An Improved Trust-Region Method for Off-Policy Deep Reinforcement Learning

Hepeng Li[1], Xiangnan Zhong[2], and Haibo He[1]

[1]Department of Electrical, Computer, and Biomedical Engineering, University of Rhode Island
Kingston, RI, USA, Email: hepengli@uri.edu and haibohe@uri.edu
[2]Department of Electrical Engineering and Computer Science, Florida Atlantic University
Boca Raton, FL, USA, Email: xzhong@fau.edu

2023 International Joint Conference on Neural Networks (IJCNN) | 978-1-6654-8867-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/IJCNN54540.2023.10191837

***Abstract*—Reinforcement learning (RL) is a powerful tool for training agents to interact with complex environments. In particular, trust-region methods are widely used for policy optimization in model-free RL. However, these methods suffer from high sample complexity due to their on-policy nature, which requires interactions with the environment for each update. To address this issue, off-policy trust-region methods have been proposed, but they have shown limited success in high-dimensional continuous control problems compared to other off-policy DRL methods. To improve the performance and sample efficiency of trust-region policy optimization, we propose an off-policy trust-region RL algorithm. Our algorithm is based on a theoretical result on a closed-form solution to trust-region policy optimization and is effective in optimizing complex nonlinear policies. We demonstrate the superiority of our algorithm over prior trust-region DRL methods and show that it achieves excellent performance on a range of continuous control tasks in the Multi-Joint dynamics with Contact (MuJoCo) environment, comparable to state-of-the-art off-policy algorithms.**

I. Introduction

Model-free reinforcement learning (RL) has become a popular approach for solving complex sequential decision-making problems [1], [2] and optimal control tasks [3]. This success can be attributed to the progress made in deep RL (DRL) algorithms that combine traditional RL methods with deep learning techniques [4] to scale up to high-dimensional problems. DRL algorithms can be categorized into two types: value-based methods and policy-based methods. Value-based methods, such as deep Q-learning (DQN) [1] and double DQN [5], learn an action-value function using deep neural networks and represent the policy with a greedy or $\epsilon-$greedy action-selection strategy based on the estimated action-value function. In contrast, policy-based methods, such as REINFORCE [6] and policy gradient [7], learn an explicit policy that is parameterized using a deep neural network that maps states to a policy distribution over the action space. Policy search methods are generally more effective in high-dimensional or continuous action control tasks and have become increasingly popular in model-free RL.

However, traditional policy-based methods in reinforcement learning, such as REINFORCE [6], policy gradient [7], actor-critic [8], [9], and approximate dynamic programming [10], [11], often encounter challenges with learning stability and hyperparameter sensitivity [12], [13]. They typically utilize first-order policy gradients to update policy parameters, which can lead to policy collapse due to an inappropriate choice of step-size. In practice, it can be difficult to determine the optimal step-size to ensure performance improvement in the direction of gradient ascent, particularly when using nonlinear function approximation.

Trust-region methods have been introduced to policy-based RL to improve stability by constraining policy updates to a local area surrounding the most recent policy, or the "trust region." Different trust-region RL algorithms have been developed based on how the trust-region distance is measured. For example, the natural policy gradient [14] calculates the trust region distance using a quadratic metric over policy parameters, which is determined using the Fisher information matrix. Trust region policy optimization [15] measures the distance using KL-Divergence and provides a monotonic improvement guarantee on policy performance, but it requires second-order gradients and is computationally inefficient. Proximal policy optimization [16] approximates the KL distance by clipping the policy ratio in the objective, allowing for the use of first-order optimization methods. In [17], differentiable trust-region layers for Gaussian policies are proposed based on Wasserstein L2 distance, Frobenius norm, and KL-Divergence, and provide guarantees of monotonic improvement. However, these methods are sample inefficient due to their need for a large number of on-policy interactions with the environment, making them impractical for real-world applications.

Recently, off-policy DRL algorithms have been proposed to improve the sample efficiency of trust-region methods. In [18], a trust region path consistency learning algorithm is proposed to leverage off-policy data for policy learning and value function estimation. Considering RL as a variational inference paradigm [19], [20] proposes an off-policy trust-region RL algorithm for Gaussian policies using expectation-maximization (EM) via maximum-a-posteriori estimation. In [21], a new surrogate function-based off-policy trust-region policy optimization algorithm is proposed. [22] proposes an off-policy DRL algorithm using mirror descent policy optimization, which projects gradients in a dual space of the policy onto a trust region under a Bregman divergence. In [23], off-

This material is based upon work supported by the National Science Foundation under Grant No. ECCS 1917275 and ECCS 1947419.

policy variants of trust-region policy optimization, proximal policy optimization, and mirror descent policy optimization are developed using functional mirror ascent to construct a general class of surrogate functions. Despite these advancements, their ability to compete with other policy-based DRL algorithms, such as deep deterministic policy gradient [3], [24] and maximum entropy RL [25]–[28], on high-dimensional continuous control tasks is still unsatisfactory.

In this paper, our focus is on improving the performance and sample efficiency of trust-region RL methods for continuous control tasks. Our objective is to derive a stable and sample-efficient trust-region policy optimization algorithm for model-free RL. The paper makes two main contributions. First, building on the closed-form solution to trust region policy optimization, we develop a practical deep RL algorithm that enables the exploitation of off-policy data for policy optimization using trust-region methods. Second, we provide a detailed implementation of the algorithm for parameterized policies, such as deep neural networks, based on first-order gradients that is practical for real-world applications. We evaluate the proposed algorithm on a variety of Multi-Joint dynamics with Contact (MuJoCo) robot control tasks, and demonstrate that it is effective at solving high-dimensional continuous control tasks, with significant improvements in the final return and sample efficiency over prior trust-region methods.

This paper is organized as follows. Section II presents the necessary preliminaries to understand the proposed algorithm. In Section III, we introduce the proposed algorithm and provide detailed information on its implementation. Section IV evaluates the effectiveness of the proposed method through comparison studies. Finally, we conclude our findings and discuss potential future work in Section V.

II. Preliminaries

A. Markov Decision Process

In this paper, we model model-free RL as an infinite-horizon Markov decision process (MDP), defined by $(\mathcal{S}, \mathcal{A}, p, r, \rho_0, \gamma)$, where $\mathcal{S}$ is the space of environment state s, $\mathcal{A}$ is the space of agent action a, $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_{\geq 0}$ is the unknown state transition density, $r : \mathcal{S} \times \mathcal{A} \to [r_{\min}, r_{\max}]$ is the reward function bounded by $r_{\min}$ and $r_{\max}$, $\rho_0 : \mathcal{S} \to \mathbb{R}_{\geq 0}$ is the distribution of the initial state s_0, γ is the discount factor, which can take any value in the interval $[0, 1)$.

Denote τ as the Markov process consisting of the sequence of states and actions, $\tau := (s_0, a_0, s_1, \dots)$, and $R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t$ as the discounted return of the trajectory τ. A policy $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_{\geq 0}$ is a conditional probability distribution of the action a given the state s, and it is denoted by $\pi(a|s)$. The objective of RL is to maximize the expected discounted return with respect to π:

$$\max_{\pi} \; J(\pi) = \mathbb{E}_{\tau \sim \pi} \left[R(\tau) \right] \tag{1}$$

where $\tau \sim \pi$ indicates that the Markov process τ depends on the policy π:

$$s_0 \sim \rho_0, a_t \sim \pi(\cdot|s_t), s_{t+1} \sim p(\cdot|s_t, a_t).$$

We also define as follows the state-value function

$$V_\pi(s) = \mathbb{E}_{\tau \sim \pi} \left[R(\tau) | s_0 = s \right], \tag{2}$$

the action-value function (Q-function)

$$Q_\pi(s, a) = \mathbb{E}_{\tau \sim \pi} \left[R(\tau) | s_0 = s, a_0 = a \right], \tag{3}$$

and the advantage function

$$A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s), \tag{4}$$

Also, define the discounted state visitation distribution by

$$\begin{aligned} d^\pi(s) &= (1 - \gamma) \left[\rho_0^\pi(s) + \gamma \rho_1^\pi(s) + \gamma^2 \rho_2^\pi(s) + \cdots \right] \\ &= (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \rho_t^\pi(s). \end{aligned} \tag{5}$$

where $\rho_t^\pi : \mathcal{S} \to \mathbb{R}_{\geq 0}$ is probability distribution of the state at step t under the policy π.

B. Trust-Region RL Methods

Trust-region RL methods are a type of policy-based method that improve stability by restricting the policy search to a local neighborhood around the most recent update, e.g.:

$$\begin{aligned} &\max_{\pi \in \Pi} \; J(\pi) \\ &s.t. \; D(\pi, \pi_k) \leq \delta \end{aligned} \tag{6}$$

where π_k is the most recent update of the policy, D measures the trust-region distance between π and π_k, and $\delta > 0$ defines the size of the trust region.

However, due to the unknown nature of the environment, we do not have an explicit expression of $J(\pi)$ in terms of π. This makes solving the constrained optimization problem (6) a challenging task. However, based on Lemma 6.1 in [29], the objective function $J(\pi)$ can be expressed as a function of π_k.:

$$J(\pi) = J(\pi_k) + \frac{1}{1 - \gamma} \mathbb{E}_{s \sim d^\pi, a \sim \pi} [A_{\pi_k}(s, a)]. \tag{7}$$

Although the distribution d^π is unknown, it can be approximated by d^{π_k} when they are similar. As a result, trust-region methods use the following surrogate to approximate the objective function.

$$L_{\pi_k}(\pi) = J(\pi_k) + \frac{1}{1 - \gamma} \mathbb{E}_{s \sim d^{\pi_k}, a \sim \pi} [A_{\pi_k}(s, a)]. \tag{8}$$

It is proven that the approximation error between the surrogate $L_{\pi_k}(\pi)$ and the objective $J(\pi)$ is bounded by [15]:

$$\begin{aligned} &\left| J(\pi) - L_{\pi_k}(\pi) \right| \leq C \max_s D_{\mathrm{KL}}[\pi \| \pi_k](s), \\ &\textit{where } C = \frac{4\gamma\epsilon}{(1 - \gamma)^2}, \; \epsilon = \max_{s,a} |A_{\pi_k}(s, a)|, \end{aligned} \tag{9}$$

Rewriting this inequality provides a lower bound on the objective:

$$J(\pi) \geq L_{\pi_k}(\pi) - C \max_s D_{\mathrm{KL}}[\pi \| \pi_k](s), \tag{10}$$

Then, we can instead maximize the lower bound to derive a new policy:

$$\pi_{k+1} = \arg\max_{\pi} \; L_{\pi_k}(\pi) - C \max_s D_{\mathrm{KL}}[\pi \| \pi_k](s) =$$
$$\max_{\pi} \; \frac{1}{1-\gamma} \mathbb{E}_{s\sim d^{\pi_k}, a\sim\pi}[A_{\pi_k}(s,a)] - C \max_s D_{\mathrm{KL}}[\pi \| \pi_k](s) \tag{11}$$

It follows that the new policy guarantees monotonic improvement in performance, i.e., $J(\pi_{k+1}) \geq J(\pi_k)$.

However, optimizing the lower bound directly can be impractical for problems with a high-dimensional state space because it requires an accurate estimate of the maximal KL over the state space, $\max_s D_{\mathrm{KL}}[\pi \| \pi_k](s)$. To address this issue, TRPO [15] approximates the optimization by using an importance sampling objective with a constraint on the expected KL:

$$\begin{aligned} \max_{\pi} \; & \mathbb{E}_{s\sim d^{\pi_k}, a\sim\pi_k}\left[\frac{\pi(a|s)}{\pi_k(a|s)} A_{\pi_k}(s,a)\right] \\ s.t. \; & \mathbb{E}_{s\sim d^{\pi_k}}\left[D_{KL}[\pi_k || \pi](s)\right] < \delta. \end{aligned} \tag{12}$$

To update the policy π using the above method, it is necessary to collect on-policy samples at every training step k from $s \sim d^{\pi_k}$, which are used to estimate the expectation terms in the objective and the KL constraint. This makes the method sample inefficient because it requires massive interactions with the environment for each policy update.

III. An Off-Policy Trust-Region DRL Algorithm

In this section, we present an off-policy DRL algorithm based on trust-region policy optimization methods that is suitable for practical applications. We begin by introducing a theoretical result established in [30]. We then propose a trust-region DRL algorithm that allows us to use parameterized function approximators in practice. This enables us to exploit off-policy data, which is more efficient than on-policy data collection. Finally, we provide a detailed description of the implementation of this algorithm for parameterized policies.

A. Closed-Form Solution to Trust-Region Policy Optimization

To address the challenge of optimizing the lower bound in Equation (10), Li et al. [30] proposed a new bound for the approximation error when $\gamma \in [0.5, 1)$:

$$\left|J(\pi) - L_{\pi_k}(\pi)\right| \leq \frac{1}{1-\gamma} C_{\pi_k} \mathbb{E}_{s\sim d^{\pi_k}}\left[D_{\mathrm{KL}}[\pi \| \pi_k](s)\right],$$
$$\text{where } C_\pi = \frac{\gamma^2 \epsilon}{(1-\gamma)^3}, \; \epsilon = \max_{s,a} |A_\pi(s,a)|. \tag{13}$$

This bound improves the result of TRPO by establishing a connection between the approximation error and the expected KL-Divergence $\mathbb{E}_{s\sim d^{\pi_k}}[D_{\mathrm{KL}}[\pi \| \pi_k](s)]$. Using this new bound, we obtain the following lower bound:

$$J(\pi) \geq L_{\pi_k}(\pi) - \frac{1}{1-\gamma} C_{\pi_k} \mathbb{E}_{s\sim d^{\pi_k}}\left[D_{\mathrm{KL}}[\pi \| \pi_k](s)\right]. \tag{14}$$

Since the new lower bound does not depend on the maximum KL compared with (10), it is easier to directly maximize the lower bound to obtain a new policy:

$$\pi_{k+1} = \arg\max_{\pi} \; L_{\pi_k}(\pi) - \frac{1}{1-\gamma} C_{\pi_k} \mathbb{E}_{s\sim d^{\pi_k}}\left[D_{\mathrm{KL}}[\pi \| \pi_k](s)\right] \tag{15}$$

The new policy π_{k+1} guarantees monotonic improvement in performance, i.e., $J(\pi_{k+1}) \geq J(\pi_k)$. As proven in [30], the above optimization has a closed-form solution for π_{k+1}:

$$\pi_{k+1}(a|s) = \pi_k(a|s) \cdot \frac{e^{\alpha_{\pi_k}(s,a)}}{\mathbb{E}_{a\sim\pi_k}\left[e^{\alpha_{\pi_k}(s,a)}\right]}, \forall s, a \tag{16}$$

where $\alpha_{\pi_k}(s,a) = A_{\pi_k}(s,a)/C_{\pi_k}$.

Compare with the TRPO algorithm in (12), Equation (16) provides a new policy update method that does not involve on-policy calculation for the expectation term $\mathbb{E}_{s\sim d^{\pi_k}}[\cdot]$. This enables the development of an off-policy trust-region RL algorithm, which is presented in the next section.

B. An Off-Policy Trust-Region RL Algorithm

As mentioned in the previous section, the policy π_{k+1} in Equation (16) guarantees monotonic improvement. To optimize any arbitrary policy π, we can minimize a loss function that measures the distance between π and π_{k+1}. In our proposed algorithm, we use the square error loss defined as follows:

$$\min_{\pi} \iint \left[\frac{\pi(a|s)}{\pi_k(a|s)} - \frac{e^{A_{\pi_k}(s,a)/C_{\pi_k}}}{\mathbb{E}_{a\sim\pi_k}[e^{A_{\pi_k}(s,a)/C_{\pi_k}}]}\right]^2 dsda. \tag{17}$$

However, in continuous control problems, it is not practical to compute the accurate policy $\pi_k(a|s)$ and the advantage function $A_{\pi_k}(s,a)$ for all (s,a). Instead, function approximators are usually used to represent the policy and estimate the advantage values. Moreover, computing the integral exactly is also challenging. To address these issues, we derive a practical algorithm based on the loss function presented earlier.

Specifically, we will consider a parameterized policy $\pi_\theta(a|s)$, and a parameterized Q-function $Q_\phi(s,a)$, and the parameters of these differentiable approximators are θ and ϕ, respectively.

To optimize the policy parameters θ, we propose to minimize the following mean square error (MSE) loss:

$$J_\pi(\theta) = \frac{1}{2} \mathbb{E}_{s_t\sim\mathcal{D}, a_t\sim\pi_{\theta^k}}\left[(r_\theta(s_t,a_t) - \hat{r})^2\right] \tag{18}$$

where

$$r_\theta(s,a) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta^k}(a_t|s_t)}, \quad \hat{r} = \frac{e^{A_{\pi_{\theta^k}}(s_t,a_t)/C_{\pi_{\theta^k}}}}{\mathbb{E}_{a\sim\pi_{\theta^k}}\left[e^{A_{\pi_{\theta^k}}(s_t,a_t)/C_{\pi_{\theta^k}}}\right]}.$$

The MSE loss (18) aims to minimize the expected square error between the policy ratio r_θ and its target $\hat{r}$ using the most recent iterate of the policy parameters θ^k and a replay buffer $\mathcal{D}$ that stores the past visited states during training. Specifically, the optimization is carried out over the past visited states $s \sim \mathcal{D}$ and the actions following the most recent policy $a \sim \pi_{\theta^k}$.

It's worth noting that this policy update using (18) is off-policy, because it doesn't require the expectation over the on-policy states $\mathbb{E}_{s\sim d^{\pi_k}}[\cdot]$, as is the case with the TRPO algorithm (12).

Gradient decent can be used to minimize the MSE loss. The gradient of $J_\pi(\theta)$ can be expressed as

$$\nabla_\theta J_\pi(\theta) = \mathbb{E}_{s_t\sim\mathcal{D}, a_t\sim\pi_{\theta^k}}\left[\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta^k}(a_t|s_t)} - \hat{r}\right)\frac{\nabla_\theta \pi_\theta(a_t|s_t)}{\pi_{\theta^k}(a_t|s_t)}\right]. \tag{19}$$

At $\theta = \theta_k$, we have

$$\nabla_\theta J_\pi(\theta)|_{\theta=\theta^k} = \mathbb{E}_{s_t\sim\mathcal{D}, a_t\sim\pi_{\theta^k}}\left[(1-\hat{r})\nabla_\theta \log \pi_\theta(a_t|s_t)\right]. \tag{20}$$

Since neural networks are powerful and common differentiable function approximators, we will use the terms of policy network and Q-network to denote the parameterized policy and the parameterized Q-function. We will update both networks alternately with mini-batch stochastic gradient descent (SGD) as the way of training in deep deterministic policy gradient [3]. In practice, the policy parameters θ are optimized with the mini-batch SGD:

$$\hat{\nabla}_\theta J_\pi(\theta)|_{\theta=\theta^k} = \frac{1}{B}\sum_{b=1}^{B}\sum_{n=1}^{N}(1-\hat{r}_n^b)\nabla_\theta \log \pi_\theta(a_t^n|s_t^b) \tag{21}$$

where the target policy ratio $\hat{r}_n^b$ is calculated by

$$\hat{r}_n^b = \frac{\exp\{A_{\pi_{\theta^k}}(s_t^b, a_t^n)/C_{\pi_{\theta^k}}\}}{\frac{1}{N}\sum_{n=1}^{N}\left[\exp\{A_{\pi_{\theta^k}}(s_t^b, a_t^n)/C_{\pi_{\theta^k}}\}\right]}. \tag{22}$$

The states s_t^b are sampled from the replay buffer $\mathcal{D}$, and the actions a_t^n are sampled from the most recent policy π_{θ^k} given s_t^b. Note that obtaining action samples from π_{θ^k} does not require additional interactions with the environment, and therefore we can update the policy in an off-policy manner using (21).

To compute the target policy ratio $\hat{r}_n^b$, we need to know the advantage function $A_{\pi_{\theta^k}}(s,a)$ with respect to the most recent policy π_{θ^k}. To this end, our algorithm makes use of a target policy with parameters $\overline{\theta}$ as the most recent iterate θ^k. Based on the definition in Equation (4), we can calculate the advantage value with the Q-function:

$$\begin{aligned} A_{\pi_{\overline{\theta}}}(s_t, a_t) =& Q_{\pi_{\overline{\theta}}}(s_t, a_t) - V_{\pi_{\overline{\theta}}}(s_t) \\ =& Q_{\pi_{\overline{\theta}}}(s_t, a_t) - \mathbb{E}_{a_t\sim\pi_{\overline{\theta}}}[Q_{\pi_{\overline{\theta}}}(s_t, a_t)]. \end{aligned} \tag{23}$$

Not the advantage function is defined as a function over the whole state and action space, rather than a specific trajectory. In our algorithm, we estimate the advantage function by randomly sampling state s_t from a buffer and taking actions by following the policy $\pi_{\overline{\theta}}$. Therefore, the sampled values do not necessarily belong to the same trajectory. In practice, the advantage function is estimated by using the Q-network $Q_\phi(s_t, a_t)$ as follows:

$$A_{\pi_{\overline{\theta}}}(s_t, a_t) = Q_\phi(s_t, a_t) - \frac{1}{N}\sum_{n=1}^{N}\left[Q_\phi(s_t, a_t^n)\right], \tag{24}$$

where the state are sampled according to $s_t \sim \mathcal{D}$ and the actions are sampled according to $a_t^n \sim \pi_{\overline{\theta}}$.

The Q-network is trained to minimize the mean square Bellman error as follows:

$$J_Q(\phi) = \frac{1}{2}\mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}\left[\left(Q_\phi(s_t, a_t) - \hat{Q}(s_t, a_t)\right)^2\right] \tag{25}$$

where the targe Q-value is

$$\hat{Q}(s_t, a_t) = r_t + \gamma\mathbb{E}\left[Q_\phi(s_{t+1}, a_{t+1})\right], a_{t+1}\sim\pi_\theta. \tag{26}$$

In our algorithm, the Q-network is optimized with the mini-batch SGD

$$\begin{aligned} \hat{\nabla}_\phi J_Q(\phi) =& \frac{1}{B}\sum_{b=1}^{B}\nabla_\phi Q_\phi(s_t^b, a_t^b)\cdot\Big(Q_\phi(s_t^b, a_t^b) - \\ & r_t^b - \gamma Q_{\overline{\phi}}(s_{t+1}^b, a_{t+1}^b)\Big), \end{aligned} \tag{27}$$

where B is the batch size, s_t^b, a_t^b, r_t^b, and s_{t+1}^b are sampled from a reply buffer $\mathcal{D}$, and a_{t+1}^b is sampled from π_θ. The update makes use of a target Q-function to compute the Bellman error, and the parameters $\overline{\phi}$ of the target Q-function are obtained as a moving average of the SGD iterates of the Q-function parameters, which has been shown to help stabilize Q-function learning [5]. The update also makes use of two Q-functions to mitigate overestimated value estimates, as proposed in [31]. Specifically, we construct two Q-networks with parameters $\phi_i, i = 1, 2$, which are trained independently using the mini-batch SGD in Eq. (27). We also obtain two target Q-functions with $\overline{\phi}_i, i = 1, 2$ using Polyak moving averages of the SGD updates of $\phi_i, i = 1, 2$. Then, we use the minimum of the target Q-functions as the target Q-value in Eq. (27).

To compute the target policy ratio $\hat{r}_n^b$, we also need the value of $C_{\pi_{\overline{\theta}}}$ (note $\overline{\theta} = \theta^k$), which requires the exact value of $\max_{s,a}|A_{\pi_{\overline{\theta}}}(s,a)|$. Computing the exact value of $\max_{s,a}|A_{\pi_{\overline{\theta}}}(s,a)|$ can be challenging for high-dimensional, continuous problems. Besides, if we used the coefficient $C_{\pi_{\overline{\theta}}}$ recommended by the theory in (13), the step sizes would be very small. In our algorithm, we approximate it using a sliding window maximum of sampled estimates with a window size w:

$$C_{\pi_{\overline{\theta}}} := \max\{C_{k-w}, \ldots, C_{k-1}, C_k\} \tag{28}$$

where

$$C_{k-w} = \frac{\gamma^2\epsilon}{(1-\gamma)^2},\ \epsilon = \max\{|A_{\pi_{\theta^{k-w}}}(s_t^b, a_t^n)|, \forall b, n\}. \tag{29}$$

The policy parameters are updated by applying $\theta := \theta - \lambda_\pi\nabla_\theta J_\pi(\theta)$, and the target policy parameters are then completely replaced with the most recent iterate of the policy parameters, i.e. $\overline{\theta} := \theta$.

To ensure bounded actions, a squashed Gaussian is used as the policy distribution, as proposed in [27]. In particular, let u be a multivariate random variable following the diagonal gaussian distribution $\mu(u|s)$. Then, the action a is a function of u:

$$a = l + \frac{h-l}{2}(\tanh(u) + 1) \tag{30}$$

Algorithm 1 An Off-policy Trust-Region DRL Algorithm

Input: Initial parameters θ, $\overline{\theta}$, ϕ_1, ϕ_2, $\overline{\phi}_1$, $\overline{\phi}_2$;
for $k = 0, 1, 2, \ldots$ **do**
  for each environment step **do**
    $a_t \sim \pi_\theta(\cdot|s_t)$;
    $s_{t+1} \sim P(\cdot|s_t, a_t)$;
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{s_t, a_t, r_t, s_{t+1}\}$;
  end for
  for each gradient step **do**
    $\phi_i \leftarrow \phi_i - \lambda_Q \hat{\nabla}_{\phi_i} J_Q(\phi_i), i \in \{1, 2\}$;
    $\overline{\phi}_i \leftarrow \beta\overline{\phi}_i + (1-\beta)\phi_i, i \in \{1, 2\}$;
    $\theta \leftarrow \theta - \lambda_\pi \nabla_\theta J_\pi(\theta)$;
    $\overline{\theta} \leftarrow \theta$;
  end for
end for

where tanh is applied elementwise; l and h are the lower and upper bound of the action a. Based on distribution function technique, the action distribution can be calculated by $\pi(a|s) = \mu(a|s)\left|\det\left(\frac{da}{du}\right)\right|^{-1}$. Then, we can calculate the log-likelihood of action a by

$$\log \pi(a|s) = \log \mu(a|s) - \sum_{i=1}^{|\mathcal{A}|} \log\left(1 - \tanh^2(u_i)\right) - \sum_{i=1}^{|\mathcal{A}|} \log\left(\frac{h_i - l_i}{2}\right) \tag{31}$$

where h_i, l_i, u_i are the ith elements of h, l, u, respectively. The algorithm is summarized in 1.

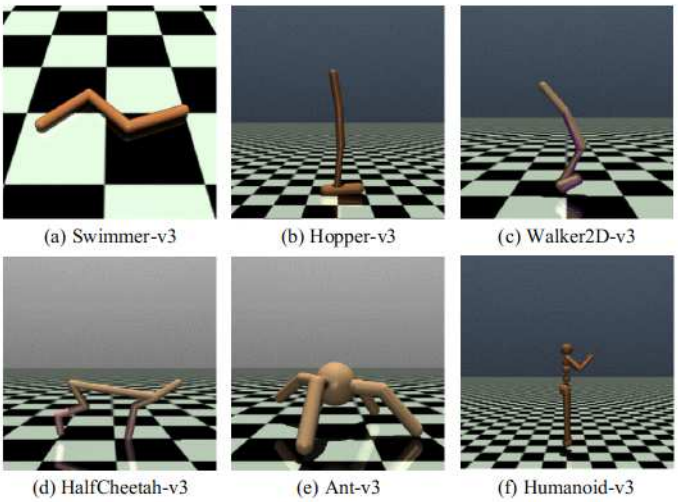
IV. Experiments

We evaluate the effectiveness of the proposed method on a range of high-dimensional, continuous robot control tasks using the MuJoCo simulator [32], which is interfaced through the OpenAI Gym environment [33]. Our aim is to compare the proposed method with prior trust-region-based DRL methods and state-of-the-art off-policy algorithms in terms of their performance and sample efficiency.

A. Setup

We use six robotic locomotion tasks from the MuJoCo simulator as benchmarks for our experiments. These tasks include (a) Swimmer-v3, (b) Hopper-v3, (c) Walker2D-v3, (d) HalfCheetah-v3, (e) Ant-v3, and (f) Humanoid-v3, as illustrated in Fig. 1. The state and action spaces for these tasks are continuous and have varying dimensions, ranging from 8 states and 2 actions (Swimmer-v3) to 376 states and 17 actions (Humanoid-v3). While some tasks with low-dimensional state and action spaces are easier to solve, high-dimensional benchmarks such as Humanoid-v3 are extremely challenging for prior trust-region deep reinforcement learning algorithms.

To demonstrate the effectiveness of our proposed algorithm, we compare it with three trust-region deep reinforcement learning (DRL) methods: (1) trust-region policy optimization (TRPO) [15], an on-policy algorithm that uses second-order gradients and a line search procedure for policy parameter updates; (2) proximal policy optimization (PPO) [16], a variant of TRPO that uses first-order gradients and policy ratio clipping; and (3) trust region path consistency learning (Trust-PCL) [18], an off-policy trust-region algorithm that has shown better performance and efficiency than TRPO. We also compare our proposed algorithm with two off-policy DRL algorithms: (1) deep deterministic policy gradient (DDPG) [3], which learns a deterministic policy for continuous control tasks, and (2) soft actor-critic (SAC) [27], a state-of-the-art DRL algorithm based on maximum entropy policy optimization. We implemented the TRPO, PPO, DDPG, and SAC algorithms using the OpenAI Spinning Up module [34]. For Trust-PCL, we used the implementation provided by the authors in their paper [18].



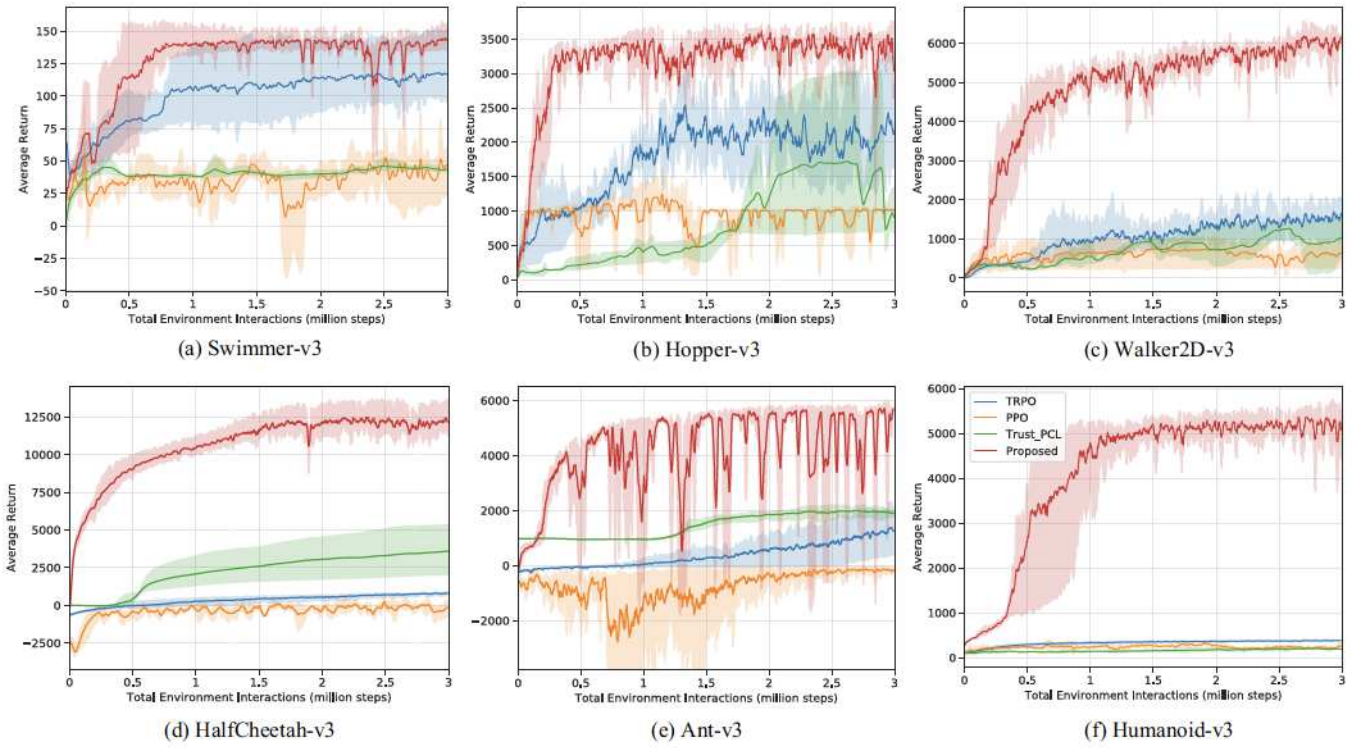
(a) Swimmer-v3 (b) Hopper-v3 (c) Walker2D-v3
(d) HalfCheetah-v3 (e) Ant-v3 (f) Humanoid-v3

Fig. 1. Six representative MuJoCo robot control tasks [32] used in our evaluation.

In our study, we use a feedforward neural network with two hidden layers of 256 ReLU units to parameterize the policy, which is the same across our method and the baseline algorithms. The value network (for TRPO, PPO, Trust-PCL) and the Q-network (for DDPG, SAC, and our method) share the same architecture in the hidden layers, but differ in the dimensions of the inputs and outputs. We implement our method and the baseline algorithms using TensorFlow [35]. We train the policy for 3 million environment steps and evaluate it after every 2000 steps by running 10 episodes and selecting the mean of the policy distribution as the agent's action for optimal performance. The hyperparameters of our method are summarized in Table I.

B. Comparison With Prior Trust-Region methods

The learning curves of the proposed algorithm are compared to those of prior trust-region DRL methods on six representative MuJoCo continuous control tasks. The solid curves represent the mean performance of different algorithms over



(a) Swimmer-v3 (b) Hopper-v3 (c) Walker2D-v3

(d) HalfCheetah-v3 (e) Ant-v3 (f) Humanoid-v3

Fig. 2. Learning curves of the proposed algorithm in contrast to prior trust-region DRL methods on six representative MuJoCo continuous control tasks. The solid curves represent the mean performance of different algorithms over five random seeds, and the shaded areas correspond to the 95% confidence intervals.

TABLE I
PARAMETERS OF THE PROPOSED OFF-POLICY TRUST-REGION DEEP REINFORCEMENT LEARNING ALGORITHM.

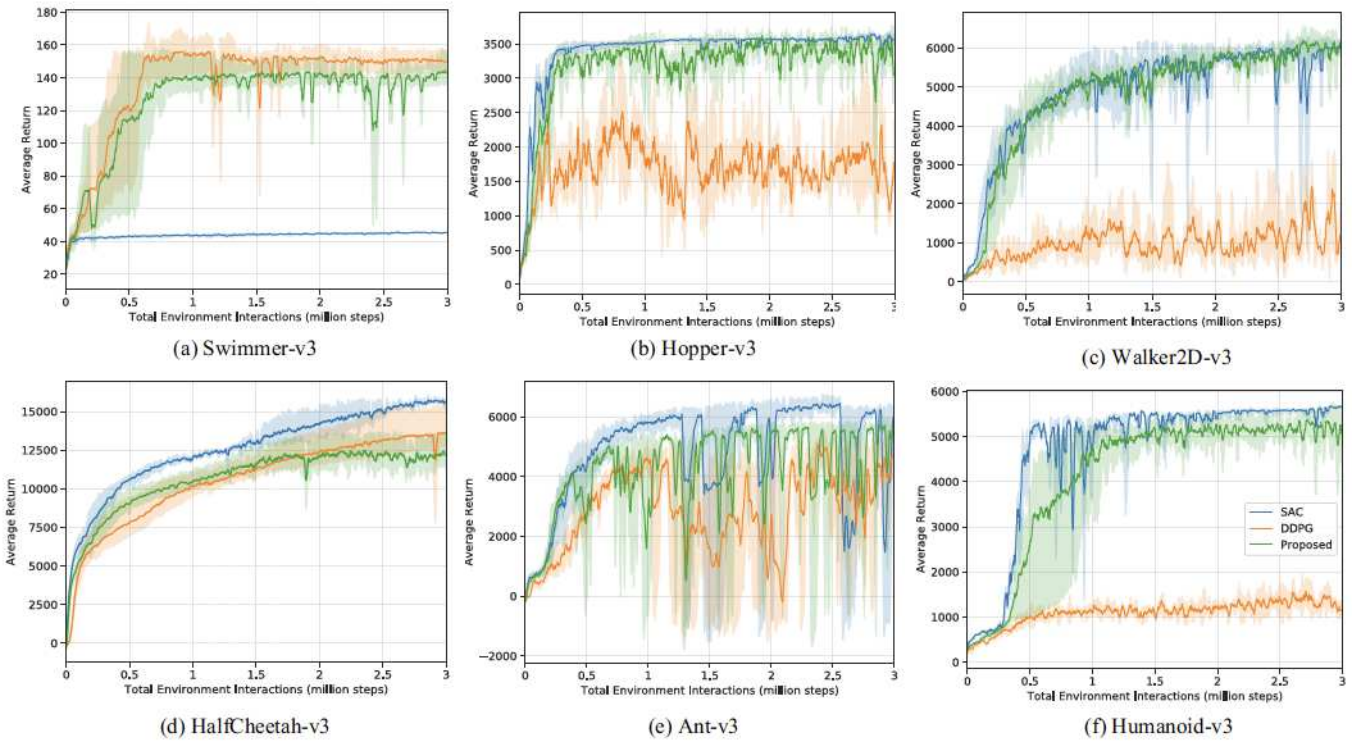
Hyperparameter	Value
optimizer	Adam [36]
learning rate λ_π	0.001
learning rate λ_Q	0.001
discount factor γ	0.99
replay buffer size $\lvert\mathcal{D}\rvert$	10^6
number of hidden layers (all networks)	2
number of units per hidden layer	256
activation function	ReLU
batch size B	256
number of action samples per state N	50
Polyak coefficient β	0.995
Sliding window size for estimating C_{π_θ}	200

five random seeds, and the shaded areas correspond to the 95% confidence intervals.

The comparison results reveal that the proposed algorithm significantly outperforms the baseline trust-region methods in terms of both the final average return and sample efficiency on all six tasks. Specifically, the proposed algorithm achieves much higher final average returns than the baseline methods on all tasks, indicating that it can learn more effective policies. Additionally, the proposed algorithm requires fewer samples to achieve comparable performance to the baseline methods, demonstrating its superior sample efficiency. One possible reason for the superior performance of the proposed algorithm is that it is derived directly from a novel theory that guarantees monotonic improvement guarantee, allowing it to discover better policies. On the easier tasks, Swimmer-v3 and Hopper-v3, TRPO learns a relatively good policy but fails to do so on the harder tasks, such as HalfCheetah, Ant-v3, and Humanoid-v3. Although Trust-PCL performs slightly better than TRPO and PPO on more high-dimensional tasks, i.e., HalfCheetah and Ant-v3, it fails to solve these tasks and can merely learn sub-optimal policies. On the most complex task, Humanoid-v3, all benchmark trust-region methods fail to make any progress, indicating that it is a particularly challenging task that requires new and innovative approaches. Overall, the proposed algorithm shows great potential for solving high-dimensional, continuous control problems, especially those that are difficult for traditional trust-region methods.

C. Comparison With Other Off-Policy methods

Figure 3 shows the learning curves our proposed algorithm and two different off-policy DRL algorithms, DDPG and SAC. The comparison indicates that our method achieves similar performance to the state-of-the-art SAC algorithm on most tasks, except for Swimmer-v3, where SAC fails to learn an effective control policy. Moreover, our method significantly outperforms DDPG on Hopper-v3, Walker2D-v3, and Humanoid-v3, and performs similarly to DDPG on other tasks. It's worth noting that the learning curve for the Ant-v3 environment has a higher variance than the others, which may be due to the



(a) Swimmer-v3 (b) Hopper-v3 (c) Walker2D-v3

(d) HalfCheetah-v3 (e) Ant-v3 (f) Humanoid-v3

Fig. 3. Learning curves of the proposed algorithm in contrast to other off-policy DRL methods on six representative MuJoCo continuous control tasks. The solid curves represent the mean performance of different algorithms over five random seeds, and the shaded areas correspond to 95% confidence intervals.

high dimensionality of the state space, making the learning process more challenging. These results demonstrate that our trust-region method achieves excellent performance in terms of both final return and sample efficiency, compared to prior off-policy algorithms.

V. Conclusions

In this work, we presented an off-policy trust-region reinforcement learning algorithm that offers a sample-efficient approach for learning deep neural network policies while retaining the stability of trust-region methods. Our empirical results on a set of MuJoCo continuous control tasks demonstrate that our proposed approach outperforms prior trust-region deep RL algorithms such as TRPO, PPO, and Trust-PCL in terms of both the final return and sample efficiency. Moreover, we found that our approach achieves comparable performance to SAC, a state-of-the-art off-policy algorithm, while outperforming DDPG on most tasks. Our findings suggest that trust-region methods can be not only effective for stabilizing policy optimization but also sample efficient in learning high-quality policies for complex continuous control tasks. Overall, our proposed approach offers a promising direction for future research in deep RL algorithms.

References

[1] 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, no. 7540, pp. 529–533.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan 2016.

[3] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *Proceedings of the 4th International Conference on Learning Representations*, San Juan, Puerto Rico, May 2-4 2016.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[5] H. v. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, 2016, pp. 2094–2100.

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

[7] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, 1999, pp. 1057–1063.

[8] T. Degris, M. White, and R. S. Sutton, "Off-policy actor-critic," in *Proceedings of the 29th International Coference on International Conference on Machine Learning*. Madison, WI, USA: Omnipress, 2012, pp. 179–186.

[9] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," in *Advances in Neural Information Processing Systems*, vol. 12. MIT Press, 1999, pp. 1008–1014.

[10] X. Zhong and H. He, "A reinforcement learning-based control approach for unknown nonlinear systems with persistent adversarial inputs," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–8.

[11] Z. Ni, N. Malla, and X. Zhong, "Towards enabling deep learning

techniques for adaptive dynamic programming," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 2828–2835.

[12] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, vol. 48, 2016, pp. 1329–1338.

[13] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," 2017. [Online]. Available: https://arxiv.org/abs/1709.06560

[14] S. Kakade, "A natural policy gradient," in *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, Vancouver, British Columbia, Canada, 2001, pp. 1531–1538.

[15] J. Schulman, S. Levine, P. Moritz, M. Jordan, and P. Abbeel, "Trust region policy optimization," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, vol. 37, Lille, France, 2015, pp. 1889–1897.

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

[17] F. Otto, P. Becker, V. A. Ngo, H. C. M. Ziesche, and G. Neumann, "Differentiable trust region layers for deep reinforcement learning," in *Proceedings of the 9th International Conference on Learning Representations*, ser. ICLR'21, May 3-7 2021.

[18] O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans, "Trust-PCL: An off-policy trust region method for continuous control," in *Proceedings of the 6th International Conference on Learning Representations*, ser. ICLR'18, Vancouver, Canada, Apr 30 - May 3 2018.

[19] S. Levine, "Reinforcement learning and control as probabilistic inference: Tutorial and review," 2018. [Online]. Available: https://arxiv.org/abs/1805.00909

[20] A. Abdolmaleki, J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. Riedmiller, "Maximum a posteriori policy optimisation," in *Proceedings of the 6th International Conference on Learning Representations*, ser. ICLR'18, Vancouver, Canada, Apr 30 - May 3 2018.

[21] W. Meng, Q. Zheng, Y. Shi, and G. Pan, "An off-policy trust region policy optimization method with monotonic improvement guarantee for deep reinforcement learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 5, pp. 2223–2235, 2022.

[22] M. Tomar, L. Shani, Y. Efroni, and M. Ghavamzadeh, "Mirror descent policy optimization," 2020. [Online]. Available: https://arxiv.org/abs/2005.09814

[23] S. Vaswani, O. Bachem, S. Totaro, R. Muller, S. Garg, M. Geist, M. C. Machado, P. S. Castro, and N. L. Roux, "A general class of surrogate functions for stable and efficient reinforcement learning," in *AISTATS*, 2022, pp. 8619–8649.

[24] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proceedings of the 31st International Conference on Machine Learning*, vol. 32, Bejing, China, 22–24 Jun 2014, pp. 387–395.

[25] B. D. Ziebart, "Modeling purposeful adaptive behavior with the principle of maximum causal entropy," Ph.D. dissertation, Pittsburgh, PA, USA, 2010, carnegie Mellon University.

[26] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *Proceedings of the 34th International Conference on Machine Learning*, vol. 70, 2017, pp. 1352–1361.

[27] 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*, ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 10–15 Jul 2018, pp. 1861–1870.

[28] L. Shi, S. Li, Q. Zheng, L. Cao, L. Yang, and G. Pan, "Maximum entropy reinforcement learning with evolution strategies," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–8.

[29] S. Kakade and J. Langford, "Approximately optimal approximate reinforcement learning," in *Proceedings of the 19th International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002, p. 267–274.

[30] H. Li, N. Clavette, and H. He, "An analytical update rule for general policy optimization," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 162. PMLR, 17–23 Jul 2022, pp. 12 696–12 716.

[31] X. Li, W. Yang, J. Liang, Z. Zhang, and M. I. Jordan, "Polyak-ruppert-averaged q-learning is statistically efficient," 2021. [Online]. Available: https://arxiv.org/abs/2112.14582

[32] 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*. IEEE, 2012, pp. 5026–5033.

[33] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[34] J. Achiam, "Spinning Up in Deep Reinforcement Learning," 2018.

[35] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," 2016. [Online]. Available: https://arxiv.org/abs/1605.08695

[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: https://arxiv.org/abs/1412.6980