$ produced the best results. In the next section, we also present a sensitivity analysis based on the λ parameter. Empirical complexity analysis is provided in Appendix A.4. Note that for some of the tasks (e.g., HalfCheetah, Hopper, Walker2d) the baseline competing algorithms performed worse than what was reported in the original articles. This is due to the stochasticity of the simulators and used random seeds. Nevertheless, regardless of where the baselines converge, the performance disparity between the competing approaches would practically remain the same if we used different sets of random seeds.

3.4 Results

3.4.1 Comparative Evaluation

Figures 3.1 and 3.2 report the learning curves for the selected set of OpenAI Gym continuous control environments for the SAC and TD3 algorithms, respectively. Based on the learning plots, we notice that LA3P performs comparably or better than the competing approaches in the majority of the tasks tested. However, in the BipedalWalker and LunarLanderContinuous environments under the SAC algorithm, LAP achieves slightly higher rewards than our method. Nevertheless, these differences are negligible, and as shown by the learning curves, LA3P converges faster. In these relatively trivial environments, we observe only a minor improvement. However, in the Swimmer environment, where no algorithm could

[4]`https://github.com/baturaysaglam/LA3P`

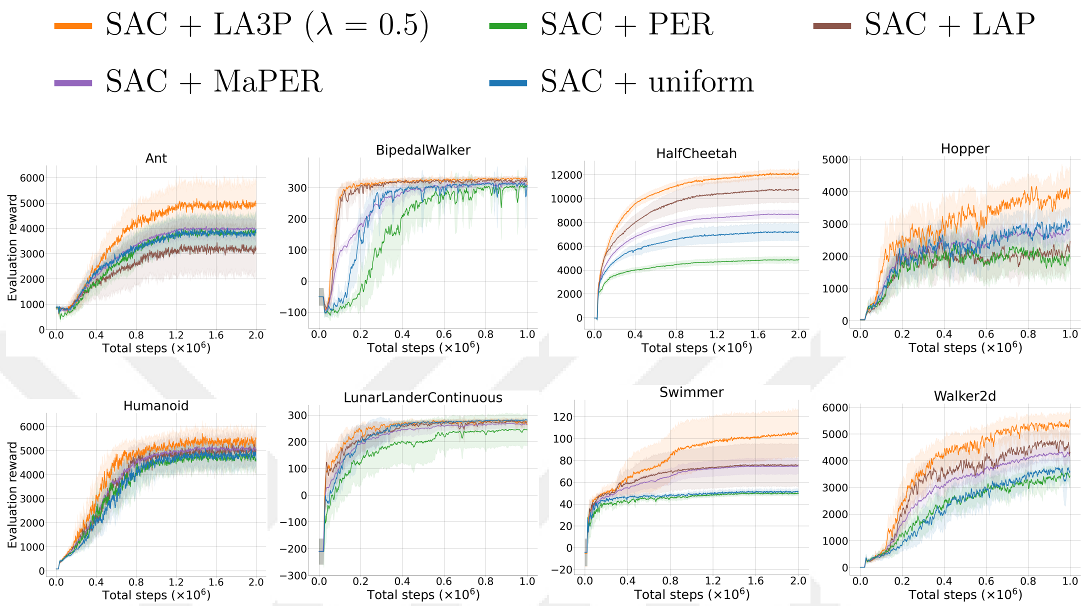

Figure 3.1: The learning curves for the set of MuJoCo and Box2D continuous control tasks under the SAC algorithm. The shaded region represents a 95% confidence interval over 10 trials. A sliding window of size 5 smooths the curves for visual clarity.

converge, the performance gains resulting from our modifications are particularly notable. Furthermore, we also observe a significant improvement by LA3P in the HalfCheetah environment relative to the prior approaches. As HalfCheetah and Swimmer are considered "stable", that is, episodes terminate only after a prespecified number of time steps have been reached, these tasks entail the simulation of long horizons. Consequently, as noted by [8], the advantages of a corrected prioritization scheme are more prominent in environments with extended horizons.

In contrast to the mean curves, the shaded areas show a high degree of overlapping confidence intervals, which may suggest that the results obtained are not significant. However, it is generally accepted that this type of hypothesis test is weak. To establish whether the performance improvement provided by LA3P is substantial, we have followed the statistical testing approach described by [30] to analyze our results in depth. A 2-sample t-test with a confidence interval of 0.95 is recommended as an appropriate choice to compare the performance of two algorithms. To this end, we have conducted pairwise comparisons of the

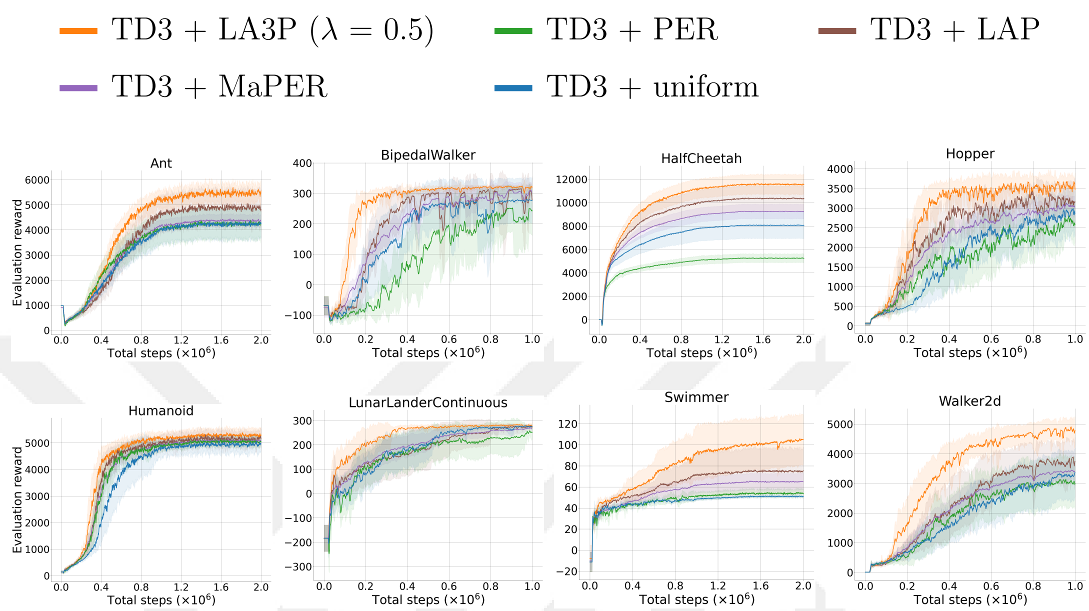

Figure 3.2: The learning curves for the set of MuJoCo and Box2D continuous control tasks under the TD3 algorithm. The shaded region represents a 95% confidence interval over 10 trials. A sliding window of size 5 smooths the curves for visual clarity.

last 10 evaluation rewards obtained by our algorithm and those obtained by each competing method. The resulting p-values are reported in Table 3.1. Our null hypothesis is that the difference between the last 10 evaluation returns (i.e., where the algorithm converge) of two algorithms is statistically insignificant. A p-value less than 0.05 indicates that there is sufficient evidence to reject the null hypothesis, leading us to conclude that the mean of the last 10 rewards for our algorithm is greater than that for the competing algorithm with 95% confidence. On the contrary, if the p-value is greater than 0.05, we fail to reject the null hypothesis, and we cannot conclude that there is a significant difference the converged levels of the two algorithms.

The statistical analysis using 2-sample t-tests confirms that the performance improvement offered by LA3P is statistically significant, with p-values lower than 0.05 in most of the domains tested, except for the LunarLanderContinuous and BipedalWalker environments, which are relatively trivial. It should also be noted

	Environment	p_{LAP}	p_{MaPER}	p_{PER}	p_{uni}
SAC	Ant	0.002	0.011	0.012	0.016
	BipedalWalker	0.539	0.187	0.106	0.129
	HalfCheetah	0.009	0.000	0.000	0.000
	Hopper	0.000	0.000	0.000	0.001
	Humanoid	0.014	0.000	0.000	0.000
	LunarLanderContinuous	0.744	0.335	0.191	0.992
	Swimmer	0.017	0.006	0.000	0.000
	Walker2d	0.000	0.000	0.000	0.000
TD3	Ant	0.024	0.000	0.000	0.000
	BipedalWalker	0.084	0.107	0.024	0.107
	HalfCheetah	0.029	0.000	0.000	0.000
	Hopper	0.032	0.001	0.000	0.007
	Humanoid	0.263	0.054	0.038	0.018
	LunarLanderContinuous	0.313	0.045	0.033	0.350
	Swimmer	0.026	0.002	0.000	0.000
	Walker2d	0.004	0.000	0.000	0.001

Table 3.1: The resulting p-values from a 2-sample t-test performed over the last 10 evaluation returns of LA3P and the competing methods over 10 trials under SAC and TD3. Subscripts denote the competing method with which LA3P is compared. A p-value less than the significance level of 0.05 indicates that the difference in performance is statistically significant. Values are rounded up to three decimal points.

that the difference between LA3P and LAP algorithms remains statistically insignificant in the Humanoid task under the TD3 algorithm. However, these findings provide strong evidence that LA3P is a promising approach that can offer better performance compared to other methods. Therefore, the reward curve plots that show LA3P outperforming the competing approaches can be confidently supported by the results of the t-test per the deep RL benchmarking standards [30], indicating the robustness and reliability of the experimental findings.

We also verify the findings of previous empirical studies by [8] and [5], that off-policy actor-critic methods do not benefit from PER and their performance usually drop. In the prior work [8], this poor performance is attributed to the MSE loss, yet we observe that using the corrected loss function (i.e., LAP) does not make much difference in Ant, Hopper, and Walker2d compared to LA3P.

Moreover, the SAC algorithm performs worse in Ant and Hopper. This is in line with our theoretical analysis in Corollary 1, that is, optimizing the actor network with transitions that have large TD errors can lead to the divergence of the approximate policy gradient from the one computed under the optimal Q-function, even if the loss function is corrected. In these environments, the learning curves show that the main source of performance improvement for LA3P is the inverse prioritized sampling for the actor network. This implies that the inverse sampling in the LA3P framework has a more significant role than the LAP and PAL functions that we use. Therefore, a combined solution, inverse sampling with corrected loss functions, performs better.

Moreover, it is observed that MaPER does not perform well as expected since it only achieves slight improvements over PER. We mainly attribute this suboptimal performance to the model prediction structure of the algorithm. As we discussed earlier, the MaPER algorithm relies on new learnable features derived from the components in model-based RL to compute the scores on experiences since critic networks often under- or overestimate Q-values. However, the Clipped Double Q-learning algorithm proposed by [22] already addresses the issues of inaccurate Q-value estimates, which is already employed in SAC and TD3. Therefore, we infer that the main drawback of PER is not the inaccurate Q-value estimates used in the priority computations but the biased loss function and training the actor network with transitions that have large TD errors. Additionally, the model prediction module in MaPER decreases the convergence rate, yet adds notable stability to the learning. Nevertheless, the resulting performance is not significant. Hence, we believe that LA3P is a preferable and comprehensive way of overcoming the underlying issues of PER in actor-critic methods.

3.4.2 Ablation Studies

We conduct an ablation study to better understand the contribution of each component in LA3P. The LA3P algorithm introduces several modifications to PER. In summary, LA3P consists of: *(i)* inverse sampling for the actor, *(ii)* uniform sampling for the actor and critic networks to share a set of transitions, *(iii)* the LAP function applied to the prioritized transitions, and *(iv)* the PAL function applied to the uniformly sampled transitions.

We examine and discuss the performances obtained upon removing each of these components. Hence, we test the performance of LA3P in cases where the shared transitions are low TD error experiences, instead of uniformly sampled ones, to demonstrate the reduced data efficiency previously mentioned. Furthermore, we perform a sensitivity analysis for the λ parameter. We do not remove the inverse sampling as it is the backbone of our algorithm, that is, removing the inverse sampling for the actor network would not relate to any of the modifications introduced in this work as it would be only a combination of uniform and prioritized sampling. We choose four challenging environments with different characteristics for a comprehensive inference. As described by [30] and [8], we consider the stable environment HalfCheetah, the unstable environment Walker2d, and the high-dimensional Ant and Humanoid environments. The latter two are widely considered to be among the most challenging environments in the MuJoCo suite [8].

Table 3.2 shows the average of the last ten evaluation returns over ten trials for our ablation studies and sensitivity analysis. The same experimental setup is used to perform the ablation studies, and $\lambda = 0.5$ is used for all experiments unless otherwise stated. Note that $\lambda = 0.0$ results in the LA3P setting without shared set of transitions and the evaluation results of which are already given in Table 3.2. Moreover, $\lambda = 1.0$ corresponds to uniform sampling, which we already compare with LA3P in Figures 3.1, 3.2, and Table 3.1.

First, we deduce that the set of shared transitions is the most crucial component of our framework. Independently training the actor and critic networks violate their correlation as the actor is optimized by maximizing the Q-values estimated by the Q-network, and the Q-network is trained using the actions selected by the actor. Thus, they should not be separated in training, and we empirically verify Observation 2. Although the LAP and PAL functions apply the same number of transitions in each update step (i.e., $\lambda = 0.5$), we observe that the contribution of LAP is more significant than PAL. As discussed in our comparative evaluations, the performance improvement by LA3P largely relies on inverse sampling for the actor network. As expected, correcting the prioritization in inverse sampling for the actor network through the LAP approach, i.e., (3.15), is more crucial than correcting the loss by PAL for the uniformly sampled batch. Lastly, we infer that using low TD error transitions instead of uniformly sampled ones substantially degrades the performance. Although this setting would seem to be a reasonable choice at a first glance, the data efficiency notably decreases as the Q-network repeatedly trains with transitions that it has already learned well. In expectation, the uniformly sampled batch of transitions corresponds to an intermediate TD error value compared to the transitions contained in the entire replay buffer. As we experimentally show, this may benefit both the actor and critic networks since inverse prioritized and prioritized sampling may not include transitions with intermediate TD error values, compared to the rest of the experiences.

Our sensitivity analysis on the λ parameter suggests that $\lambda = 0.5$ produces the best results across most of the environments by a considerable margin. As λ decreases, the correlation between the actor and critic networks starts to be ignored, and the performance drops. Also, the larger λ values yield the performance to converge to that of uniform sampling. Hence, it is shown that the introduced framework does not require intensive hyperparameter tuning, and $\lambda = 0.5$ can apply to many tasks.

Setting	**Ant**	**HalfCheetah**	**Humanoid**	**Walker2d**
LA3P (complete)	**5197.46 ± 162.04**	**11225.14 ± 800.59**	**5131.11 ± 193.26**	**4776.68 ± 339.80**
Low TD error	3485.06 ± 834.26	10992.05 ± 467.13	3938.13 ± 1395.00	4438.29 ± 320.47
w/o LAP	3408.55 ± 569.04	4580.27 ± 250.82	3585.06 ± 830.28	3262.49 ± 252.69
w/o PAL	3975.29 ± 1130.62	7560.5 ± 762.22	4879.43 ± 187.23	4543.92 ± 398.97
w/o Uniform sampling	4431.87 ± 716.48	10483.11 ± 594.31	5058.69 ± 111.42	4254.55 ± 387.41
$\lambda = 0.1$	3768.74 ± 1007.13	11203.83 ± 550.87	4695.52 ± 99.62	4562.77 ± 372.07
$\lambda = 0.3$	4903.34 ± 385.48	10460.52 ± 933.14	4528.28 ± 1113.55	4425.08 ± 324.29
$\lambda = 0.5$	**5197.46 ± 162.04**	**11225.14 ± 800.59**	**5131.11 ± 193.26**	**4776.68 ± 339.80**
$\lambda = 0.7$	4383.97 ± 828.71	10656.92 ± 735.91	4874.5 ± 130.33	4253.76 ± 829.14
$\lambda = 0.9$	3882.08 ± 936.85	10547.85 ± 871.63	3757.9 ± 1344.11	4545.97 ± 567.28

Table 3.2: The average return over the last 10 evaluations over 10 trials of 1 million time steps, comparing ablation over LA3P under low TD error shared transitions, LA3P without the LAP function, LA3P without the PAL function, LA3P without the shared set of transitions, and LA3P under $\lambda = \{0.1, 0.3, 0.5, 0.7, 0.9\}$. ± captures a 95% confidence interval over the trials. Bold values represent the best-performing configuration under each environment in terms of the mean rewards. The TD3 algorithm is used as the underlying actor-critic algorithm.

Chapter 4

Deep Intrinsically Motivated Exploration

The content of this chapter reflects the works described in the following studies:

- B. Saglam, F. B. Mutlu, and S. S. Kozat, "An optimistic approach to the temporal difference error in off-policy actor-critic algorithms," in *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 875–883, 2022.
- B. Saglam and S. S. Kozat, "Deep intrinsically motivated exploration in continuous control," 2022.

This chapter presents a novel exploration algorithm for actor-critic methods in continuous control. The algorithm is inspired by the *self-determination* theory [31] from animal psychology. We first explain the theory and show how it can be linked to RL. Then, we derive the algorithm and analyze its convergence rate. Finally, we survey the related work and compare the algorithm with the existing exploration techniques in the literature.

4.1 An Optimistic Approach to the Temporal-Difference Error

The first section provides the technical background required for developing and explaining the proposed exploration framework. The second section introduces the self-determination theory and shows how TD error can be an effective indicator for intrinsically motivating the agents.

4.1.1 Exploration in Actor-Critic Methods

Exploration in RL can be examined under *directed* and *undirected* methods [9]. Undirected exploration strategies in high dimensional action spaces aim to add noise to the chosen actions or the parameters of the policies such as additive Gaussian noise [32] or deep parameter space noise [33, 34, 11]. These deep methods have the benefits of being coherent, structured, and generalizable due to the state-dependent exploration in the parameter space, yet they also incur high computational cost and heavily depend on randomness [9]. While undirected methods aim to introduce randomness into the agents' action choices, directed methods rely on extracting useful information from the gathered experiences in a rule-based manner [10, 35, 36]. These methods need lower-level exploration to gather experiences and are proven to be effective. However, they are usually studied for hard exploration tasks where rewards are delayed, deceptive, or sparse [11].

Intrinsic motivation is an effective alternative to the directed exploration methods in hard exploration tasks, and it can be applied from three perspectives: prediction error, state novelty, and information gain [37]. The first perspective drives the agents towards the state spaces where the prediction is challenging. The second perspective is to provide an intrinsic bonus when the agent enters a state that it rarely visits, and the latter is an intrinsic reward based on the decrease of uncertainty on the environment dynamics. However, intrinsically motivated exploration methods have several drawbacks [37]. These shortcomings include the lack of generalizability to different domains as they only target low- or high-dimensional environments and are mostly designed for discrete action spaces. Moreover, the use of representation learning may hinder the long-term control from being incorporated into the exploration. Therefore, undirected methods remain the only feasible option in continuous control [11]. However, their performance can degrade in the off-policy setting due to the misalignment of the exploration and the control [38, 10]. When the exploration strategy does not align the policy update, undirected methods may fail to provide useful exploratory guidance to the agents.

4.1.2 The Self-Determination Theory and Temporal-Difference Error

Motivation is a term that refers to the processes that influence arousal, intensity, and action orientation. To be motivated is to feel driven to do an action. Psychologists classify motivation into two types, *extrinsic motivation*, taking action because of external rewards, and *intrinsic motivation*, doing something for its own sake as it is inherently enjoyable or stimulating [39].

There is a conformity between the animal motivation and RL systems in that both aim to maximize external rewards [40]. The striking resemblance between the key components of RL and the activity of dopamine neurons [41] play a crucial role in animal motivational systems [40]. The essence of computational models of the dopamine function lies in identifying the challenges that neural reward systems must face, including the vital task of choosing important actions necessary for survival [42]. The TD learning algorithm, used for planning actions to obtain future rewards, underlies the prediction error account of dopaminergic neuron responses [42]. The hypothesis that the rate of dopamine neuron spiking encodes a TD prediction error signal accounts for a large amount of neuronal recording data [41, 43, 44]. Moreover, this model also explains dopamine concentrations in the brain during electrical stimulation [45, 46] and accurately describes activity changes in animal brain reward structures measured using functional magnetic resonance imaging [47, 48, 49, 50].

Specifically, dopamine has two main functions in TD learning. First, it acts as a learning signal for the state- or action-value to enable the expectation of future rewards. Second, dopamine release guides action selection toward situations that are expected to result in a reward [42]. Dopamine either indirectly affects 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 assigned to a state or action, thereby increasing the probability of choosing it in the future. Dopamine release may also have more immediate effects on action selection. The prediction error signal indicates whether to stick with a particular course of action or switch to another based on whether the reward received is better or worse than expected [43, 51].

Therefore, we suggest that dopamine neurons can be regarded as an RL agent's state- or action-value function. Moreover, as dopamine neuron spiking represents the TD prediction error, an RL agent in continuous action spaces can be motivated by constantly maximizing the prediction error of its value function. Continually forcing the agent to select actions that lead to large value error can reveal whether the corresponding reward will be better or worse than expected, similar to dopamine release in the animal brain. Thus, a more systematic promotion of exploratory behaviors can be achieved. We present the TD error based intrinsic motivation for on- and off-policy actor-critic algorithms in Remark 1 and 2, respectively.

Remark 4 (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* $V^{\pi}(s)$.

Remark 5 (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* $Q^{\pi}(s, a)$.

4.2 Deep Directed Exploration Motivated by the Temporal-Difference Error

Here, we derive our algorithm which we examine under the on- and off-policy setting. Then, we provide insights on applying the proposed method to general actor-critic methods, and perform a convergence analysis as a comparison with the prior undirected baselines.

4.2.1 On-Policy Intrinsically Motivated Exploration

To build our directed and intrinsically motivated exploration framework, we first consider a separate, deep deterministic *exploration policy*, $\xi : \mathcal{S} \rightarrow \mathcal{E}$ that maps states to distinct exploratory directions. We represent the exploration policy by a deep neural network: ξ_φ with parameters φ. Furthermore, we assume that the state-value function estimates the value of a state by also taking into account the exploration directions that are in the subspace of the state space $\mathcal{E} \in \mathcal{S}$. Similarly, the policy also considers the exploration directions while choosing actions. Thus, the exploration policy can affect the distribution of the states received by the state-value function and policy under the parameters φ. Let $V^*(s, \vartheta)$ be the optimal state value for a given state s and exploration direction ϑ under the policy π, defined by the Bellman equation:

$$V^*(s, \vartheta) = r + \gamma V^*(s', \vartheta')|_{\vartheta' = \alpha \cdot \xi_\varphi(s')}, \tag{4.1}$$

where ϑ' is the exploration direction in the next step and α is a regularization term for the exploration policy not to adversely affect the state distribution. By Remark 4, intrinsically motivated exploration can be viewed 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 given by:

$$\max_\varphi \min_\psi \sum_{s \sim P_\pi} |V^*(s, \vartheta) - V_\psi(s, \vartheta)|^2_{\vartheta = \alpha \cdot \xi_\varphi(s)}. \tag{4.2}$$

However, the access to the optimal state-value function beforehand is impossible in practical applications. A common approach in on-policy learning is to treat empirical reversed sum of rewards $\hat{R}$, i.e., rewards-to-go, as the fixed objective for the state-value function, defined by:

$$\hat{R}_t = \sum_{i=t}^{T} R(s_i, a_i, s_{i+1}) \tag{4.3}$$

where T is the length of the horizon for which rollout transitions are gathered to train the agent. Using (4.3) and taking into account the exploration directions, the objective for the state-value network is given by:

$$J(\psi) = |\hat{R} - V_\psi(s, \vartheta)|^2_{\vartheta=\alpha\cdot\xi_\varphi(s)}. \tag{4.4}$$

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

$$\begin{aligned} J(\varphi) &= -J(\psi), \\ &= -|\hat{R} - V_\phi(s, \alpha \cdot \xi_\varphi(s))|^2, \\ &:= -\Phi(s, \alpha \cdot \xi_\varphi(s)). \end{aligned} \tag{4.5}$$

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 drives agents to state spaces where state value prediction is challenging and hence, enables the correction of the values of unknown or less visited states. The exploration network is then updated through the deterministic policy gradient (DPG) theorem [52] using the constructed objective (4.5):

$$\nabla_\varphi J(\varphi) = \mathbb{E}_{s\sim P_\pi}[\nabla_\vartheta \Phi(s, \vartheta)|_{\vartheta=\alpha\cdot\xi_\varphi(s)} \nabla_\varphi \xi_\varphi(s)]. \tag{4.6}$$

Note that the existence of the explorer network's gradient follows DPG. 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** **D**eep **I**ntrin**SC**ally M**O**ti**V**ated **E**xplo**R**ation (On-Policy DISCOVER), and provide a pseudocode in Algorithm 2.

Algorithm 2 On-Policy Deep Directed Intrinsically Motivated Exploration (On-Policy DISCOVER)

```
Input: Minibatch size N and target step-size ζ
Initialize the explorer network ξ_φ with parameters φ
for each rollout time step do
    Observe state s
    Obtain the exploration direction: ϑ = α · ξ_φ(s)
    Select an action considering the exploration direction: a ~ π_ϕ(·|s, ϑ)
    Receive reward r and observe next state s'
    Store the transition tuple (s, a, ϑ, r, s') into the rollout buffer
end for
for each policy update step do
    Update φ by the deterministic policy gradient and minibatch learning on
        the rollouts: ∇_φ J(φ) = (1/N) Σ_i^N ∇_ϑ Φ(s_i, ϑ)|_{ϑ=α·ξ_φ(s_i)} ∇_φ ξ_φ(s_i)
    Update the policy and state-value network by considering the sampled ex-
        ploration directions ϑ
end for
```

4.2.2 Off-Policy Intrinsically Motivated Exploration

For off-policy learning, now assume that the exploration directions are in a subspace of the action space $\xi : \mathcal{S} \to \mathcal{E} \in \mathcal{A}$. We also initialize a secondary frozen target explorer network with parameters φ', to obtain a fixed objective and stability in the explorer network updates, similar to the Deep Q-learning algorithm [19]. The objective of the exploration policy is to maximize the TD error of the Q-networks in accordance with the observed states and actions chosen by the policy. First, let the actions selected by the policy be perturbed by the explorer networks:

$$\tilde{a} = a + \alpha \cdot \xi_{\varphi}(s), \tag{4.7}$$

$$\tilde{a}' = a' + \alpha \cdot \xi_{\varphi'}(s'), \tag{4.8}$$

where $a' \sim \pi_{\phi'}(\cdot|s')$ and α now regularizes the additive exploration direction for the actions. Note that the action in the next state may also be selected by the behavioral policy such as in the Soft Actor-Critic algorithm [28]. In addition, to achieve exploration in the next state, target policy smoothing regularization is obtained through the target explorer network in (4.8), which is proven to improve the performance of off-policy methods [22].

As discussed in Remark 5, TD error should be kept maximum by taking into account the chosen 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 off-policy setting:

$$\begin{aligned} y &= r + \gamma Q_{\theta'}(s', \tilde{a}'), \\ J(\varphi) &= -|y - Q_\theta(s, \tilde{a})|^2, \\ &:= -\tilde{\delta}(s, a + \alpha \cdot \xi_\varphi(s)). \end{aligned} \tag{4.9}$$

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

$$\nabla_\varphi J(\varphi) = \mathbb{E}_{s \sim P_\pi, a \sim \pi}[\nabla_\vartheta \tilde{\delta}(s, a + \vartheta)|_{\vartheta = \alpha \cdot \xi_\varphi(s)} \nabla_\varphi \xi_\varphi(s)]. \tag{4.10}$$

To apply the target policy smoothing regularization in the Q-network updates and preserve 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 (4.8). This enables the directed exploration to be sustained in the critic updates:

$$J(\theta) = |y - Q_\theta(s, \tilde{a})|^2. \tag{4.11}$$

This forms the off-policy variant of DISCOVER, which is summarized in Algorithm 3. In the following section, we show how DISCOVER can be integrated with actor-critic in continuous control. Furthermore, we conduct a thorough theoretical analysis of the complexity of the proposed directed exploration approach and compare it with undirected exploration techniques.

4.2.3 Actor-Critic with Deep Directed Intrinsically Motivated Exploration

The exploration strategy we introduce is based on the TD learning prediction error, yet it also encompasses the intrinsic motivations for directed exploration as explained in Remark 6, 7 and 8, which correspond to the incentives of prediction error, state novelty and information gain, respectively.

Algorithm 3 Off-Policy Deep Directed Intrinsically Motivated Exploration (Off-Policy DISCOVER)

Input: Minibatch size N and target step-size ζ
Initialize the explorer network ξ_φ with parameters φ
Initialize the target explorer network $\varphi' \leftarrow \varphi$
for each exploration time step **do**
    Choose an action: $a \sim \pi_\phi(\cdot|s)$
    Perturb the selected action: $\tilde{a} = a + \alpha \cdot \xi_\varphi(s)$
    Observe reward r and next state s'
    Store transition tuple $(s, a, \tilde{a}, r, s')$ into the replay buffer $\mathcal{R}$
end for
for each Q-network update step **do**
    Perturb the next action through the target explorer network to modify the fixed objective for Q-learning: $\tilde{a}' = a' + \alpha \cdot \xi_{\varphi'}(s')$
end for
for each policy update step **do**
    Update φ by the deterministic policy gradient through the sampled batch of N transitions: $\nabla_\varphi J(\varphi) = \frac{1}{N}\sum_{i=1}^{N} \nabla_\vartheta \tilde{\delta}(s_i, a_i + \vartheta)|_{\vartheta = \alpha \cdot \xi_\varphi(s_i)} \nabla_\varphi \xi_\varphi(s_i)$
end for
for each target policy update period **do**
    Update the target explorer network: $\varphi' \leftarrow \zeta\varphi + (1 - \zeta)\varphi'$
end for

Remark 6 (Prediction error)**.** *The goal of maximizing the TD error makes the approximate state- and action-value functions focus on the transitions with high TD error where the prediction of states or state-action pairs is challenging. This is achieved in on-policy learning by altering the distribution over the observed states and actions that are selected by the policy for off-policy learning.*

Remark 7 (State novelty)**.** *The intrinsic reward for a state space in state novelty decreases when the agent visits it repeatedly. Therefore, undirected methods may cause the agent to revisit the same state space due to randomness [9]. However, the agent is always driven to visit the tuples with the highest TD error in the deterministic TD error maximization objective. Hence, the agent always explores novel state spaces without any chance of repetition in the action selection.*

Remark 8 (Information gain)**.** *The uncertainty on the environment dynamics is indicated by high TD error, as the value function does not have a good knowledge of the expected sum of rewards in the corresponding state space and the underlying*

MDP [26]. The explorer network tries to reduce this uncertainty by learning the value of uncertain states or state-action pairs through the exploration direction that is added to the original direction.

The proposed algorithm is a max-min optimization problem, where the value functions become more robust in states with high uncertainty and a more reliable value prediction is obtained. Interestingly, our contributions offer a potential performance improvement that is very similar to the way Generative Adversarial Networks (GAN) [53] operate. Our framework adversarially perturbs the actions chosen by the greedy policy to challenge the critic, analogous to how the generator in a GAN tries to fool the discriminator. On the other hand, the critic learns the values of the perturbed states or actions to reduce uncertainty about them, similar to the discriminator learning to distinguish fake images. Therefore, the presented max-min optimization has the same motivation as GANs, and the effectiveness and convergence of this approach have been previously demonstrated by [53].

In the off-policy counterpart, the exploration driven by intrinsic motivation can also take into account the agent's learning history by training from the batch of transitions drawn from the experience replay buffer. However, as mentioned, the separate updates of exploration strategy and policies have problems with off-policy learning [54]. Therefore, the exploration policy should be updated and evaluated whenever the policy of the underlying actor-critic algorithm is optimized. This ensures that the exploration framework is aligned with the agent's policy update, with the same update rule and update frequency of the target networks, i.e., soft or hard update. Furthermore, the exploration policy should have the same structure as the agent's policy when combined with both on- and off-policy algorithms. The reason for this imitation is that the exploration policy should be equally capable as the agent's policy. If the agent's policy can learn to control the environment with a policy gradient algorithm, the exploration policy can also learn to maximize the TD error with DPG under the same update structure and hyperparameters, which we show through empirical studies in the following section. This requirement is emphasized in Remark 9.

Remark 9. *The exploration policy should have the same architecture and hyperparameter setting as the policy of the underlying algorithm when used with actor-critic algorithms, e.g., learning rate, actor network size and depth, update frequency, and target network usage.*

The high computational complexity is a major drawback in the undirected methods [9]. We analyze the time complexity of the proposed exploration strategy in Theorem 2 under deterministic ergodic MDPs. Moreover, Lemma 2 comments on the time complexity of any exploration strategy under non-deterministic ergodic MDP setting. We then compare the time complexity of our approach with undirected strategies in Corollary 2, from which we infer that the time to find an optimal policy under the exploration driven by maximum TD error is always less than or equal to the undirected exploration. In the next section, we present an extensive set of simulations that show that DISCOVER achieves remarkable results in on- and off-policy settings, surpassing the state-of-the-art exploration techniques. Additionally, we show that only a single hyperparameter choice is required for a wide range of tasks when the explorer network has the same structure and synchronization as the agent's actor network, validating Remark 9.

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 [55]. □

Lemma 2 (Time complexity of the exploration under ergodic stochastic MDPs)**.** *For each $n \in \{2, 3, \dots\}$, 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 [9]. □

Theorem 2 (Time complexity of the exploration motivated by the maximum temporal-difference error under ergodic deterministic MDPs)**.** *The upper bound* *for the number of actions needed to discover an optimal policy in any finite ergodic deterministic Markov decision process with a goal state is $\mathcal{O}((d^2m+m^2)n^2)$, where m is the number of unknown states or state-action pairs, and $\mathcal{O}(d)$ is the time complexity for each iteration in the temporal-difference learning. This bound is achieved by exploring according to the maximum temporal-difference error.*

Proof. The proof is based on the generalization of Theorem 3 in [9] to the exploration driven by the maximum TD error. The update rule for TD learning at iteration t under the current policy π is given for on- and off-policy learning, respectively, as:

$$V^{\pi}_{t+1}(s) = r + \gamma V^{\pi}_{t}(s'), \tag{4.12}$$

$$Q^{\pi}_{t+1}(s,a) = r + \gamma \max_{a'} Q^{\pi}_{t}(s',a'). \tag{4.13}$$

First, the equations show that $V^{\pi}(\cdot)$ and $Q^{\pi}(\cdot)$ are monotonically increasing with the learning time. Next, suppose an optimal path $\langle s_1 = s_{init}, s_1, s_2, \ldots, s_k = s_{goal}\rangle$ from some initial state s_{init} to a goal state s_{goal}. If $V^{\pi}(s_i)$ is accurate 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 accurate state-value to s_{i-1} [9]. This also holds for Q-learning denoted by (4.13), namely if $Q^{\pi}(s_i, a_i)$ is accurate 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 accurate action-value to (s_{i-1}, a_{i-1}) [9].

Learning the environment will determine the accurate value function and thus, an optimal policy if the goal state exists. The exploration strategy in interest will guide 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 $\mathcal{O}(d^2mn^2)$ actions [9]. Moreover, by Theorem 1 shown by [9], the complexity of each exploration run is bounded above by $\mathcal{O}(mn^2)$. Hence, the resulting worst-case complexity for an exploration driven by the maximum TD error under ergodic deterministic MDPs is then in $\mathcal{O}((d^2m + m^2)n^2)$. $\square$

Corollary 2. *The expected time to discover an optimal policy using exploration driven by the maximum temporal-difference error is upper bounded by the expected time using undirected exploration in any finite ergodic Markov decision process.*

Proof. The proof follows from Lemma 1, 2, and Theorem 2. The polynomial or quadratic time is upper bounded by the exponential time. Therefore, by Lemma 1 and Theorem 2, the TD error based exploration is upper bounded by undirected exploration for ergodic deterministic MDPs. From Lemma 2, we know that both maximum TD error based directed exploration and undirected exploration are lower bounded by the time complexity exponential in the state space dimension for ergodic stochastic MDPs. Therefore, exploration driven by maximum TD error has a time complexity that is either equal or less than undirected exploration's time complexity for any MDP, in the worst-case scenario. $\square$

4.3 Experiments

We conduct experiments to show the effectiveness of DISCOVER and contrast it with the existing exploration strategies. Following the structure of Chapter 3, we first review the existing studies in the literature that we compare our method with. Then, we present the experimental details and report the results. Finally, we discuss the results obtained.

4.3.1 Competing Methods

The literature on exploration methods mostly focuses on straightforward cases where the function approximators are often linear and the state and action spaces are low-dimensional [11]. These methods can explore the parameter space effectively, yet they are not adequate for more complex environments. Hence, we examine the undirected deep exploration approaches: NoisyNet [33], Parameter Space Noise for Exploration (PSNE) [34], Deep Coherent Exploration (Coherent) [11], standard Gaussian exploration [32], and greedy action selection. We give a brief overview of each method and point out their limitations for our discussion of the empirical results.

The first two techniques use a distribution over policies to explore the parameter space in a trajectory-based manner. A possible way to obtain exploring policies is by adding noise to the weights of all layers of a deep neural network. However, this sampling process creates uncertainty, which can be handled using Monte Carlo integration. Nevertheless, these methods have several drawbacks that affect their performance. First, trajectory-based strategies can be inefficient as they only test one exploration strategy per trajectory, which may be trapped in the local optima. Second, Monte-Carlo integration introduces a high variance in the gradient estimates.

While NoisyNet and PSNE use a parameter-space distribution over policies to explore in a trajectory-based fashion, Coherent extends the step-based and trajectory-based exploration to add noise to the last layer of the policy networks. The framework also considers 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 all evaluate the exploration strategy only when the exploration is done. This makes them fail in the off-policy setting since the exploration is separated from learning [38, 10], as our empirical results show. Lastly, the standard Gaussian exploration simply samples an exploration noise from a zero-mean Gaussian random variables the standard deviation of which remains a hyperparameter.

4.3.2 Experimental Details

We use the experimental setup described in Chapter 3.3.2 unless stated otherwise. We test each method with on-policy actor-critic algorithms, Advantage Actor-Critic (A2C) [56], Proximal Policy Optimization (PPO) [57], and also the off-policy actor-critic algorithms, Soft Actor-Critic (SAC) [28] and Twin Delayed Deep Deterministic Policy Gradient (TD3) [22]. We again use challenging MuJoCo [12] and Box2D [13] continuous control tasks interfaced by OpenAI Gym [14]. Algorithm-specific adaptations of DISCOVER are available in our code[1].

Our implementation of NoisyNet is based on the code from authors' GitHub repository[2]. We refer to the authors' implementation in OpenAI Baselines[3] [58] for PSNE, and the original paper for Deep Coherent Exploration. In addition, we implement the on-policy algorithms A2C and PPO through the well-known GitHub repository[4] with tuned hyperparameters for the continuous control tasks in OpenAI Gym.

For the action space noise, we use a fixed zero-mean Gaussian distribution with a standard deviation of 0.1. Competing algorithms follow the hyperparameter settings described in the original papers closely. For the parameter-space noise algorithms, we set the parameter noise at 0.017 for on-policy and 0.034 for off-policy algorithms as they were found to produce the best performance. Additionally, 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 mentioned in Remark 9, we start our exploration framework to exactly match the policy network's structure and hyperparameter setting for all baselines. This initialization includes the network size and depth, optimizer, learning rate, update frequency, and weight decay if it exists. Furthermore, we use target explorer networks in off-policy algorithms for which the target network update

[1] `https://github.com/baturaysaglam/DISCOVER`
[2] `https://github.com/Kaixhin/NoisyNet-A3C`
[3] `https://github.com/openai/baselines`
[4] `https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail`

rule of the underlying actor-critic methods, i.e., soft or hard update, and learning rate also apply to the target explorer network. This also makes the delayed policy updates in the TD3 algorithm apply to the explorer network. We consider five values for the exploration regularization term $\alpha = \{0.1, 0.3, 0.6, 0.9, 1.0\}$, where we use $\alpha = 0.1$ for On-Policy DISCOVER and $\alpha = 0.3$ for Off-Policy DISCOVER.

4.4 Results

4.4.1 Comparative Evaluation

4.4.1.1 On-Policy Learning

The learning curves under the on-policy algorithms A2C and PPO are depicted in Figures 4.1 and 4.2, respectively. Note that the default exploration used in these algorithms is the inherent entropy maximization of the actions generated by the stochastic policy. As the actions are drawn from the Gaussian policy, a maximized entropy produces diverse actions and thus, more effective exploration. From a general perspective, we observe that On-Policy DISCOVER consistently enhances the baseline algorithms and either matches or surpasses the competing approaches in all tasks in terms of the convergence rate and highest evaluation returns. Mainly, we observe a significant performance improvement in more challenging tasks and high dimensional environments such as Ant, Humanoid, and Swimmer. Therefore, we deduce that when the underlying algorithm fails at learning, the effectiveness of the used exploration strategy becomes prominent.

For the competing approaches, although our implementation of NoisyNet and PSNE follow the authors' implementation and tuned hyperparameters, we notice that they usually perform worse than the baseline, while Deep Coherent Exploration shows a considerable performance. It was previously demonstrated by [11] that NoisyNet and PSNE fail in the continuous setting as they were originally designed for the standard Atari games [59] in which action spaces are discrete.

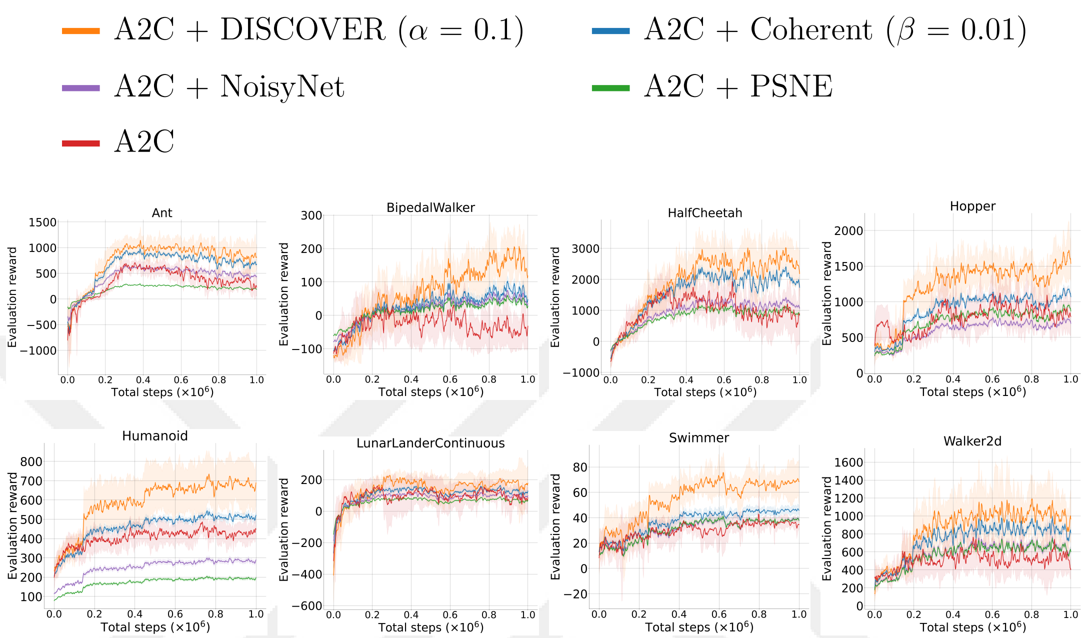

Figure 4.1: The learning curves for the set of MuJoCo and Box2D continuous control tasks under the A2C algorithm. The shaded region represents a 95% confidence interval over 10 trials. A sliding window of size 5 smooths the curves for visual clarity.

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 surpasses it by a significant margin. Lastly, On-Policy DISCOVER exhibits an improved but suboptimal behavior in environments when the baseline does not converge such as for PPO in HalfCheetah and Humanoid. Naturally, when the off-policy method cannot solve the environment sufficiently, the performance improvement offered by DISCOVER does not reach optimal policies.

Finally, the performance improvement by our algorithm is verified by performing the 2-sample statistical test. The setup highlighted in Chapter 3.4.1 is directly followed, and a detailed description is also provided in Appendix A.3. The results are presented in Table 4.1. It is observed that the performance improvement by DISCOVER is statistically significant over the competing approaches in the majority of the tasks tested. Specifically, PPO benefits more from the exploratory

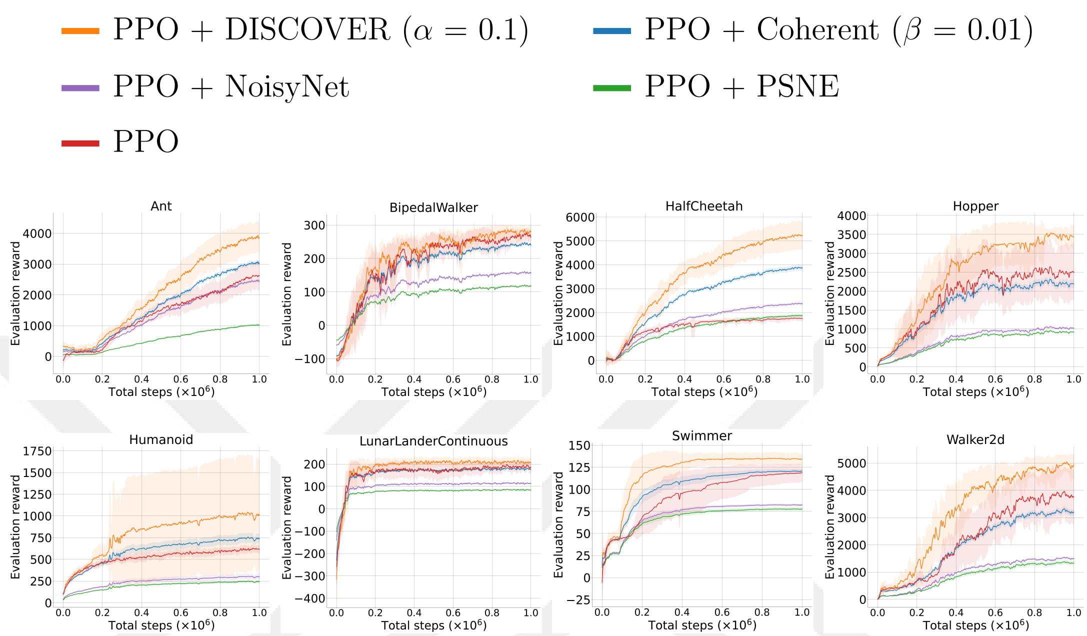

Figure 4.2: The learning curves for the set of MuJoCo and Box2D continuous control tasks under the PPO algorithm. The shaded region represents a 95% confidence interval over 10 trials. A sliding window of size 5 smooths the curves for visual clarity.

guidance of DISCOVER. We attribute this finding to the suboptimal performance of A2C in most of the environments. As shown on the learning curves, the gains of DISCOVER will be more prominent when the underlying actor-critic method performs well. Nonetheless, On-Policy DISCOVER is well-supported by the statistical evidence.

4.4.1.2 Off-Policy Learning

The evaluation results for the off-policy setting are shown in Figures 4.3 and 4.4 for the SAC and TD3 algorithms, respectively. The learning curves demonstrate that Off-Policy DISCOVER considerably enhances the baselines in most of the tasks. Specifically, in high-dimensional environments, Ant and Humanoid, which are considered challenging [30], DISCOVER significantly speeds up the learning while noise-based methods have a little or no improvement. Moreover, TD3

	Environment	p_{Coherent}	p_{NoisyNet}	p_{PSNE}	p_{entropy}
A2C	Ant	0.106	0.003	0.000	0.000
	BipedalWalker	0.005	0.002	0.001	0.000
	HalfCheetah	0.050	0.000	0.000	0.005
	Hopper	0.001	0.000	0.000	0.000
	Humanoid	0.004	0.000	0.000	0.000
	LunarLanderContinuous	0.141	0.058	0.022	0.048
	Swimmer	0.003	0.001	0.001	0.000
	Walker2d	0.079	0.002	0.002	0.000
PPO	Ant	0.001	0.000	0.000	0.000
	BipedalWalker	0.000	0.000	0.000	0.044
	HalfCheetah	0.000	0.000	0.000	0.000
	Hopper	0.000	0.000	0.000	0.010
	Humanoid	0.176	0.015	0.011	0.095
	LunarLanderContinuous	0.000	0.000	0.000	0.008
	Swimmer	0.000	0.000	0.000	0.005
	Walker2d	0.000	0.000	0.000	0.017

Table 4.1: The resulting p-values from a 2-sample t-test performed over the last 10 evaluation returns of On-Policy DISCOVER and the competing methods over 10 trials under the on-policy algorithms. Subscripts denote the competing method with which DISCOVER is compared and "entropy" denotes the standard entropy maximization objective of the stochastic policy. A p-value less than the significance level of 0.05 indicates that the difference in performance is statistically significant. Values are rounded up to three decimal points.

and SAC cannot overcome 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 noticeable only when the baseline performs well. Our method, in contrast, can break this correlation to an extent and converge to an improved policy.

We further 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 slightly enhance the baseline or closely follow the no exploration (greedy) setting in terms of the learning speed and achieved evaluation returns. Although the undirected algorithms are previously shown to have a considerable impact on the on-policy algorithms such as A2C and

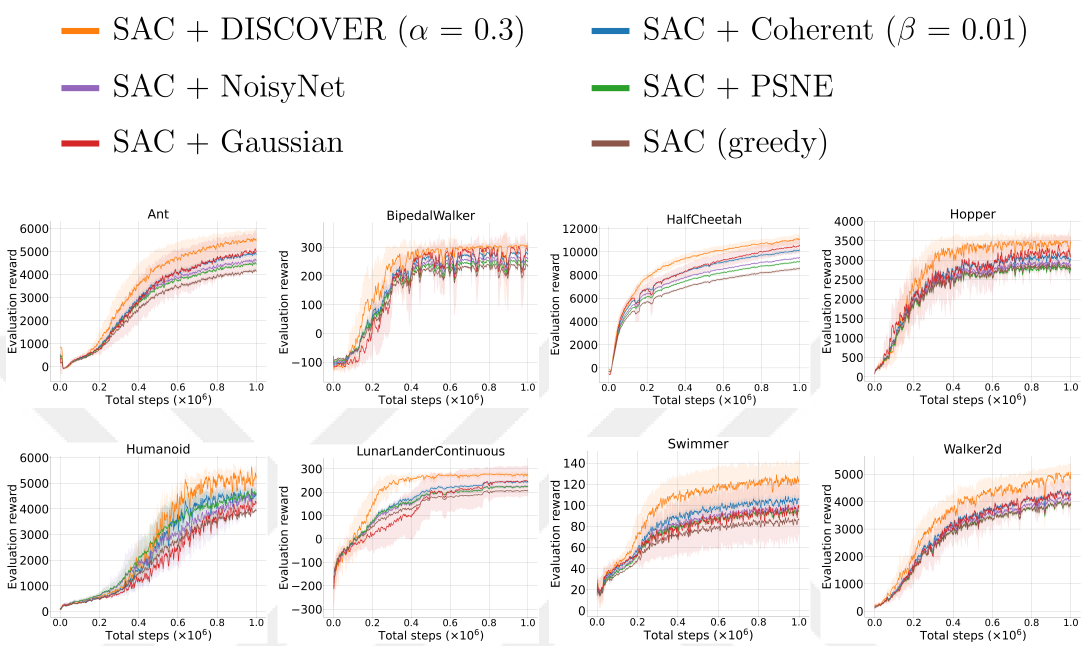

Figure 4.3: The learning curves for the set of MuJoCo and Box2D continuous control tasks under the SAC algorithm. The shaded region represents a 95% confidence interval over 10 trials. A sliding window of size 5 smooths the curves for visual clarity.

PPO, they show a poor improvement in the off-policy setting. In fact, for some of the environments, they perform worse than the Gaussian action noise. This poor performance is due to the asynchronous exploration and policy updates in the off-policy learning [54]. The undirected methods only evaluate and update their deep exploration strategy when the exploration is done. The separated updates of the agent's policy and exploration strategy cannot guide agents to useful state spaces. Conversely, our method updates its exploration policy in sync with the agent's policy update in an off-policy manner, which results in a apparent and substantial improvement in the off-policy setting, verifying Remark 9.

Although DISCOVER is shown to obtain 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 [30]. Contrarily, the performance improvement in unstable environments is

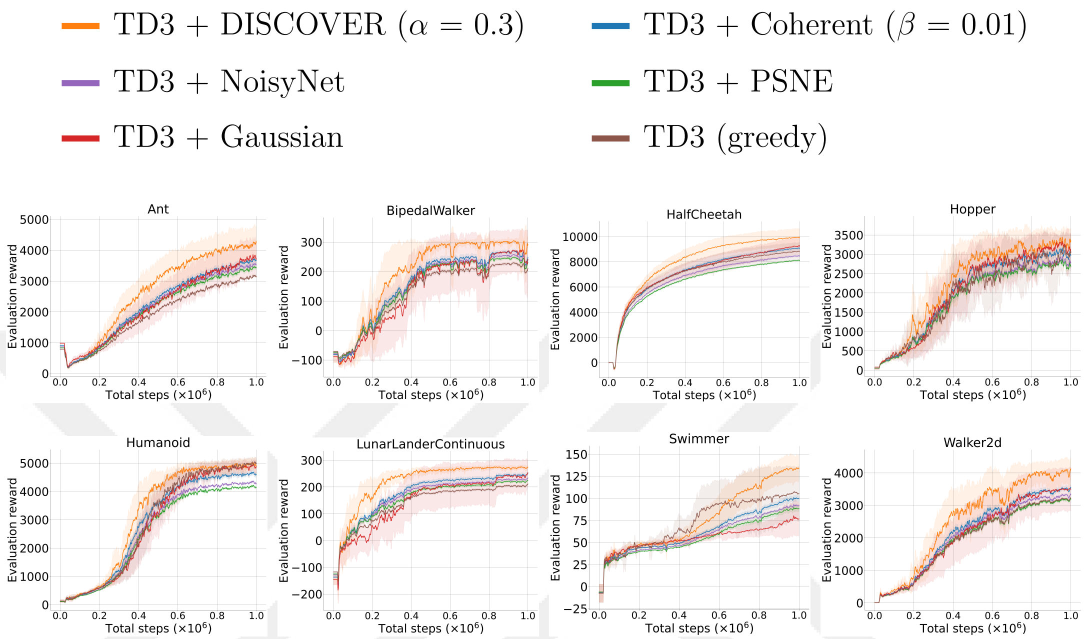

Figure 4.4: The learning curves for the set of MuJoCo and Box2D continuous control tasks under the TD3 algorithm. The shaded region represents a 95% confidence interval over 10 trials. A sliding window of size 5 smooths the curves for visual clarity.

more significant as undirected exploration strategies can cause sudden failures in off-policy learning [30]. In addition, we still observe an improved but sub-optimal behavior when the underlying off-policy algorithm does not converge to the optimal policy. As discussed, it is natural for an exploration strategy not to achieve optimal policies when the baseline is unable to sufficiently solve the environment. Nevertheless, 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 Remark 4 and 5. Thus, we conclude that the introduced directed exploration scheme can overcome the limitations induced by the random walk behavior in the undirected approaches.

Table 4.2 presents the results of the pairwise t-test comparing Off-Policy DISCOVER with other competing methods. It is evident from the analysis that DISCOVER surpasses Coherent, NoisyNet, and PSNE in terms of performance.

The subpar performance of the deep exploration baselines can be attributed to the asynchronous updates in policy and exploration strategies, as mentioned. The mean curves clearly demonstrate the superiority of DISCOVER over the additive Gaussian exploration approach. However, in some tasks, the difference between the two methods is not statistically significant. Nevertheless, DISCOVER consistently exhibits notable performance on average, and by increasing the number of seeds, the resulting p-value would decrease, rendering the difference statistically significant. Hence, based on our empirical analysis, DISCOVER demonstrates the ability to generalize to both on- and off-policy learning scenarios while achieving a state-of-the-art performance.

	Environment	p_{Coherent}	p_{NoisyNet}	p_{PSNE}	p_{Gaussian}	p_{greedy}
SAC	Ant	0.007	0.001	0.000	0.133	0.000
	BipedalWalker	0.000	0.000	0.000	0.124	0.000
	HalfCheetah	0.000	0.000	0.000	0.046	0.000
	Hopper	0.000	0.000	0.000	0.094	0.000
	Humanoid	0.000	0.003	0.000	0.000	0.000
	LunarLanderContinuous	0.000	0.000	0.000	0.193	0.000
	Swimmer	0.005	0.001	0.000	0.015	0.000
	Walker2d	0.001	0.000	0.000	0.008	0.000
TD3	Ant	0.015	0.006	0.002	0.079	0.000
	BipedalWalker	0.003	0.001	0.001	0.141	0.000
	HalfCheetah	0.010	0.000	0.000	0.059	0.010
	Hopper	0.002	0.000	0.000	0.101	0.038
	Humanoid	0.000	0.000	0.000	0.200	0.743
	LunarLanderContinuous	0.000	0.000	0.000	0.163	0.000
	Swimmer	0.000	0.000	0.000	0.000	0.040
	Walker2d	0.001	0.000	0.000	0.049	0.000

Table 4.2: The resulting p-values from a 2-sample t-test performed over the last 10 evaluation returns of Off-Policy DISCOVER and the competing methods over 10 trials under the off-policy algorithms. Subscripts denote the competing method with which DISCOVER is compared. A p-value less than the significance level of 0.05 indicates that the difference in performance is statistically significant. Values are rounded up to three decimal points.

4.4.2 Ablation Studies

To examine the impact of the components: exploration direction regularization λ, delayed exploration policy updates, target explorer network, and target smoothing regularization in the off-policy TD learning, we conduct ablation studies. We use TD3 as the base algorithm for our ablations since it incorporates delayed policy updates, target policy networks, and target smoothing regularization. We then evaluate the performance of Off-Policy DISCOVER when each of these components is removed or modified. In addition, we report the results for On-Policy DISCOVER with PPO for different values of λ. The other components of DISCOVER are not applicable in the on-policy setting because there is no delay in policy updates and no target network in on-policy algorithms.

We select four environments with different features for our ablation studies to account for the possible variation of each component's importance across tasks. We use the low-dimensional Hopper, high-dimensional Humanoid, and Swimmer and HalfCheetah that require a large amount of on- and off-policy samples to be solved, respectively [30]. Table 4.3 shows the ablation results. We set $\lambda = 0.3$ for Off-Policy DISCOVER unless specified otherwise.

The parameter λ controls the exploration level, with $\lambda = 1$ corresponding to full exploration and $\lambda = 0$ corresponding to greedy action selection. As shown in Table 4.3, when λ is greater than around 0.3, the actions become too noisy and hinder the learning of the environment for Off-Policy DISCOVER. Conversely, when λ is too low, the exploration is insufficient and results in suboptimal policies. A similar trend is observed for the on-policy setting, where the state distribution is distorted when λ is higher than around 0.1, which reduces the learning efficiency. Furthermore, when $\lambda = 0.0$, the agent only chooses greedy actions and is stuck at a suboptimal policy.

We also evaluate the impact of the target explorer network, the target smoothing regularization that perturbs the next action in the TD learning, and the delayed exploration policy updates in the off-policy setting. We apply the target policy regularization with the behavioral explorer network when the target network is not used in the full algorithm. We find that the performance is similar to the case without target policy smoothing. These minor changes have a slight benefit, but we still see a further improvement in Swimmer. Our ablation studies suggest that our exploration framework can achieve optimal performance when it follows the policy framework of the baseline, due to the coupled explorer and policy networks and in accordance with Remark 9. Therefore, when DISCOVER is used with an actor-critic algorithm, it should match the exact setting of the policy framework. This leads to a single hyperparameter to optimize, which is the exploration regularization term λ.

	Setting	HalfCheetah	Hopper	Humanoid	Swimmer
On-Policy	$\lambda = 0.0$	3179.35 ± 1272.24	3126.34 ± 314.89	700.42 ± 84.10	127.85 ± 4.29
	$\lambda = 0.1$	**5224.88 ± 559.66**	**3436.32 ± 297.48**	**1010.48 ± 628.32**	**134.26 ± 6.03**
	$\lambda = 0.3$	2500.32 ± 927.24	1890.07 ± 482.35	634.86 ± 43.07	122.57 ± 13.87
	$\lambda = 0.6$	2621.43 ± 1284.58	2140.21 ± 787.06	626.17 ± 51.14	90.26 ± 33.05
	$\lambda = 0.9$	1822.65 ± 1014.22	1607.60 ± 1007.96	528.45 ± 29.92	56.78 ± 18.70
	$\lambda = 1.0$	1932.92 ± 793.51	1581.94 ± 879.27	555.24 ± 28.30	74.23 ± 13.40
Off-Policy	$\lambda = 0.0$	8843.51 ± 692.55	3041.59 ± 299.47	4993.90 ± 87.88	105.19 ± 31.07
	$\lambda = 0.1$	8762.55 ± 1135.57	2704.40 ± 668.97	**5080.31 ± 102.71**	107.76 ± 27.88
	$\lambda = 0.3$	**9939.39 ± 687.32**	**3350.27 ± 212.95**	4941.76 ± 152.62	**133.94 ± 14.50**
	$\lambda = 0.6$	7971.19 ± 516.31	2651.23 ± 438.42	4719.90 ± 190.64	88.81 ± 17.31
	$\lambda = 0.9$	4594.68 ± 605.53	795.43 ± 294.24	393.91 ± 96.95	43.13 ± 4.28
	$\lambda = 1.0$	2732.76 ± 501.10	203.45 ± 61.01	253.12 ± 78.61	25.92 ± 5.11
	DISCOVER (complete)	**9939.39 ± 687.32**	**3350.27 ± 212.95**	**4941.76 ± 152.62**	**133.94 ± 14.50**
	w/o DPU	8900.50 ± 927.28	2794.50 ± 529.77	4924.07 ± 128.46	105.55 ± 18.51
	w/o TN	8756.69 ± 986.64	3248.61 ± 420.11	4463.95 ± 1115.36	115.48 ± 25.50
	w/o TSR	8795.56 ± 892.98	2953.44 ± 378.20	4857.31 ± 224.26	113.02 ± 29.15

Table 4.3: The average return over the last 10 evaluations over 10 trials of 1 million time steps, comparing ablation over DISCOVER under $\lambda = \{0.0, 0.1, 0.3, 0.6, 0.9, 1.0\}$, Off-Policy DISCOVER without delayed explorer policy updates (DPU), target network (TN), and target smoothing regularization (TSR). Bold values represent the maximum for each environment under on- or off-policy setting in terms of the mean rewards. The PPO and TD3 algorithms are used as the underlying actor-critic algorithms for On-Policy and Off-Policy DISCOVER, respectively.

4.4.3 Visualization of the State Visitations

We analyze the behavior of our exploration policy by examining the set of transitions that are collected during the training of Off-Policy DISCOVER with the TD3 algorithm. We choose off-policy learning for this analysis because it enables a gradual learning process rather than the on-policy algorithms that use a single behavioral policy with fixed parameters to collect the experiences for a long rollout duration. Therefore, the transitions stored in the experience replay buffer can offer a comprehensible visualization of the state space visitations, as demonstrated by [10].

We train the TD3 algorithm with and without exploration using Off-Policy DISCOVER for 1 million time steps in the Swimmer environment. We use a replay buffer of size 1 million transitions where we sample randomly. Therefore, each state that is visited is available at the end of the training under both exploration settings. We show the state visitation probabilities in Figure 4.5. We use Kernel Density Estimation based on the samples in 2D embedding space to estimate the state visitation probabilities. The 2D embeddings are obtained by projecting the datasets jointly onto 2D embedding space using t-SNE [60]. We select the Swimmer environment because its state space dimension is relatively low compared to the other environments, which avoids potential artifacts in the low-dimensional visualization. Furthermore, in Chapter 3.4.1, we see that actor-critic methods usually perform poorly in the Swimmer environment, which can be regarded as difficult [30]. To further minimize the projection error, we first apply PCA to reduce the state dimension from 8 to 4 before t-SNE. The PCA operation yields a proportion of variance explained value of 0.984 and 0.966 for no exploration and Off-Policy DISCOVER datasets, respectively.

As observed in Figure 4.5, we have three groups of visualization for each dataset. To provide insights on the TD error maximization in the exploration, we color the states with the TD errors corresponding to their transitions. Thus, darker regions indicate large absolute TD errors. We group the state space visualizations into the states visited in the early, middle, and late stages of the training. In the early stages of the data collection, the greedy action selection covers the two sides of the state space. The agent hardly sees the middle space throughout the training, and visited states converge to a smaller region on both sides of the state space. Furthermore, the TD errors of the corresponding states are almost identical. This is a result of the greedy action selection in that the agent often chooses the same actions and observes the same states, whose TD error are continuously reduced by Q-learning.

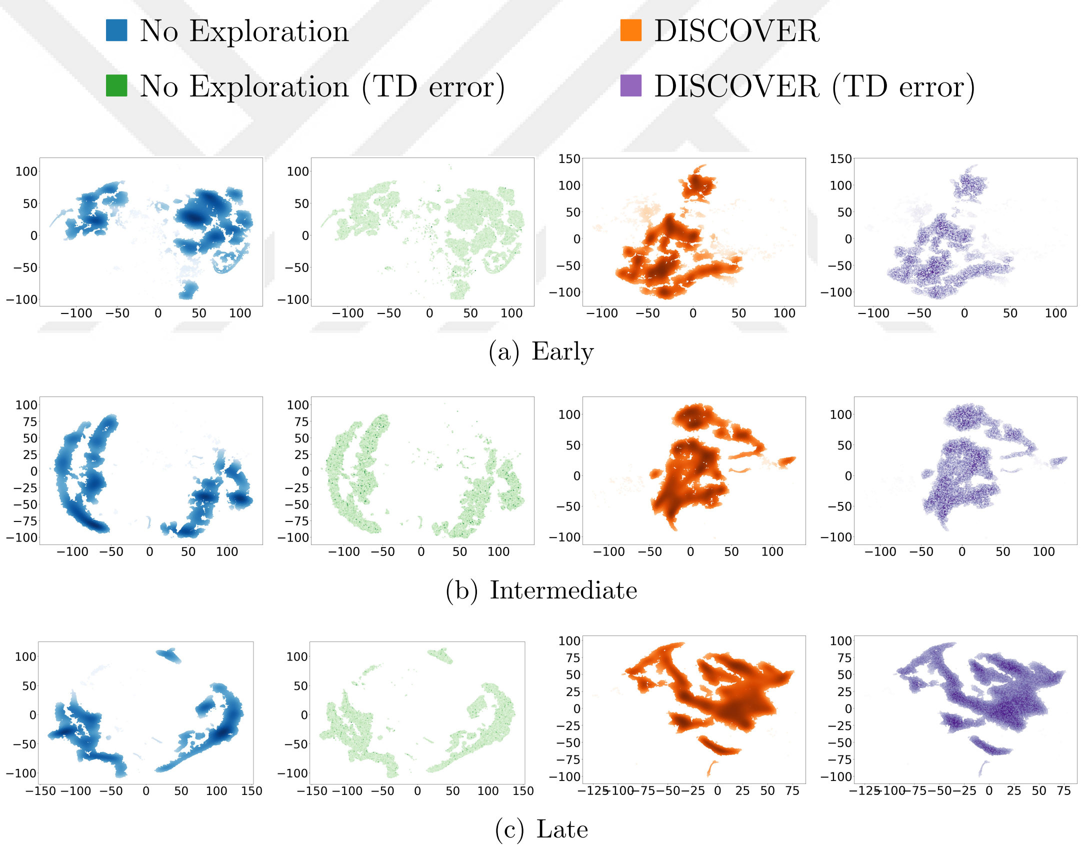

Figure 4.5: The density contours for the states visited by the TD3 algorithm under greedy action selection and DISCOVER. Larger temporal-difference errors are illustrated by darker regions in the TD error contours.

In contrast, our approach diversifies the state visitation distributions across different modes of regions, compared to the greedy action selection. From the early to the late stages of data collection, the exploration policy learns how to guide the agent to state-action pairs with high TD error, which we see from the increasing similarity of the state visitation probabilities and TD error heat map. Therefore, this comparison shows that our exploration policy is gradually learning a stationary policy that maximizes the TD error and provides a robust exploration strategy to enhance the underlying continuous control method.

Our empirical studies also report an interesting insight. In all stages of the training, DISCOVER consistently visits almost opposite distinct states in the distribution space compared to the greedy action selection. We infer that our exploration policy also learns how the agent selects the actions greedily and systematically guides it to explore diverse state regions based on this selection. This enables a consistent and continuous improvement in the agent's performance and indicates our global and directed exploration strategy, which is notably different from noise-based exploration, which remains local. Finally, we achieve a significant 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.

Chapter 5

Conclusion

This thesis presents two novel techniques to enhance the performance of continuous control deep reinforcement learning (RL) algorithms by tackling the challenges of sample efficiency and exploration-exploitation trade-off. The first technique is a novel framework for experience replay sampling in actor-critic methods that balances temporal-difference (TD) error and policy gradient. The second technique is a directed exploration strategy that leverages intrinsic motivation and maximizes the error of the value function. We prove the theoretical soundness of both techniques and evaluate their empirical effectiveness in various continuous control benchmarks.

The first technique, Loss Adjusted Approximate Actor Prioritized Experience Replay (LA3P), is a novel framework for sampling experiences in actor-critic methods that stabilizes and prevents divergence caused by the Prioritized Experience Replay (PER) algorithm. We theoretically show that in PER, actor networks cannot be effectively trained with transitions that have large TD errors, and that training actor and critic networks with different transitions throughout the learning violates the actor-critic theory. We empirically demonstrate that LA3P significantly outperforms competing methods and improves upon the state-of-the-art in standard off-policy actor-critic algorithms.

The second technique, DISCOVER, is a directed exploration strategy that leverages intrinsic motivation to generate exploratory behaviors that are both informative and diverse. We draw inspiration from established theories on animal motivational systems and adapt them to the actor-critic methods in the continuous control setting. We empirically show that DISCOVER guides the agents to diverse state regions that are orthogonal to the greedy actions and outperforms the competing exploration methods by a large margin in most of the tested tasks.

The proposed techniques open up several avenues for future research. For instance, one could investigate how to combine LA3P and DISCOVER in a single algorithm to further enhance the performance of continuous control deep RL algorithms. One could also explore how to extend LA3P and DISCOVER to other domains such as discrete action spaces, multi-agent settings, or hierarchical RL. Moreover, one could study how to incorporate other sources of intrinsic motivation such as curiosity, novelty, or empowerment into DISCOVER or other exploration strategies. Lastly, one could apply LA3P and DISCOVER to real-world applications such as robotics, autonomous driving, or smart manufacturing, where sample efficiency and exploration are essential for achieving practical and robust solutions.

Bibliography

[1] B. Saglam, D. Gurgunoglu, and S. S. Kozat, "Deep reinforcement learning based joint downlink beamforming and ris configuration in ris-aided mu-miso systems under hardware impairments and imperfect csi," 2023.

[2] L. ji Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," in *Machine Learning*, pp. 293–321, 1992.

[3] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015. cite arxiv:1511.05952Comment: Published at ICLR 2016.

[4] Y. Oh, K. Lee, J. Shin, E. Yang, and S. J. Hwang, "Learning to sample with local and global contexts in experience replay buffer," in *International Conference on Learning Representations*, 2021.

[5] Y. Oh, J. Shin, E. Yang, and S. J. Hwang, "Model-augmented prioritized experience replay," in *International Conference on Learning Representations*, 2022.

[6] B. Saglam, F. B. Mutlu, D. C. Cicek, and S. S. Kozat, "Actor prioritized experience replay," in *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.

[7] D. C. Cicek, E. Duran, B. Saglam, F. B. Mutlu, and S. S. Kozat, "Off-policy correction for deep deterministic policy gradient algorithms via batch prioritized experience replay," in *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 1255–1262, 2021.

[8] S. Fujimoto, D. Meger, and D. Precup, "An equivalence between loss functions and non-uniform sampling in experience replay," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), vol. 33, pp. 14219–14230, Curran Associates, Inc., 2020.

[9] S. Thrun, "The role of exploration in learning control," in *Handbook for Intelligent Control: Neural, Fuzzy and Adaptive Approaches* (D. White and D. Sofge, eds.), Florence, Kentucky 41022: Van Nostrand Reinhold, 1992.

[10] B. Saglam, F. B. Mutlu, and S. S. Kozat, "An optimistic approach to the temporal difference error in off-policy actor-critic algorithms," in *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 875–883, 2022.

[11] Y. Zhang and H. Van Hoof, "Deep coherent exploration for continuous control," in *Proceedings of the 38th International Conference on Machine Learning* (M. Meila and T. Zhang, eds.), vol. 139 of *Proceedings of Machine Learning Research*, pp. 12567–12577, PMLR, 18–24 Jul 2021.

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

[13] I. Parberry, *Introduction to Game Physics with Box2D.* USA: CRC Press, Inc., 1st ed., 2013.

[14] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *CoRR*, vol. abs/1606.01540, 2016.

[15] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," in *Advances in Neural Information Processing Systems* (S. Solla, T. Leen, and K. Müller, eds.), vol. 12, MIT Press, 1999.

[16] R. Sutton, "Learning to predict by the method of temporal differences," *Machine Learning*, vol. 3, pp. 9–44, 08 1988.

[17] R. E. Bellman, *Dynamic Programming.* USA: Dover Publications, Inc., 2003.

[18] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, May 1992.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb 2015.

[20] B. Saglam, D. C. Cicek, F. B. Mutlu, and S. S. Kozat, "Safe and robust experience sharing for deterministic policy gradient algorithms," 2022.

[21] B. Saglam, D. C. Cicek, F. B. Mutlu, and S. S. Kozat, "Off-policy correction for actor-critic methods without importance sampling," 2022.

[22] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, (Stockholmsmässan, Stockholm SWEDEN), pp. 1587–1596, PMLR, 10–15 Jul 2018.

[23] B. Saglam, E. Duran, D. C. Cicek, F. B. Mutlu, and S. S. Kozat, "Estimation error correction in deep reinforcement learning for deterministic actor-critic methods," in *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 137–144, 2021.

[24] D. C. Cicek, E. Duran, B. Saglam, K. Kaya, F. Mutlu, and S. S. Kozat, "Awd3: Dynamic reduction of the estimation bias," in *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 775–779, 2021.

[25] B. Saglam, F. B. Mutlu, D. C. Cicek, and S. S. Kozat, "Parameter-free reduction of the estimation bias in deep reinforcement learning for deterministic policy gradients," 2022.

[26] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[27] R. Sutton, D. Mcallester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Adv. Neural Inf. Process. Syst*, vol. 12, 02 2000.

[28] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870, PMLR, 10–15 Jul 2018.

[29] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft actor-critic algorithms and applications," 2018.

[30] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18, (New Orleans, Louisiana, USA), AAAI Press, 2018.

[31] E. L. Deci and R. M. Ryan, *Intrinsic motivation and self-determination in human behavior.* Springer Science & Business Media, 2013.

[32] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, May 1992.

[33] M. Fortunato, M. G. Azar, B. Piot, J. Menick, M. Hessel, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, C. Blundell, and S. Legg, "Noisy networks for exploration," in *International Conference on Learning Representations*, 2018.

[34] M. Plappert, R. Houthooft, P. Dhariwal, S. Sidor, R. Y. Chen, X. Chen, T. Asfour, P. Abbeel, and M. Andrychowicz, "Parameter space noise for exploration," in *International Conference on Learning Representations*, 2018.

[35] B. Saglam, F. B. Mutlu, O. Dalmaz, and S. S. Kozat, “Unified intrinsically motivated exploration for off-policy learning in continuous action spaces,” in *2022 30th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, 2022.

[36] B. Saglam, F. B. Mutlu, K. Gonc, O. Dalmaz, and S. S. Kozat, “An intrinsic motivation based artificial goal generation in on-policy continuous control,” in *2022 30th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, 2022.

[37] A. G. Barto, *Intrinsic Motivation and Reinforcement Learning*, pp. 17–47. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[38] B. Saglam and S. S. Kozat, “Deep intrinsically motivated exploration in continuous control,” 2022.

[39] R. M. Ryan and E. L. Deci, “Intrinsic and extrinsic motivations: Classic definitions and new directions,” *Contemporary Educational Psychology*, vol. 25, no. 1, pp. 54–67, 2000.

[40] A. G. Barto and O. Simsek, “Intrinsic motivation for reinforcement learning systems,” in *The Thirteenth Yale Workshop on Adaptive and Learning Systems*, p. 113–118, 2005.

[41] W. Schultz, “Predictive reward signal of dopamine neurons,” *J Neurophysiol*, vol. 80, pp. 1–27, July 1998.

[42] S. M. McClure, N. D. Daw, and P. Read Montague, “A computational substrate for incentive salience,” *Trends in Neurosciences*, vol. 26, no. 8, pp. 423–428, 2003.

[43] P. R. Montague, P. Dayan, and T. J. Sejnowski, “A framework for mesencephalic dopamine systems based on predictive hebbian learning,” *J Neurosci*, vol. 16, pp. 1936–1947, Mar. 1996.

[44] W. Schultz, P. Dayan, and P. R. Montague, “A neural substrate of prediction and reward,” *Science*, vol. 275, no. 5306, pp. 1593–1599, 1997.

[45] P. A. Garris, M. Kilpatrick, M. A. Bunin, D. Michael, Q. D. Walker, and R. M. Wightman, "Dissociation of dopamine release in the nucleus accumbens from intracranial self-stimulation," *Nature*, vol. 398, pp. 67–69, Mar 1999.

[46] M. R. Kilpatrick, M. B. Rooney, D. J. Michael, and R. M. Wightman, "Extracellular dopamine dynamics in rat caudate-putamen during experimenter-delivered and intracranial self-stimulation," *Neuroscience*, vol. 96, no. 4, pp. 697–706, 2000.

[47] G. S. Berns, S. M. McClure, G. Pagnoni, and P. R. Montague, "Predictability modulates human brain response to reward," *J Neurosci*, vol. 21, pp. 2793–2798, Apr. 2001.

[48] G. Pagnoni, C. F. Zink, P. R. Montague, and G. S. Berns, "Activity in human ventral striatum locked to errors of reward prediction," *Nat Neurosci*, vol. 5, pp. 97–98, Feb. 2002.

[49] S. M. McClure, G. S. Berns, and P. R. Montague, "Temporal prediction errors in a passive learning task activate human striatum," *Neuron*, vol. 38, pp. 339–346, Apr. 2003.

[50] J. P. O'Doherty, P. Dayan, K. Friston, H. Critchley, and R. J. Dolan, "Temporal difference models and reward-related learning in the human brain," *Neuron*, vol. 38, pp. 329–337, Apr. 2003.

[51] P. Dayan, "Motivated reinforcement learning," in *Advances in Neural Information Processing Systems* (T. Dietterich, S. Becker, and Z. Ghahramani, eds.), vol. 14, MIT Press, 2002.

[52] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," *31st International Conference on Machine Learning, ICML 2014*, vol. 1, 06 2014.

[53] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems* (Z. Ghahramani,

M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, eds.), vol. 27, Curran Associates, Inc., 2014.

[54] S. Singh, T. Jaakkola, M. L. Littman, and C. Szepesvári, "Convergence results for single-step on-policy reinforcement-learning algorithms," *Machine Learning*, vol. 38, pp. 287–308, Mar 2000.

[55] S. D. Whitehead, "A complexity analysis of cooperative mechanisms in reinforcement learning," in *Proceedings of the Ninth National Conference on Artificial Intelligence - Volume 2*, AAAI'91, p. 607–613, AAAI Press, 1991.

[56] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proceedings of The 33rd International Conference on Machine Learning* (M. F. Balcan and K. Q. Weinberger, eds.), vol. 48 of *Proceedings of Machine Learning Research*, (New York, New York, USA), pp. 1928–1937, PMLR, 20–22 Jun 2016.

[57] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

[58] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov, "Openai baselines." `https://github.com/openai/baselines`, 2017.

[59] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, p. 253–279, Jun 2013.

[60] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008.

[61] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR (Poster)*, 2015.

[62] S. Ruder, "An overview of gradient descent optimization algorithms," 2016.

[63] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[64] P. G. Poličar, M. Stražar, and B. Zupan, "opentsne: a modular python library for t-sne dimensionality reduction and embedding," *bioRxiv*, 2019.

Appendix A

Experimental Details

A.1 Architecture and Hyperparameter Setting

A.1.1 Architecture

The on-policy actor-critic methods, A2C and PPO, employ a single state-value network and a single policy network. Each network consists of two hidden layers, each containing 64 hidden units. Following each hidden layer, a hyperbolic tangent (tanh) activation function is applied. The value networks generate a scalar state-value for each given state s by processing the input through a linear layer. Also, the actor network generates a multi-dimensional action a for each state s it receives as input through a linear layer. The output is then scaled based on the action scale of the environment.

In contrast to the on-policy actor-critic methods, the off-policy methods TD3 and SAC use two Q-networks with target networks and one actor network, following the Clipped Double Q-learning algorithm proposed in TD3 [22]. All networks in TD3 and SAC have two hidden layers with 256 hidden units each and ReLU activation after each layer. The critic networks output a scalar Q-value for each state-action pair (s,a) after passing the input through a linear layer. The policy structure is the same as in the on-policy algorithms. Only TD3 uses a target policy network that is identical to the behavioral policy network.

A.1.2 Network Hyperparameters

We use the Adam optimizer [61] for PPO, SAC, and TD3, while A2C uses RMSProp. [62]. The target networks are updated with polyak averaging after each update step: $\zeta = 0.005$ and $\theta' \leftarrow 0.995 \times \theta' + 0.005 \times \theta$.

A.1.3 Terminal Transitions

We use a discount factor of $\gamma = 0.99$ for the action- and state-value networks in non-terminal transitions and zero in terminal transitions. A transition is terminal only if it ends due to a termination condition, such as failure or time limit exceeded.

A.1.4 Actor-Critic Algorithms

We use the learned entropy variant of SAC [29], where it is optimized to an objective of $-$`action dimensions` using an Adam optimizer with a learning rate of 3×10^{-4}. We clip the log standard deviation to $(20, 2)$ and add a small constant 10^{-6} to avoid numerical instability in the logarithm operation, as specified by the author.

A.1.5 Prioritized Sampling Algorithms and Temporal-Difference Learning

We follow [3] and use $\varepsilon = 0.6$ and $\upsilon = 0.4$ for PER. For our algorithm, we use LAP and PAL functions and set $\varepsilon = 0.4$ and $\upsilon = 0.4$. We did not perform hyperparameter optimization on the ε and υ parameters since they are already fine-tuned [8].

SAC and TD3 use two Q-networks such that they produce two different TD errors: $\delta_1 = y - Q_{\theta_1}$ and $\delta_2 = y - Q_{\theta_2}$, where y is the target value defined earlier in (2.5). We follow [8] and use the maximum value of $|\delta_1|$ and $|\delta_2|$ to determine each priority for optimal performance. In a similar fashion, new samples are given a priority value equal to the maximum priority $p_{\text{init}} = 1$ observed during the learning process, which is consistent with the structure used in PER.

The implementation and algorithmic setup of LA3P are the same for SAC and TD3. The main distinctions between these two actor-critic algorithms are the policy gradient computation (i.e., the use of a stochastic or deterministic policy), entropy tuning, and the existence of a target actor network. As we have shown earlier, Theorem 1 applies to both deterministic and stochastic policies. Hence, algorithmic differences between SAC and TD3 do not affect the implementation and operation of LA3P.

A.1.6 Exploration

The off-policy agents are not trained for the first 25000 time steps and selects actions randomly with uniform probability to fill the experience replay buffer. Then, TD3 explores the action space by adding a Gaussian noise of $\mathcal{N}(0, \sigma_E^2 \times \texttt{max\_action\_size})$, where $\sigma_E = 0.1$ is scaled by the action space range. This approach is also used to create the standard Gaussian noise baseline in the experiments with DISCOVER. SAC does not use any exploration noise as it already uses a stochastic policy.

We follow [11] and set the parameter noise to 0.017 and 0.034 for the parameter-space noise algorithms, which are the best values for on- and off-policy algorithms in practice. Moreover, 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 follows the same actor architecture of the underlying actor-critic algorithm. This includes the network depth and size, learning rate, optimizer, nonlinearity, target and behavioral policy update frequency, target network learning rate, and the number of gradient steps in the updates. Furthermore, we still use the exploration policy in the exploration time steps at the start of each training, yet the action perturbation is drawn from the exploration policy instead.

A.1.7 Hyperparameter Optimization

No hyperparameter optimization was performed on any baseline or competing algorithm except for SAC. We optimized the reward scale for the BipedalWalker, LunarLanderContinuous, and Swimmer tasks, as they were not reported in the original article. We tested the values of $\{5, 10, 20\}$, and found that scaling the rewards by 5 gave the best results in these environments.

All algorithms follow the parameter settings and methodology described in their respective articles or the latest version of their code available on their GitHub repositories. Specifically, the on-policy methods, A2C and PPO use the tuned hyper-parameters for the MuJoCo and Box2D tasks provided by a well-known repository[1]. SAC uses the same hyperparameter configuration as in the original paper, except for increased exploration time steps to 25000 and entropy tuning. For TD3, we used the code from the author's repository[2], which has minor variations in parameter settings compared to the original paper. In particular, the repository code increases the start steps to 25000 and batch size to 256 for all environments, which improves the results.

For LA3P, we tested $\lambda = \{0.1, 0.3, 0.5, 0.7, 0.9\}$ on the Ant, Hopper, Humanoid, and Walker2d tasks, and found that $\lambda = 0.5$ performed the best. For DISCOVER, we tested $\alpha = \{0.1, 0.3, 0.6, 0.9, 1.0\}$ on the HalfCheetah, Humanoid, Hopper, and Swimmer tasks, and inferred that $\alpha = 0.1$ and $\alpha = 0.3$ performed the best for

[1] `https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail`
[2] `https://github.com/sfujim/TD3`

on- and off-policy algorithms, respectively. We have shown these results under different λ and α values in our ablation studies in Chapters 3.4.2 and 4.4.2, respectively. For clarity, all hyperparameters are given in Tables A.1, A.2, and A.3.

Hyperparameter	**Value**
Optimizer	Adam
Learning rate	3×10^{-4}
Minibatch size	256
Discount factor γ	0.99
Target update rate	0.005
Initial exploration steps	25000
TD3 exploration policy σ_E	0.1
TD3 policy noise σ_N	0.2
TD3 policy noise clipping	$(-0.5, 0.5)$
SAC entropy target	`-action dimensions`
SAC log-standard deviation clipping	$(-20, 2)$
SAC log constant	10^{-6}
SAC reward scale (except Humanoid)	5
SAC reward scale (Humanoid)	20
PER priority exponent ε	0.6
PER importance sampling exponent υ	0.4
PER added priority constant	10^{-4}
LAP & PAL exponent ε	0.4
LA3P uniform fraction λ	0.5

Table A.1: The hyperparameters used for the off-policy algorithms.

Hyperparameter	Value
Policy	Multivariate Gaussian
Adam optimizer ϵ	10^{-5}
Adam optimizer α	0.99
Maximum gradient norm	0.5

Table A.2: The shared hyperparameters for the on-policy algorithms.

Hyper-parameter	A2C	PPO
Learning rate*	0.0013	0.0003
Regularization	None	None
Explorer learning rate	0.0013	0.0003
Optimizer	RMSProp	Adam
Number of rollout steps (horizon)	32	2048
Number of minibatches†	1	32
GAE††	False	True
GAE λ	None	0.95
Entropy coefficient	0.0	0.0
Linear decay‡	False	True

* Used for both value and actor networks

† The minibatch size can be computed by: `(# of rollout steps / # of minibatches)`

†† The use of General Advantage Estimation [63]

‡ The use of linear decay on the learning rate

Table A.3: Algorithm specific hyper-parameters used for the implementation of the on-policy algorithms.

A.2 Implementation

A.2.1 Actor-Critic Algorithms

We implemented TD3 using the author's GitHub repository[2]. We implemented SAC manually by following the original paper and adding entropy tuning [29]. In addition, our implementation of A2C and PPO is based on the code from the well-known repository[1], using the tuned hyper-parameters for the continuous control tasks we considered.

A.2.2 Experience Replay Sampling Methods

We used the code from the TD3 author's LAP-PAL repository[3] to implement PER and the LAP and PAL functions. The PER implementation is based on proportional prioritization through the sum tree data structure, as discussed. Moreover, we directly used the MaPER code from the paper's submission files from the OpenReview website[4]. No changes were made to the MaPER code.

The implementation of LA3P includes the cascaded uniform, prioritized, and inverse prioritized sampling, which precisely follows the pseudocode in Algorithm 1. We do not update the priorities after the actor update with inverse prioritized sampling since the PER implementation with standard actor-critic algorithms only updates the priorities after each Q-network update.

A.2.3 Exploration Baselines

We implemented NoisyNet by adapting the code from authors' GitHub repository[2] to the baseline actor-critic algorithms. Authors' OpenAI Baselines implementation[3] was used to implement PSNE. Similar to SAC, we referred to the

[3]`https://github.com/sfujim/LAP-PAL`
[4]`https://openreview.net/forum?id=WuEiafqdy9H`

original papers in implementing Deep Coherent Exploration since the authors did not provide a valid code repository.

A.3 Experimental Setup

A.3.1 Simulation Environments

All agents are evaluated on the continuous control benchmarks of the MuJoCo[5] and Box2D[6] physics engines, which are interfaced by OpenAI Gym[7] using `v2` environments. The state-action spaces and reward functions of the environments were not changed or preprocessed for practical reproducibility and fair comparison with the empirical findings. Each environment has a multi-dimensional action space with values between $[-1, 1]$, except Humanoid, which has a range of $[-0.4, 0.4]$.

A.3.2 Evaluation

An evaluation is performed every 1000 time steps, using the average reward over 10 episodes, with the deterministic policy from TD3 without exploration noise and the deterministic mean action from A2C, PPO, and SAC. We use a new environment with a fixed seed (training seed + constant) for each evaluation to reduce the variation caused by different seeds. Thus, each evaluation uses the same set of initial start states. We used different sets of random seeds for LA3P and DISCOVER in our experiments. As a result, the performance of the actor-critic methods, SAC and TD3, varied.

[5] `https://mujoco.org/`

[6] `https://box2d.org/`

[7] `https://www.gymlibrary.ml/`

A.3.3 Visualization of the Learning Curves

The performance is shown by the learning curves, which are an average of 10 trials with a shaded region showing a 95% confidence interval over the trials. The curves are smoothed uniformly with a sliding window of 5 evaluations for visual clarity.

A.3.4 Statistical Testing for the Evaluation Results

We follow [30] and used Python's SciPy library[8] to conduct the 2-sample t-test. We used the "greater" option for the `alternative` parameter in the `ttest_ind` function to test if the mean of the last 10 rewards of LA3P is higher than that of the competing algorithm. Thus, a p-value less than 0.05 means enough evidence to reject the null hypothesis and confirm with 95% confidence that the mean of the last 10 rewards of our algorithm is higher than that of the competing algorithm. Otherwise, if the p-value is more than 0.05, we cannot reject the null hypothesis, and therefore, cannot claim that there is a significant difference between the means of the two reward curves.

A.3.5 Visualization of the State Visitations

To compare the greedy action selection and DISCOVER methods under the TD3 algorithm in the Swimmer environment, we plot the states from the collected transitions over 1 million time steps. We use a single seed for this experiment. We exclude the first 25000 transitions from the replay buffer as they are sampled from the action space of the environment and consider the remaining 975000 transitions. We apply PCA to reduce the visual artifacts and project the datasets onto a 4D state space separately.

[8] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html`

	Result	PER	LA3P
SAC	Runtime - 1 million steps (mins)	312.92 ± 1.63	459.18 ± 1.47
	Runtime - 2 million steps (mins)	536.46 ± 1.61	858.50 ± 1.50
	Time increase (%)	171.44%	186.96%
TD3	Runtime - 1 million steps (mins)	145.83 ± 2.09	238.29 ± 2.13
	Runtime - 2 million steps (mins)	252.21 ± 2.15	437.18 ± 2.08
	Time increase (%)	172.95%	183.47%

Table A.4: The average runtime of PER and LA3P for 1 million and 2 million training steps under the SAC and TD3 algorithms, and the corresponding percentage increase. The values are averaged over 10 random seeds and the Ant, HalfCheetah, Humanoid, and Swimmer environments. A replay buffer of size equal to the number of training steps is used in all experiments. ± captures a 95% confidence interval over the runtime.

We use the openTSNE library[9] [64] to embed the datasets onto a 2D state space using t-SNE. We set the perplexity to 1396 and the distance metric to euclidean. We run t-SNE for 1500 iterations. We use the default values for the other parameters in openTSNE. We divide the datasets into three parts and plot them separately, with each part having 325000 samples. The proportion of variance explained by PCA is 0.984 for the greedy dataset and 0.966 for the DISCOVER dataset.

A.4 Empirical Complexity Analysis of LA3P

We compare the runtime of PER and our algorithm on four environments: Ant, HalfCheetah, Humanoid, and Swimmer. We used replay buffer sizes that match the number of training steps. We use SAC and TD3 as the off-policy actor-critic algorithms for both sampling methods. We measured the total runtime for 1 million and 2 million training steps. We run all experiments on a single GeForce RTX 2070 SUPER GPU and an AMD Ryzen 7 3700X 8-Core Processor, which are compatible with the SIMD operations. Table A.4 shows the results of our comparison.

[9] `https://opentsne.readthedocs.io/en/latest/`

We make two observations from our results. First, SAC has a higher runtime than TD3. This is because SAC needs to tune the entropy and keep a stochastic actor, which require an additional backpropagation and increases the runtime. Second, our results agree with the theoretical upper bound of $\mathcal{O}(\log|R|)$ for the runtime of PER, which grows logarithmically with the replay buffer size instead of doubling. When we doubled the replay buffer size from 1 million to 2 million, we also saw that the empirical runtime of LA3P increased. However, the increase was not as severe as expected based on the theoretical runtime of $\mathcal{O}(|R|)$. We note that theoretical runtime analysis gives an upper bound on the runtime, and actual empirical runtime may vary due to various practical factors that are not accounted for in the theoretical analysis. Moreover, SIMD operations could be one of the factors that affect the observed behavior. Specifically, the main computational complexity of LA3P comes from taking the inverse of each element of the replay buffer array by multiplication to determine which transitions to sample for training the actor network. This operation can be optimized for SIMD operations, as long as the inverse operation is well-defined for each element and there are no division-by-zero errors. In our implementation, we already use the clipping defined in (3.18) to avoid zero probability values. Therefore, the inverse operation is always well-defined for each element, and we can attribute the limited runtime of LA3P to the SIMD operations.