SOFT DETERMINISTIC POLICY GRADIENT WITH GAUSSIAN SMOOTHING

Anonymous authors
Paper under double-blind review

ABSTRACT

Deterministic policy gradient (DPG) is widely utilized for continuous control; however, it inherently relies on the differentiability of the critic with respect to the action during policy updates. This assumption is violated in practical control problems involving sparse or discrete rewards, leading to ill-defined policy gradients and unstable learning. To address these challenges, we propose a principled alternative based on a smoothed Bellman equation formulated via Gaussian smoothing. Specifically, we define a novel action-value function based on a smoothed Bellman equation and derive the soft deterministic policy gradient (Soft-DPG). Our formulation eliminates explicit dependence on critic action-gradients and ensures that the gradient remains well-defined even for non-smooth Q-functions. We instantiate this framework into a deep reinforcement learning algorithm, which we call soft deep deterministic policy gradient (Soft DDPG). Empirical evaluations on standard continuous control benchmarks and their discretized-reward variants show that Soft DDPG remains competitive in dense-reward settings and provides clear gains in most discretized-reward environments, where standard DDPG is more sensitive to irregular critic landscapes.

1 INTRODUCTION

Reinforcement learning (RL) has been successfully applied to a wide range of domains, including robotics, autonomous driving, and control systems, by learning optimal policies through interactions with the environment (Kiran et al., 2021; Recht, 2019; Tang et al., 2025). Depending on how the optimal policy is represented and optimized, RL algorithms are commonly categorized into value-based and policy-based approaches. Value-based methods such as Deep Q-Networks (DQN) (Mnih et al., 2013) have demonstrated remarkable success in discrete action spaces (Mnih et al., 2015; Hessel et al., 2018; Van Hasselt et al., 2016), while policy-based methods, particularly policy gradient approaches, have been widely used for continuous control problems (Silver et al., 2014; Lillicrap et al., 2015; Schulman et al., 2015; 2017; Haarnoja et al., 2018). Unlike value-based methods, policy-based methods directly parameterize the policy and optimize its parameters via gradient-based methods. This direct optimization makes them particularly suitable for high-dimensional or continuous action spaces (Williams, 1992; Sutton et al., 1999). Early approaches primarily focused on stochastic policies, known as stochastic policy gradient (SPG) methods, where the policy gradient involves an expectation over the state and action distributions. However, such estimators typically suffer from high variance, which can lead to unstable and inefficient learning. To address the computational inefficiency and high variance inherent in SPG, deterministic policy gradient (DPG) was proposed by Silver et al. (2014). Unlike stochastic policies, which require expectations over both state and action spaces, deterministic policies only require expectations over the state space. This structural difference results in more efficient estimation and reduces variance, as the estimator no longer needs to integrate over a stochastic action distribution. However, deterministic policies suffer from limited exploration capability compared to stochastic policies, as the latter inherently explore the environment through stochasticity. To overcome this limitation, off-policy learning with exploratory behavior policies has been introduced (Silver et al., 2014). This approach facilitates effective exploration while learning a deterministic target policy.

Building upon these advantages, deep RL algorithms such as Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2015) and Twin Delayed DDPG (TD3) (Fujimoto et al., 2018) have been developed. These methods have demonstrated strong performance and robustness, not only

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

on standard continuous control benchmarks but also across a wide range of real-world scenarios, including autonomous driving, robotics, and complex industrial control (Qiu et al., 2019; Xu et al., 2020; Bouhamed et al., 2020; Liu & Huang, 2021; Yu et al., 2021).

Despite their success, DPG relies on a critical assumption: the action-value function must be differentiable with respect to the action. However, in many practical environments, this assumption is violated. For example, sparse or discrete reward structures are commonly used in robotics and autonomous driving (Kiran et al., 2021; Vecerik et al., 2017), leading to non-smooth or even non-differentiable Q-functions. When such non-smoothness is present, DPG may become ill-defined and can result in unstable or degraded performance in practice. To circumvent these limitations, we propose soft deterministic policy gradient (Soft-DPG), a principled framework that relaxes the stringent differentiability assumption of the critic. Our approach is based on a novel smoothed Bellman equation formulated via Gaussian smoothing (GS) (Nesterov & Spokoiny, 2017), which induces a well-behaved action-value function. By leveraging this smoothed Bellman equation, we derive a new policy gradient theorem that eliminates the explicit dependence on the critic's action-gradient. Consequently, the resulting policy gradient remains well-defined and stable even when the underlying Q-function is non-smooth.

The contributions of this work are as follows:

(a) We introduce Soft-DPG, a novel framework designed to overcome the limitations of standard DPG in environments with non-smooth action-value functions. To derive the Soft-DPG, we introduce a smoothed Bellman equation and provide a rigorous theoretical derivation of the Soft-DPG theorem.

(b) We establish analytical upper bounds for the approximation errors in both the action-value and state-value functions between the original MDP and the one induced by the smoothed Bellman equation. These bounds provide a formal guarantee on the bias introduced by the smoothing parameter.

(c) We instantiate our theoretical framework into a practical deep RL algorithm named soft deep deterministic policy gradient (Soft DDPG). Through extensive experiments on standard continuous control benchmark and their discretized reward variants, we empirically validate that Soft DDPG offers competitive stability and performance, particularly in challenging control tasks with irregular reward surfaces.

2 Related works

Deterministic policy gradient (DPG) updates the actor using the gradient of the critic with respect to the action. Unlike stochastic policy gradient (SPG), where the accuracy of the value estimate is the primary concern, DPG critically depends on the accuracy, and even the existence, of action gradients of the critic. This fundamental dependency has motivated several prior studies.

For example, Balduzzi & Ghifary (2015) argued that, in DPG, the standard critic learned via temporal-difference (TD) learning is not necessarily suitable for policy optimization. To address this, they proposed an approach that explicitly learns action gradients using an additional deviant network. While this approach addresses the mismatch between TD learning and gradient estimation, it introduces extra networks and increases the number of learnable parameters. They also considered zeroth-order gradient estimation techniques, which are conceptually related to our motivation, but at the cost of additional architectural complexity. The most closely related works to ours are those by Nachum et al. (2018); Kumar et al. (2020); Saglam & Kalogerias (2024). These works share the common idea of using smoothed value functions as alternatives to the original Q-function, without introducing additional networks and within a model-free framework. In particular, Nachum et al. (2018) introduced smoothed Q-functions in the context of stochastic policies. Their objective was not to derive DPG, but rather to enable Gaussian policy optimization by extracting gradients of both the mean and covariance from the smoothed value function. Kumar et al. (2020) and Saglam & Kalogerias (2024) employed smoothed Q-functions as surrogates for the action gradients of the original critic in DPG. By smoothing the Q-function directly, these approaches reduce sensitivity to local non-smoothness and noisy gradient estimates. Our formulation differs from these approaches in where smoothing is applied. Rather than externally smoothing a learned critic or using smoothed values only as surrogate action-gradient estimators, we incorporate Gaussian smoothing (GS) directly

into the Bellman backup. A related idea of smoothing has also been explored in Fujimoto et al. (2018); however, their approach applies smoothing at the level of the target as a regularization technique, whereas our method introduces smoothing directly into the Bellman backup, leading to a Bellman-consistent formulation.

3 PRELIMINARIES

3.1 MARKOV DECISION PROBLEM

We consider the infinite-horizon discounted Markov decision problem and Markov decision process (MDP), where the agent sequentially takes actions to maximize cumulative discounted rewards. In a MDP with the state-space $\mathcal{S} := \{1, 2, \ldots, |\mathcal{S}|\}$ and action-space $\mathcal{A} := \{1, 2, \ldots, |\mathcal{A}|\}$, the decision maker selects an action $a \in \mathcal{A}$ at the current state $s \in \mathcal{S}$, then the state transits to the next state $s' \in \mathcal{S}$ with probability $P(s'|s, a)$, and the transition incurs a reward $r(s, a, s') \in \mathbb{R}$, where $P(s'|s, a)$ is the state transition probability from the current state $s \in \mathcal{S}$ to the next state $s' \in \mathcal{S}$ under action $a \in \mathcal{A}$, and $r(s, a, s')$ is the reward function. Furthermore, we define $R(s, a) := \mathbb{E}_{s' \sim P(\cdot|s,a)} [r(s, a, s')]$ as the expected one-step reward. Moreover, $|\mathcal{S}|$ and $|\mathcal{A}|$ denote cardinalities of $\mathcal{S}$ and $\mathcal{A}$, respectively. For convenience, we consider a deterministic reward function and simply write $r(s_k, a_k, s_{k+1}) =: r_{k+1}, k \in \{0, 1, \ldots\}$. A deterministic policy, $\pi : \mathcal{S} \to \mathcal{A}$, maps a state $s \in \mathcal{S}$ to an action $\pi(s) \in \mathcal{A}$. The objective of the Markov decision problem is to find an optimal policy, π^*, such that the cumulative discounted rewards over infinite time horizons is maximized, i.e.,

$$\pi^* := \arg\max_{\pi \in \Theta} \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{k+1} \middle| \pi \right],$$

where $\gamma \in [0, 1)$ is the discount factor, Θ is the set of all deterministic policies, $(s_0, a_0, s_1, a_1, \ldots)$ is a state-action trajectory generated by the Markov chain under policy π, and $\mathbb{E}[\cdot|\pi]$ is an expectation conditioned on the policy π. Moreover, for finite state–action pairs $(s, a) \in \mathcal{S} \times \mathcal{A}$, the Q-function under policy π is defined as

$$Q^\pi(s, a) = \mathbb{E}\left[\sum_{k \geq 0} \gamma^k r_{k+1} \mid s_0 = s,\ a_0 = a,\ \pi \right].$$

and the optimal Q-function is defined as $Q^*(s, a) = Q^{\pi^*}(s, a)$ for all $(s, a) \in \mathcal{S} \times \mathcal{A}$. Once Q^* is known, then an optimal policy can be retrieved by the greedy policy $\pi^*(s) = \arg\max_{a \in \mathcal{A}} Q^*(s, a)$. Throughout, we assume that the MDP is ergodic so that the stationary state distribution exists.

3.2 DETERMINISTIC POLICY GRADIENT

DPG is a class of policy gradient algorithms that optimize deterministic policies in continuous action spaces (Silver et al., 2014). Unlike SPG, which model the policy as a probability distribution, DPG assumes a deterministic policy of the form $a = \pi_\theta(s)$ which directly maps states to actions. Accordingly, the objective of a deterministic policy can be written as

$$J(\theta) = \mathbb{E}_{s \sim \rho^{\pi_\theta}} \left[R(s, \pi_\theta(s)) \right], \tag{1}$$

where ρ^{π_θ} denotes the unnormalized discounted state visitation measure under policy π_θ, defined as $\rho^{\pi_\theta}(s') = \sum_{t=0}^{\infty} \gamma^t \Pr(s_t = s' \mid s_0 \sim p_0, \pi_\theta)$, that is, the discounted occupancy measure induced by following π_θ from the initial state distribution. In contrast to SPG, DPG does not require an expectation over the action space. As a result, the policy gradient admits a simplified form that depends only on an expectation over the state distribution. Silver et al. (2014) derived the DPG, given by

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim \rho^{\pi_\theta}} \left[\nabla_\theta \pi_\theta(s)\, \nabla_a Q^{\pi_\theta}(s, a) \big|_{a = \pi_\theta(s)} \right], \tag{2}$$

As a consequence, DPG benefits from reduced gradient variance, often resulting in improved sample efficiency in continuous control tasks. However, deterministic policy suffers from limited exploration capability, as they lack inherent stochasticity. To mitigate this issue, Silver et al. (2014) showed that DPG can be estimated in an off-policy manner using importance sampling. This allows learning a deterministic target policy π_θ from data generated by a stochastic behavior policy, thereby decoupling exploration from policy optimization and enabling more efficient exploration in continuous action spaces.

3.3 GAUSSIAN SMOOTHING

GS is a fundamental technique in zeroth-order optimization, where gradients are estimated using only function evaluations when direct differentiation is difficult or infeasible (Nesterov & Spokoiny, 2017). The core idea is to construct a smoothed surrogate of the original function by perturbing the input with Gaussian noise. This procedure yields a function that is differentiable everywhere, even when the original function is non-smooth or non-differentiable. We now formally define the Gaussian-smoothed function used throughout this paper.

Definition 3.1. (Nesterov & Spokoiny, 2017) For a given function $f : \mathbb{R}^n \to \mathbb{R}$, the Gaussian-smoothed function f_σ is defined as

$$f_\sigma(x) := \mathbb{E}_{w\sim\mathcal{N}(0,I)}[f(x+\sigma w)] = \int_{\mathbb{R}^n} f(x+\sigma w)\,\mathcal{N}(w;0,I)\,dw, \tag{3}$$

where $\sigma > 0$ is the smoothing parameter, and $\mathcal{N}(w;0,I)$ is the probability density function of the n-dimensional standard Gaussian distribution.

The Gaussian-smoothed function admits a well-defined gradient, which can be expressed without requiring access to the gradient of the original function. The following result gives a gradient representation of the Gaussian-smoothed function that does not require differentiating the original function.

Definition 3.2. (Nesterov & Spokoiny, 2017) For any function $f : \mathbb{R}^n \to \mathbb{R}$, the Gaussian-smoothed function $f_\sigma(x)$ is differentiable for all x, and its gradient is given by

$$\nabla_x f_\sigma(x) = \mathbb{E}_{w\sim\mathcal{N}(0,I)}\left[\frac{f(x+\sigma w)}{\sigma}\,w\right]. \tag{4}$$

This identity forms the basis of zeroth-order gradient estimators. Moreover, when the original function f is differentiable, the convergence of the smoothed gradient to the true gradient can be formally stated as the following lemma:

Lemma 3.3. *(Nesterov & Spokoiny, 2017) Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function. As the smoothing parameter σ approaches zero, the gradient of the Gaussian-smoothed function $f_\sigma(x)$ converges to the true gradient of the original function $f(x)$, such that:*

$$\lim_{\sigma\to 0} \nabla_x f_\sigma(x) = \nabla_x f(x). \tag{5}$$

Under mild regularity conditions, the Euclidean distance between the gradients of the original function and its smoothed counterpart can be explicitly bounded (Nesterov & Spokoiny, 2017). As a result, optimizing the smoothed function serves as a practical surrogate for optimizing the original objective. It has also been shown that the maximum of the smoothed function provides a lower bound on the maximum of the original function, often inducing a smoother and more favorable optimization landscape (Salimans et al., 2017).

3.4 FROM STOCHASTIC TO DETERMINISTIC POLICY GRADIENT VIA GAUSSIAN SMOOTHING

In this section, we show that DPG can be interpreted as a limiting case of SPG through the lens of GS. For completeness and pedagogical clarity, we provide a self-contained derivation that makes this connection explicit.

Lemma 3.4. *Let $\nu_\theta(a|s)$ be a Gaussian policy of the form $\nu_\theta(a|s) = \mathcal{N}(a;\,\pi_\theta(s),\,\sigma^2 I)$, where $a = \pi_\theta(s) + \sigma w, w \sim \mathcal{N}(0,I)$. Let $J(\nu_\theta)$ denote the expected discounted return under ν_θ. Suppose that Assumption B.1 in the Appendix holds. Recall the original stochastic policy gradient theorem:*

$$\nabla_\theta J(\nu_\theta) = \sum_s \rho^{\nu_\theta}(s) \int_a Q^{\nu_\theta}(s,a)\,\nabla_\theta \nu_\theta(a|s)\,da. \tag{6}$$

For this Gaussian policy, the stochastic policy gradient can be equivalently expressed as

$$\nabla_\theta J(\nu_\theta) = \mathbb{E}_{s\sim\rho^{\nu_\theta}, w\sim\mathcal{N}(0,I)}\left[\nabla_\theta \pi_\theta(s)\,\frac{Q^{\nu_\theta}(s,\pi_\theta(s)+\sigma w)\,w}{\sigma}\right]. \tag{7}$$

Moreover, as $\sigma \to 0$, by Lemma 3.3, we have

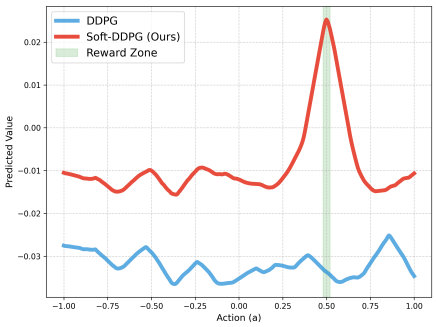
$$\lim_{\sigma \downarrow 0} \nabla_\theta J(\nu_\theta) = \mathbb{E}_{s \sim \rho^{\pi_\theta}} \left[\nabla_\theta \pi_\theta(s) \, \nabla_a Q^{\pi_\theta}(s, a) \big|_{a=\pi_\theta(s)} \right], \tag{8}$$

which recovers the deterministic policy gradient.

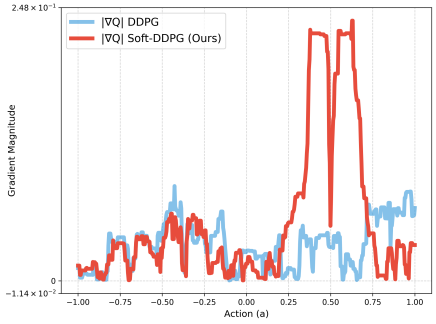
The proof is in Section B.1 of Appendix. A closely related idea already appears implicitly in Silver et al. (2014), where DPG is derived as a limiting case of SPG. By making the role of GS explicit, the above result provides an alternative derivation.

4 Action-Gradient Dependence of the Critic

DPG updates the actor by leveraging the gradient of the critic with respect to the action. Thus, the policy improvement direction is determined by $\nabla_a Q(s, a)$ evaluated at the current policy action. This formulation implicitly assumes that the action-gradient of the critic provides a reliable and stable direction for policy improvement. However, in practical implementations of DPG within the actor–critic framework, the critic is trained by minimizing a TD error objective, which enforces accuracy of the value function but does not explicitly constrain the correctness or smoothness of its action-gradient. Particularly in such discrete reward environments, standard TD learning may fail to construct an accurate Q-function landscape, rendering the subsequent action-gradients inherently unreliable. As a result, there is no guarantee that the learned $\nabla_a Q(s, a)$ accurately reflects the true policy improvement direction.



(a) $Q(s, a)$ landscape



(b) $|\nabla_a Q(s, a)|$ landscape

Figure 1: Visualization of the learned critic and its action-gradients in the toy environment with a discrete reward.

To illustrate this issue, we construct a simple toy environment designed to expose instability in the critic action-gradient. The environment consists of a single dummy state $s = [1]$ and a continuous action $a \in [-1, 1]$. The reward function is deliberately discrete: a positive reward is given only when the action lies within a narrow interval around $a = 0.5$, and zero otherwise. Formally, the reward is defined as

$$r(s, a, s') = \begin{cases} 1, & \text{if } |a - 0.5| < \epsilon, \\ 0, & \text{otherwise,} \end{cases}$$

where $\epsilon > 0$ is a small threshold. The episode lasts for a fixed number of steps to allow TD learning. To instantiate the DPG framework in practice, we employ the DDPG algorithm to learn the critic, and subsequently analyze both the learned $Q(s, a)$ and its action-gradient $\nabla_a Q(s, a)$. We visualize the learned critic landscape $Q(s, a)$ and the corresponding action-gradient magnitude $|\nabla_a Q(s, a)|$ after training in Figure 1. The results show that the critic can exhibit irregular variations across the action space, which directly translate into noisy and unstable gradients. Since DPG relies directly on these gradients for policy updates, such irregularities can degrade the stability of actor learning. In contrast, as a preview of our proposed method, we also plot the landscape of soft deep deterministic policy gradient (Soft-DDPG) which will be formally introduced in next section. Empirically, we observe that the proposed method yields a more structured critic landscape, which better captures the underlying reward region. Consequently, the induced action-gradients become more stable and informative compared to those of standard DDPG.

5 Soft Deterministic Policy Gradient

To address the limitations of DPG discussed above, we propose a smoothed variant based on GS, which we call soft deterministic policy gradient (Soft-DPG). Rather than applying smoothing

heuristically at the policy or action level, our approach incorporates GS directly into the Bellman backup operator. This modification induces a smoothed action-value function that leads to stable and well-defined policy gradients.

5.1 σ-SMOOTHED BELLMAN EQUATION

To address the instability caused by the action-gradient dependence of the critic in DPG, we introduce a σ-smoothed Bellman equation, where $\sigma > 0$ is a smoothing parameter. Our approach applies GS at the level of the Bellman backup operator. This leads to a new smoothed action-value function that induces stable and well-defined policy gradient. Formally, the smoothed action-value function Q_σ^π is defined as the solution to the following Bellman equation:

$$Q_\sigma^\pi(s,a) = R(s,a) + \gamma \int_{\mathcal{S}} V_\sigma^\pi(s')\, P(s' \mid s,a)\, ds', \tag{9}$$

where the associated smoothed value function is given by

$$V_\sigma^\pi(s) = \mathbb{E}_{w\sim\mathcal{N}(0,I)}\big[Q_\sigma^\pi(s,\, \pi(s) + \sigma w)\big]. \tag{10}$$

Substituting (10) into (9) explicitly yields the σ-smoothed Bellman equation for Q_σ^π:

$$Q_\sigma^\pi(s,a) = R(s,a) + \gamma \mathbb{E}_{s'\sim P(\cdot|s,a),\, w\sim\mathcal{N}(0,I)}\big[Q_\sigma^\pi(s',\, \pi(s') + \sigma w)\big]. \tag{11}$$

This relationship motivates the definition of the σ-smoothed Bellman expectation operator T_σ^π.

Definition 5.1. The σ-smoothed Bellman expectation operator T_σ^π is defined for any function $Q : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$ as $(T_\sigma^\pi Q)(s,a) := R(s,a) + \gamma \mathbb{E}_{s'\sim P(\cdot|s,a),\, w\sim\mathcal{N}(0,I)}\big[Q(s',\, \pi(s') + \sigma w)\big]$.

By comparing (11) and Definition 5.1, it directly follows that Q_σ^π is a fixed point of the operator T_σ^π, satisfying $Q_\sigma^\pi = T_\sigma^\pi Q_\sigma^\pi$. To guarantee the uniqueness of this fixed point, we must verify that T_σ^π is a contraction mapping. The following lemma establishes a key contraction property of the σ-smoothed Bellman operator:

Lemma 5.2. *The operator T_σ^π is a γ-contraction mapping under the supremum norm. That is, for any two functions $Q, Q' : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$,*

$$\|T_\sigma^\pi Q - T_\sigma^\pi Q'\|_\infty \le \gamma \|Q - Q'\|_\infty.$$

The proof is in Section B.2 of Appendix. Note that it is important to distinguish Q_σ^π from the GS of the original action-value function Q^π. In general, $Q_\sigma^\pi(s,a) \;\neq\; \mathbb{E}_{w\sim\mathcal{N}(0,I)}\big[Q^\pi(s,a+\sigma w)\big]$. This distinction is crucial for preserving Bellman consistency and enables a principled derivation of Soft-DPG. We next quantify the approximation error introduced by GS. First, we establish the error bound for the state-value function through the following lemma:

Lemma 5.3. *Let V^π be the value function of the original MDP and V_σ^π the value function induced by the σ-smoothed Bellman equation. Assume that the reward function $R(s,a)$ is L_R-Lipschitz continuous and the transition kernel $P(s' \mid s,a)$ is L_P-Lipschitz continuous with respect to a. Then*

$$\|V^\pi - V_\sigma^\pi\|_\infty \le \frac{\sigma\sqrt{m}}{1-\gamma}\left(L_R + \frac{\gamma}{2} L_P V_{\max}\right),$$

where m is the dimensionality of the action space, and $V_{\max} = R_{\max}/(1-\gamma)$.

The proof is in Section B.3 of Appendix. This result indicates that the bias introduced by GS on the overall value landscape is bounded and can be controlled by the smoothing parameter σ. In particular, since the error bound scales linearly with σ, the approximation error vanishes as $\sigma \to 0$. Building upon the Lipschitz continuity of the reward and transition dynamics, we next show that the smoothed action-value function is Lipschitz continuous with respect to the action in the following lemma:

Lemma 5.4. *Under the same assumptions as Lemma 5.3, for all $s \in \mathcal{S}$, the smoothed action-value function $Q_\sigma^\pi(s,\cdot)$ is L_Q-Lipschitz continuous with respect to the Euclidean norm on the action space, where the Lipschitz constant is given by $L_Q = L_R + \gamma L_P V_{\max}$.*

The proof is provided in Section B.4 of Appendix. We can now formally bound the approximation error between the original action-value function Q^π and its smoothed counterpart Q_σ^π.

Lemma 5.5. *Under the assumptions of Lemma 5.3, the approximation error between the original and smoothed action-value functions satisfies*

$$\|Q^\pi - Q^\pi_\sigma\|_\infty \;\le\; \frac{\gamma L_Q \sigma \sqrt{m}}{1-\gamma},$$

where m is the dimensionality of the action space.

The proof is in Section B.5 of Appendix. The above results show that the smoothed action-value function Q^π_σ approximates the original action-value function with a controllable bias determined by the smoothing parameter σ. Similarly, the approximation error vanishes as $\sigma \to 0$.

5.2 Theoretical formulation of Soft-DPG

In this section, we explicitly derive the Soft-DPG. Building upon the σ-smoothed Bellman equation introduced in the previous section, we show how GS at the Bellman operator level leads to a well-defined and stable policy gradient that does not rely on fragile action-gradients of the critic. We begin by deriving a gradient Bellman equation for the smoothed value function V^π_σ, which serves as a key intermediate result. The following lemma presents this result:

Lemma 5.6. *Let $V^{\pi_\theta}_\sigma$ be the value function induced by the σ-smoothed Bellman equation. Then its gradient satisfies*

$$\nabla_\theta V^{\pi_\theta}_\sigma(s) = \gamma \int_{\mathcal{S}} \nabla_\theta V^{\pi_\theta}_\sigma(s')\, P_\sigma(s' \mid s, \pi_\theta(s))\, ds' + \nabla_\theta \int_{\mathbb{R}^m} Q^{\pi_\theta}_\sigma(s, \pi_\theta(s) + \sigma w)\, \phi(w)\, dw \Big|_{\pi=\pi_\theta},$$

where $\phi(w)$ denotes the standard Gaussian density.

The proof is in Section B.6 of Appendix. Lemma 5.6 shows that the gradient of the smoothed value function is governed by a Bellman-type recursion, with the policy appearing only through Gaussian-perturbed actions. This structure plays a crucial role in eliminating the direct dependence on the action-gradient of the critic. Building on this gradient Bellman equation, we now present the main result of this section, which provides an explicit form of the Soft-DPG.

Theorem 5.7. *Let the smoothed action be $\tilde{a} = \pi_\theta(s) + \sigma w$, where $w \sim \mathcal{N}(0, I)$, and define the GS distribution $\nu_\theta(\tilde{a} \mid s) = \mathcal{N}\big(\tilde{a};\, \pi_\theta(s),\, \sigma^2 I\big)$. Then the Soft-DPG is given by*

$$\begin{aligned}
\nabla_\theta J_\sigma &= \mathbb{E}_{s\sim\rho^{\nu_\theta}, \tilde{a}\sim\nu_\theta}\left[-\frac{1}{2\sigma^2}\nabla_\theta \big\|\tilde{a} - \pi_\theta(s)\big\|_2^2\, Q^{\pi_\theta}_\sigma(s,\tilde{a})\right] \\
&= \mathbb{E}_{s\sim\rho^{\nu_\theta}, \tilde{a}\sim\nu_\theta}\left[\frac{1}{\sigma^2}\nabla_\theta \pi_\theta(s)\,(\tilde{a} - \pi_\theta(s))\, Q^{\pi_\theta}_\sigma(s,\tilde{a})\right],
\end{aligned}$$

where ρ^{ν_θ} is the discounted state visitation measure induced by ν_θ.

The proof is provided in Section B.7 of the Appendix, alongside a simpler alternative derivation based on the standard policy gradient in Section B.8. Theorem 5.7 reveals a key distinction from the classical DPG. While standard DPG requires the critic $Q(s,a)$ to be differentiable with respect to the action, as shown in (2), the proposed Soft-DPG depends only on function evaluations of the smoothed critic at perturbed actions, and is therefore well-defined even when the original action-value function is non-smooth. Moreover, unlike prior work that applies GS directly to the original Q-function (Nachum et al., 2018; Kumar et al., 2020; Saglam & Kalogerias, 2024), our approach uses the smoothed action-value function defined as the fixed point of a smoothed Bellman operator. This distinction enables a principled application of smoothing during critic learning.

5.3 Soft deep deterministic policy gradient

To demonstrate the practical applicability of the proposed Soft-DPG, we instantiate our framework within a deep RL algorithm. Algorithm 1 summarizes the proposed method. The overall structure closely follows that of standard DDPG, with two key modifications corresponding to the critic and actor updates. For the critic update, the target value is computed using a Gaussian-perturbed target action, $\tilde{a}' = \pi_{\bar{\theta}}(s') + \sigma w$, and $w \sim \mathcal{N}(0, I)$, resulting in a smoothed target value. Note that a similar target-smoothing idea was introduced in Fujimoto et al. (2018) as a heuristic regularization technique

Algorithm 1 Soft DDPG

1: Initialize Q_w, π_θ, targets $\bar{w}, \bar{\theta}$, buffer $\mathcal{D}$
2: **for** $t = 1$ to T **do**
3: $a = \pi_\theta(s) + \epsilon,\ \epsilon \sim \mathcal{N}(0, \sigma^2_{expl} I)$ and observe (r, s'), store in $\mathcal{D}$, sample B
4: Sample $w_i \sim \mathcal{N}(0, I)$ and compute target action $\tilde{a}' = \mathrm{clip}(\pi_{\bar{\theta}}(s') + \sigma w_i, a_{\min}, a_{\max})$
5: Compute target value $y_i = r + \gamma Q_{\bar{w}}(s', \tilde{a}')$
6: Update critic parameter by minimizing: $\frac{1}{2|B|N} \sum_{i=1}^{N} \sum_{(s,a,r,s') \in B} (y_i - Q_w(s,a))^2$
7: Sample $w_i \sim \mathcal{N}(0, I)$ and compute perturbed action $a_i = \mathrm{clip}(\pi_\theta(s) + \sigma w_i, a_{\min}, a_{\max})$
8: Update actor parameter by minimizing: $\frac{1}{2|B|N\sigma^2} \sum_{i=1}^{N} \sum_{(s,a,r,s') \in B} \|a_i - \pi_\theta(s)\|^2 Q_w(s, a_i)$
9: Update targets: $\bar{w} \leftarrow (1-\tau)\bar{w} + \tau w,\ \bar{\theta} \leftarrow (1-\tau)\bar{\theta} + \tau\theta$
10: **end for**

to stabilize critic learning. However, in our framework, this target smoothing arises directly from the σ-smoothed Bellman equation that defines the smoothed action-value function Q^π_σ. Consequently, the critic is trained to approximate the fixed point of the smoothed Bellman operator, ensuring consistency with the theoretical formulation. For the actor update, instead of relying on the action-gradient of the critic, the policy is updated by sampling Gaussian-perturbed actions. This update depends only on evaluations of the smoothed critic and avoids explicit differentiation of the action-value function with respect to the action, thereby mitigating instability caused by non-smooth critics.

6 Experiments

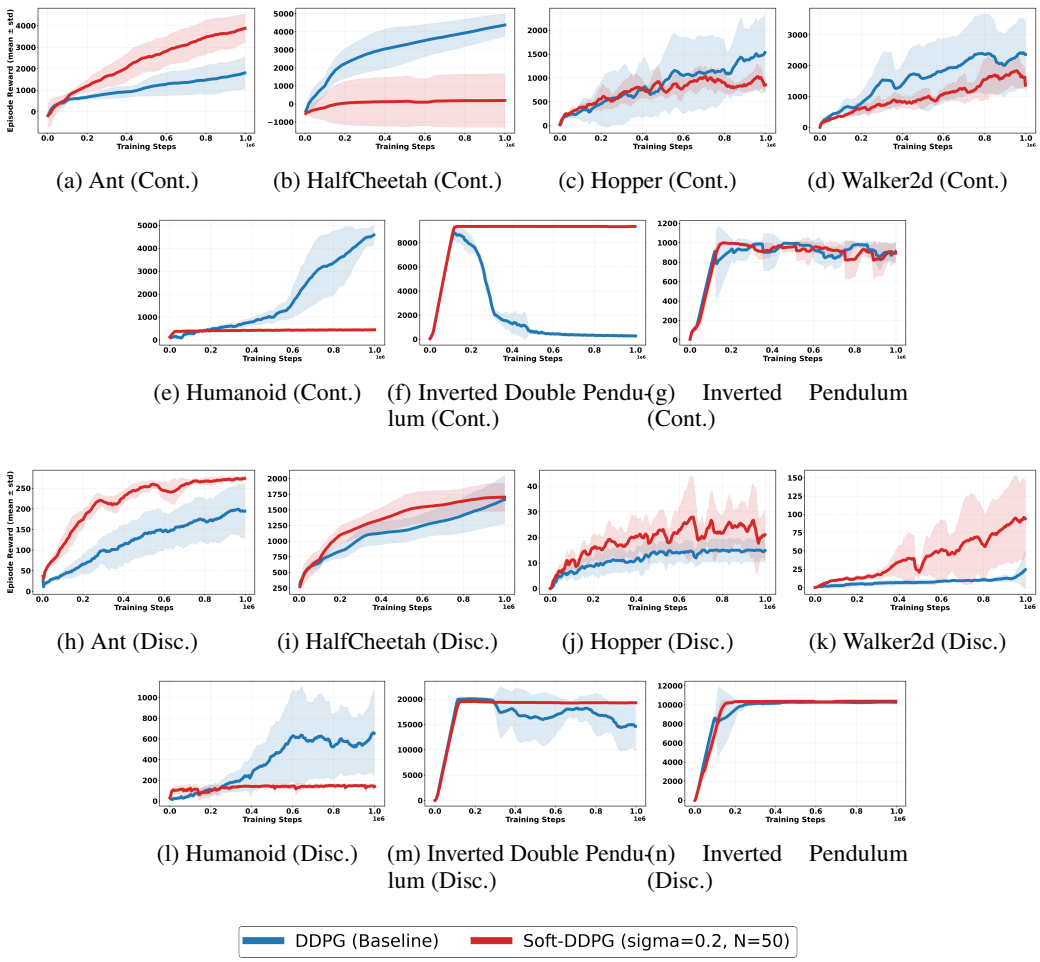
We evaluate the proposed Soft DDPG algorithm on standard MuJoCo (Todorov et al., 2012) continuous control benchmarks available in OpenAI Gym (Brockman et al., 2016) and their discrete-reward variants to assess its performance and robustness under different reward structures. Specifically, we compare Soft DDPG with the standard DDPG baseline to isolate the effect of smoothing the DPG. For Soft DDPG, the smoothing parameters, including the smoothing scale σ and the number of Monte Carlo samples N, are selected based on the sensitivity analysis detailed in Section B.9 of Appendix. All other hyperparameters are kept identical for a fair comparison (see Section B.10 of Appendix for detailed configurations). Beyond DDPG, we additionally investigate how the proposed GS principle can be incorporated into TD3. Experimental results are provided in Section D of Appendix.

Continuous reward environments: Figure 2 (top) and Table 1 present the learning curves and final performance averaged over five random seeds. In standard continuous environments where reward signals are dense, vanilla DDPG generally achieves higher performance (e.g., HalfCheetah, Hopper, Walker2d, and Humanoid). This behavior is theoretically expected, as the introduction of GS may introduce approximation bias. Nevertheless, Soft DDPG still achieves competitive results, outperforming the baseline in specific tasks such as Ant and Inverted Double Pendulum.

Discrete reward environments: The primary strength of Soft DDPG is prominently revealed in environments with non-smooth, discretized reward surfaces (see Section B.10 of the Appendix for the detailed discrete-reward settings). As shown in Figure 2 (bottom) and Table 1, Soft DDPG demonstrates consistent performance gains across most environments, including Ant, HalfCheetah, Hopper, Walker2d, Inverted Pendulum, and Inverted Double Pendulum. In these challenging settings, the standard DDPG suffers from degraded performance due to the non-differentiable nature of the discrete rewards, which induces highly unstable action-gradients. In contrast, operating on the σ-smoothed Bellman equation allows Soft DDPG to maintain well-defined policy gradients, and it leads to improved stability and higher final returns. For additional experimental results, please refer to Section C of Appendix.

7 Conclusion and Limitations

In this paper, we proposed Soft-DPG, a principled extension of DPG based on GS at the Bellman operator level. By removing the dependence on critic action-gradients, the proposed method ensures well-defined policy updates even for non-smooth Q-functions. We further instantiated this framework



(a) Ant (Cont.) (b) HalfCheetah (Cont.) (c) Hopper (Cont.) (d) Walker2d (Cont.)

(e) Humanoid (Cont.) (f) Inverted Double Pendulum (Cont.) (g) Inverted Pendulum (Cont.)

(h) Ant (Disc.) (i) HalfCheetah (Disc.) (j) Hopper (Disc.) (k) Walker2d (Disc.)

(l) Humanoid (Disc.) (m) Inverted Double Pendulum (Disc.) (n) Inverted Pendulum (Disc.)

Figure 2: The first two rows show results in the continuous-reward environments, while the last two rows show the corresponding discrete-reward variants. Solid lines denote the mean performance over multiple random seeds, and shaded regions indicate one standard deviation.

Table 1: Performance comparison (mean ± std over 5 seeds).

Environment	Discrete		Continuous	
	DDPG	Soft DDPG	DDPG	Soft DDPG
Ant	190.56 ± 60.23	$\mathbf{271.93 \pm 5.54}$	1703.54 ± 696.72	$\mathbf{3742.10 \pm 721.19}$
HalfCheetah	1599.42 ± 350.34	$\mathbf{1698.09 \pm 223.51}$	$\mathbf{4275.58 \pm 657.75}$	188.81 ± 1483.14
Hopper	14.85 ± 4.29	$\mathbf{22.29 \pm 9.26}$	$\mathbf{1243.69 \pm 567.00}$	883.59 ± 144.27
Walker2d	12.52 ± 4.13	$\mathbf{79.53 \pm 46.71}$	$\mathbf{2218.03 \pm 1036.51}$	1650.24 ± 381.54
Humanoid	$\mathbf{580.11 \pm 320.87}$	143.89 ± 14.73	$\mathbf{2496.36 \pm 695.18}$	437.65 ± 2.81
Inv. Pend.	10275.08 ± 53.76	$\mathbf{10383.10 \pm 3.35}$	$\mathbf{892.73 \pm 81.95}$	857.69 ± 132.33
Inv. Double Pend.	14678.94 ± 4693.22	$\mathbf{19284.25 \pm 58.15}$	280.40 ± 34.71	$\mathbf{9343.24 \pm 22.42}$

as Soft DDPG and demonstrated competitive performance on both standard and discretized-reward benchmarks. However, the proposed method introduces a smoothing parameter σ, which requires careful tuning. In addition, as a DDPG-based method, it inherits sensitivity to hyperparameters and lacks convergence guarantees in the deep RL setting.

AI USE STATEMENT

In this work, we did not use generative AI tools for tasks requiring mandatory disclosure under the ICLR 2027 AI Policy. We used generative AI tools to aid and polish the writing of the manuscript and for retrieval and discovery of relevant literature. We reviewed and verified all AI-assisted work. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

This work makes both theoretical and empirical contributions. The assumptions, the σ-smoothed Bellman formulation, the Soft-DPG theorem, and the Soft DDPG algorithm are stated in the main text. Supporting preliminary lemmas are provided in Sections A.1 and A.2 of Appendix, while complete proofs and additional derivations are provided in Section B of Appendix. The sensitivity analysis of the smoothing parameters σ and N is provided in Section B.9 of Appendix, and the experimental configurations, hyperparameters, and detailed constructions of the discrete-reward environments are documented in Section B.10 of Appendix. Additional experiments in discrete-reward environments are provided in Section C of Appendix, and Section D of Appendix reports the extension of the proposed Soft-DPG principle to TD3.

REFERENCES

Alekh Agarwal, Nan Jiang, Sham M Kakade, and Wen Sun. Reinforcement learning: Theory and algorithms. *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep*, 32:96, 2019.

David Balduzzi and Muhammad Ghifary. Compatible value gradients for reinforcement learning of continuous deep policies. *arXiv preprint arXiv:1509.03005*, 2015.

Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Omar Bouhamed, Hakim Ghazzai, Hichem Besbes, and Yehia Massoud. Autonomous uav navigation: A ddpg-based deep reinforcement learning approach. In *2020 IEEE International Symposium on circuits and systems (ISCAS)*, pp. 1–5. IEEE, 2020.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870, 2018.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE transactions on intelligent transportation systems*, 23(6):4909–4926, 2021.

Harshat Kumar, Dionysios S Kalogerias, George J Pappas, and Alejandro Ribeiro. Zeroth-order deterministic policy gradient. *arXiv preprint arXiv:2006.07314*, 2020.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Yen-Chen Liu and Chi-Yu Huang. Ddpg-based adaptive robust tracking control for aerial manipulators with decoupling approach. *IEEE Transactions on Cybernetics*, 52(8):8258–8271, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Ofir Nachum, Mohammad Norouzi, George Tucker, and Dale Schuurmans. Smoothed action value functions for learning gaussian policies. In *International Conference on Machine Learning*, pp. 3692–3700, 2018.

Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.

Chengrun Qiu, Yang Hu, Yan Chen, and Bing Zeng. Deep deterministic policy gradient (ddpg)-based energy harvesting wireless communications. *IEEE Internet of Things Journal*, 6(5):8577–8588, 2019.

Benjamin Recht. A tour of reinforcement learning: The view from continuous control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2(1):253–279, 2019.

Baturay Saglam and Dionysis Kalogerias. Compatible gradient approximations for actor-critic algorithms. *arXiv preprint arXiv:2409.01477*, 2024.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pp. 387–395, 2014.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Chen Tang, Ben Abbatematteo, Jiaheng Hu, Rohan Chandra, Roberto Martín-Martín, and Peter Stone. Deep reinforcement learning for robotics: A survey of real-world successes. *Annual Review of Control, Robotics, and Autonomous Systems*, 8(1):153–188, 2025.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Yi-Han Xu, Cheng-Cheng Yang, Min Hua, and Wen Zhou. Deep deterministic policy gradient (ddpg)-based resource allocation scheme for noma vehicular communications. *IEEE Access*, 8: 18797–18807, 2020.

Yu Yu, Jie Tang, Jiayi Huang, Xiuyin Zhang, Daniel Ka Chun So, and Kai-Kit Wong. Multi-objective optimization for uav-assisted wireless powered iot networks based on extended ddpg algorithm. *IEEE Transactions on Communications*, 69(9):6361–6374, 2021.

A Preliminary Lemmas

A.1 Simulation lemma

In this subsection, we state a standard simulation lemma for discounted MDPs. This lemma quantifies how errors in the reward function and transition kernel propagate to the value function under a fixed policy. It will be used in the proof of Lemma 5.3 to bound the difference between the original value function V^π and the smoothed value function V_σ^π.

Lemma A.1. *(Agarwal et al., 2019) Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$ and $\widehat{\mathcal{M}} = (\mathcal{S}, \mathcal{A}, \widehat{P}, \widehat{R}, \gamma)$ be two MDPs with the same state space, action space, and discount factor. Suppose that*

$$\max_{s\in\mathcal{S},\, a\in\mathcal{A}} |\widehat{R}(s,a) - R(s,a)| \le \epsilon_R$$

and

$$\max_{s\in\mathcal{S},\, a\in\mathcal{A}} \|\widehat{P}(\cdot \mid s,a) - P(\cdot \mid s,a)\|_1 \le \epsilon_P.$$

Then, for any policy $\pi : \mathcal{S} \to \mathcal{A}$,

$$\|V^\pi_{\widehat{\mathcal{M}}} - V^\pi_{\mathcal{M}}\|_\infty \le \frac{\epsilon_R}{1-\gamma} + \frac{\gamma\epsilon_P V_{\max}}{2(1-\gamma)},$$

where

$$V_{\max} := \frac{R_{\max}}{1-\gamma}.$$

A.2 Discounted occupancy convergence under Gaussian smoothing

In this subsection, we establish the convergence of the discounted state visitation measure induced by the Gaussian policy to that induced by the corresponding deterministic policy. In particular, we show that their discrepancy is of order $\mathcal{O}(\sigma)$ and therefore vanishes as $\sigma \downarrow 0$.

Lemma A.2. *Suppose that the transition kernel is L_P-Lipschitz continuous with respect to the action in the L_1 sense. Let $\nu_{\theta,\sigma}(\cdot \mid s) = \mathcal{N}(\pi_\theta(s), \sigma^2 I)$. Then the corresponding discounted state visitation measures satisfy*

$$\|\rho^{\nu_{\theta,\sigma}} - \rho^{\pi_\theta}\|_1 \le \frac{\gamma L_P \sqrt{m}}{(1-\gamma)^2}\sigma,$$

where m is the dimensionality of the action space. Consequently,

$$\rho^{\nu_{\theta,\sigma}} \to \rho^{\pi_\theta} \qquad \text{as } \sigma \downarrow 0.$$

Proof. Define the transition kernel induced by the Gaussian policy as

$$P_\sigma^{\pi_\theta}(\cdot \mid s) := \mathbb{E}_{w\sim\mathcal{N}(0,I)}\left[P(\cdot \mid s, \pi_\theta(s) + \sigma w)\right],$$

and let

$$P^{\pi_\theta}(\cdot \mid s) := P(\cdot \mid s, \pi_\theta(s)).$$

By the L_P-Lipschitz continuity of the transition kernel,

$$\begin{aligned}\|P_\sigma^{\pi_\theta}(\cdot \mid s) - P^{\pi_\theta}(\cdot \mid s)\|_1 &\le \mathbb{E}_w\left[\|P(\cdot \mid s, \pi_\theta(s) + \sigma w) - P(\cdot \mid s, \pi_\theta(s))\|_1\right] \\ &\le L_P\sigma\, \mathbb{E}_w[\|w\|_2] \\ &\le L_P\sigma\sqrt{m}.\end{aligned}$$

Hence,

$$\sup_{s\in\mathcal{S}} \|P_\sigma^{\pi_\theta}(\cdot \mid s) - P^{\pi_\theta}(\cdot \mid s)\|_1 \le L_P\sigma\sqrt{m}.$$

Let d_t^σ and d_t^π denote the state distributions at time t induced by $\nu_{\theta,\sigma}$ and π_θ, respectively, from the same initial distribution. Then

$$d_{t+1}^\sigma = d_t^\sigma P_\sigma^{\pi_\theta}, \qquad d_{t+1}^\pi = d_t^\pi P^{\pi_\theta}.$$

Therefore,

$$\begin{aligned}\|d_{t+1}^{\sigma} - d_{t+1}^{\pi}\|_1 &\le \|d_t^{\sigma} P_{\sigma}^{\pi_\theta} - d_t^{\sigma} P^{\pi_\theta}\|_1 + \|d_t^{\sigma} P^{\pi_\theta} - d_t^{\pi} P^{\pi_\theta}\|_1 \\ &\le L_P \sigma \sqrt{m} + \|d_t^{\sigma} - d_t^{\pi}\|_1,\end{aligned}$$

where the second term uses the contraction of a Markov kernel under the L_1 norm. Since $d_0^{\sigma} = d_0^{\pi}$, induction gives

$$\|d_t^{\sigma} - d_t^{\pi}\|_1 \le t L_P \sigma \sqrt{m}.$$

Recalling the unnormalized discounted state visitation measure,

$$\rho^{\pi} = \sum_{t=0}^{\infty} \gamma^t d_t^{\pi},$$

we obtain

$$\begin{aligned}\|\rho^{\nu_{\theta,\sigma}} - \rho^{\pi_\theta}\|_1 &\le \sum_{t=0}^{\infty} \gamma^t \|d_t^{\sigma} - d_t^{\pi}\|_1 \\ &\le L_P \sigma \sqrt{m} \sum_{t=0}^{\infty} t \gamma^t \\ &= \frac{\gamma L_P \sqrt{m}}{(1-\gamma)^2} \sigma.\end{aligned}$$

Thus,

$$\rho^{\nu_{\theta,\sigma}} \to \rho^{\pi_\theta} \qquad \text{as } \sigma \downarrow 0.$$

□

B PROOFS

B.1 PROOF OF LEMMA 3.4

Before the proof, we first explicitly state the necessary regularity conditions for the MDP and the policy.

Assumption B.1. Assume that, as $\sigma \downarrow 0$, the Gaussian policy $\nu_{\theta,\sigma}$ approaches π_θ such that

$$Q^{\nu_{\theta,\sigma}} \to Q^{\pi_\theta}.$$

We further assume that the limiting gradient is unchanged when the two Gaussian noise levels are sent to zero sequentially rather than simultaneously.

Under these assumptions, we first derive the Gaussian-policy gradient representation. The stochastic policy-gradient theorem gives

$$\nabla_\theta J(\nu_\theta) = \sum_{s \in \mathcal{S}} \rho^{\nu_\theta}(s) \int_{\mathbb{R}^m} Q^{\nu_\theta}(s,a) \nabla_\theta \nu_\theta(a \mid s)\, da.$$

For the Gaussian policy,

$$\nabla_\theta \nu_\theta(a \mid s) = \nu_\theta(a \mid s) \frac{1}{\sigma^2} \nabla_\theta \pi_\theta(s) \big(a - \pi_\theta(s)\big).$$

Substituting this identity and using $a = \pi_\theta(s) + \sigma w$ yield

$$\nabla_\theta J(\nu_\theta) = \mathbb{E}_{s \sim \rho^{\nu_\theta},\, a \sim \nu_\theta} \left[\nabla_\theta \pi_\theta(s) Q^{\nu_\theta}\big(s, \pi_\theta(s) + \sigma w\big) \frac{w}{\sigma} \right].$$

To establish the limit, we distinguish the Gaussian noise used in the policy from that used in the action perturbation. Let σ_1 denote the policy noise and σ_2 the action-perturbation noise. By Assumption B.1 and Lemma A.2, letting $\sigma_1 \downarrow 0$ gives

$$Q^{\nu_{\theta,\sigma_1}} \to Q^{\pi_\theta}, \qquad \rho^{\nu_{\theta,\sigma_1}} \to \rho^{\pi_\theta}.$$

Therefore, for any fixed $\sigma_2 > 0$,

$$\lim_{\sigma_1 \downarrow 0} \mathbb{E}_{s\sim\rho^{\nu_{\theta,\sigma_1}}, w\sim\mathcal{N}(0,I)} \left[\nabla_\theta \pi_\theta(s) Q^{\nu_{\theta,\sigma_1}}\big(s, \pi_\theta(s) + \sigma_2 w\big) \frac{w}{\sigma_2}\right] = \mathbb{E}_{s\sim\rho^{\pi_\theta}, w\sim\mathcal{N}(0,I)} \left[\nabla_\theta \pi_\theta(s) Q^{\pi_\theta}\big(s, \pi_\theta(s) + \sigma_2 w\big) \frac{w}{\sigma_2}\right].$$

Now Q^{π_θ} is fixed with respect to σ_2. Hence, applying Lemma 3.3 and letting $\sigma_2 \downarrow 0$ yields

$$\mathbb{E}_{s\sim\rho^{\pi_\theta}} \left[\nabla_\theta \pi_\theta(s) \nabla_a Q^{\pi_\theta}(s,a)\big|_{a=\pi_\theta(s)}\right].$$

By Assumption B.1 that the limiting gradient is unchanged when the two Gaussian noise levels are sent to zero sequentially rather than simultaneously, this is also the limit obtained from the original Gaussian policy as $\sigma \downarrow 0$. Therefore,

$$\lim_{\sigma\downarrow 0} \nabla_\theta J(\nu_{\theta,\sigma}) = \mathbb{E}_{s\sim\rho^{\pi_\theta}} \left[\nabla_\theta \pi_\theta(s) \nabla_a Q^{\pi_\theta}(s,a)\big|_{a=\pi_\theta(s)}\right],$$

which is precisely the deterministic policy gradient.

B.2 Proof of Lemma 5.2

Consider the difference between the two operators:

$$\begin{aligned}
&(T_\sigma^\pi Q)(s,a) - (T_\sigma^\pi Q')(s,a) \\
&\quad = (R(s,a) + \gamma\, \mathbb{E}_{s',w}[Q(s', \pi(s') + \sigma w)]) - (R(s,a) + \gamma\, \mathbb{E}_{s',w}[Q'(s', \pi(s') + \sigma w)]) \\
&\quad = \gamma\, \mathbb{E}_{s'\sim P(\cdot|s,a),\, w\sim\mathcal{N}(0,I)}[Q(s', \pi(s') + \sigma w) - Q'(s', \pi(s') + \sigma w)].
\end{aligned}$$

Taking absolute values, we obtain

$$|(T_\sigma^\pi Q)(s,a) - (T_\sigma^\pi Q')(s,a)| = \gamma\, |\mathbb{E}_{s',w}\left[Q(s', \pi(s') + \sigma w) - Q'(s', \pi(s') + \sigma w)\right]|.$$

Using Jensen's inequality (or $|\mathbb{E}[X]| \le \mathbb{E}[|X|]$),

$$|(T_\sigma^\pi Q)(s,a) - (T_\sigma^\pi Q')(s,a)| \le \gamma\, \mathbb{E}_{s',w}\left[|Q(s', \pi(s') + \sigma w) - Q'(s', \pi(s') + \sigma w)|\right].$$

For every (s', a'), the following inequality holds:

$$|Q(s',a') - Q'(s',a')| \le \|Q - Q'\|_\infty.$$

Therefore, we have

$$\begin{aligned}
|(T_\sigma^\pi Q)(s,a) - (T_\sigma^\pi Q')(s,a)| &\le \gamma\, \mathbb{E}_{s',w}\left[\|Q - Q'\|_\infty\right] \\
&= \gamma\, \|Q - Q'\|_\infty.
\end{aligned}$$

Taking the supremum over all (s,a) yields

$$\begin{aligned}
\|T_\sigma^\pi Q - T_\sigma^\pi Q'\|_\infty &= \sup_{(s,a)\in\mathcal{S}\times\mathcal{A}} |(T_\sigma^\pi Q)(s,a) - (T_\sigma^\pi Q')(s,a)| \\
&\le \gamma\, \|Q - Q'\|_\infty.
\end{aligned}$$

Hence, T_σ^π is a γ-contraction under the supremum norm.

B.3 Proof of Lemma 5.3

Let m denote the dimension of the action space, and let $w \sim \mathcal{N}(0, I)$. Recall that

$$V_\sigma^\pi(s) = \mathbb{E}_{w\sim\mathcal{N}(0,I)}\left[Q_\sigma^\pi(s, \pi(s) + \sigma w)\right].$$

By expanding Q_σ^π using the σ-smoothed Bellman equation, we obtain

$$V_\sigma^\pi(s) = \mathbb{E}_{w\sim\mathcal{N}(0,I)}\left[R(s, \pi(s) + \sigma w) + \gamma \int_{\mathcal{S}} V_\sigma^\pi(s') P(s' \mid s, \pi(s) + \sigma w)\, ds'\right].$$

Using linearity of expectation, this can be written as

$$V_\sigma^\pi(s) = \mathbb{E}_{w\sim\mathcal{N}(0,I)}\left[R(s,\pi(s)+\sigma w)\right] + \gamma \int_{\mathcal{S}} V_\sigma^\pi(s')\mathbb{E}_{w\sim\mathcal{N}(0,I)}\left[P(s' \mid s,\pi(s)+\sigma w)\right] ds'.$$

Define the smoothed reward function and transition kernel by

$$R_\sigma(s,a) := \mathbb{E}_{w\sim\mathcal{N}(0,I)}\left[R(s,a+\sigma w)\right],$$

and

$$P_\sigma(s' \mid s,a) := \mathbb{E}_{w\sim\mathcal{N}(0,I)}\left[P(s' \mid s,a+\sigma w)\right].$$

Then

$$V_\sigma^\pi(s) = R_\sigma(s,\pi(s)) + \gamma \int_{\mathcal{S}} V_\sigma^\pi(s')P_\sigma(s' \mid s,\pi(s))\, ds'.$$

Hence, V_σ^π is the value function of the modified MDP $\mathcal{M}_\sigma = (\mathcal{S},\mathcal{A},P_\sigma,R_\sigma,\gamma)$.

Applying the simulation lemma in Lemma A.1 with $\widehat{\mathcal{M}} = \mathcal{M}_\sigma$, $\widehat{R} = R_\sigma$, and $\widehat{P} = P_\sigma$, we obtain

$$\|V^\pi - V_\sigma^\pi\|_\infty \le \frac{1}{1-\gamma}\|R - R_\sigma\|_\infty + \frac{\gamma V_{\max}}{2(1-\gamma)}\|P - P_\sigma\|_{\infty,1}. \tag{12}$$

Here,

$$\|R - R_\sigma\|_\infty := \sup_{(s,a)\in\mathcal{S}\times\mathcal{A}} |R(s,a) - R_\sigma(s,a)|,$$

and

$$\|P - P_\sigma\|_{\infty,1} := \sup_{(s,a)\in\mathcal{S}\times\mathcal{A}} \int_{\mathcal{S}} |P(s' \mid s,a) - P_\sigma(s' \mid s,a)|\, ds'.$$

We now bound the two error terms. Assume that the reward function is L_R-Lipschitz continuous with respect to the Euclidean norm on the action space; that is, for all $s \in \mathcal{S}$ and $a, a' \in \mathcal{A}$,

$$|R(s,a) - R(s,a')| \le L_R\|a - a'\|_2. \tag{13}$$

Then

$$\begin{aligned}
\|R - R_\sigma\|_\infty &= \sup_{(s,a)\in\mathcal{S}\times\mathcal{A}} \left|R(s,a) - \mathbb{E}_{w\sim\mathcal{N}(0,I)}\left[R(s,a+\sigma w)\right]\right| \\
&\le \sup_{(s,a)\in\mathcal{S}\times\mathcal{A}} \mathbb{E}_{w\sim\mathcal{N}(0,I)}\left[|R(s,a) - R(s,a+\sigma w)|\right] \\
&\le \mathbb{E}_{w\sim\mathcal{N}(0,I)}\left[L_R\sigma\|w\|_2\right] \\
&= L_R\sigma\mathbb{E}_{w\sim\mathcal{N}(0,I)}\left[\|w\|_2\right] \\
&\le L_R\sigma\sqrt{m}.
\end{aligned} \tag{14}$$

The last inequality follows from Jensen's inequality and the properties of the standard multivariate normal distribution (Bishop & Nasrabadi, 2006):

$$\mathbb{E}_{w\sim\mathcal{N}(0,I)}[\|w\|_2] \le \sqrt{\mathbb{E}_{w\sim\mathcal{N}(0,I)}[\|w\|_2^2]} = \sqrt{m},$$

Next, assume that the transition kernel is L_P-Lipschitz continuous with respect to the action argument in the L_1 sense; that is, for all $s \in \mathcal{S}$ and $a, a' \in \mathcal{A}$,

$$\int_{\mathcal{S}} |P(s' \mid s,a) - P(s' \mid s,a')|\, ds' \le L_P\|a - a'\|_2. \tag{15}$$

Using this assumption,

$$\begin{aligned}
\|P - P_\sigma\|_{\infty,1} &= \sup_{(s,a)\in\mathcal{S}\times\mathcal{A}} \int_{\mathcal{S}} \left|P(s' \mid s,a) - \mathbb{E}_{w\sim\mathcal{N}(0,I)}\left[P(s' \mid s,a+\sigma w)\right]\right| ds' \\
&\le \sup_{(s,a)\in\mathcal{S}\times\mathcal{A}} \mathbb{E}_{w\sim\mathcal{N}(0,I)}\left[\int_{\mathcal{S}} |P(s' \mid s,a) - P(s' \mid s,a+\sigma w)|\, ds'\right] \\
&\le \mathbb{E}_{w\sim\mathcal{N}(0,I)}\left[L_P\sigma\|w\|_2\right] \\
&= L_P\sigma\sqrt{m}.
\end{aligned} \tag{16}$$

Substituting (14) and (16) into (12), we obtain

$$\|V^\pi - V_\sigma^\pi\|_\infty \leq \frac{L_R \sigma \sqrt{m}}{1-\gamma} + \frac{\gamma V_{\max}}{2(1-\gamma)} L_P \sigma \sqrt{m}.$$

Therefore,

$$\|V^\pi - V_\sigma^\pi\|_\infty \leq \frac{\sigma\sqrt{m}}{1-\gamma}\left(L_R + \frac{\gamma}{2} L_P V_{\max}\right).$$

This completes the proof.

B.4 Proof of Lemma 5.4

Recall the definition of the smoothed action-value function:

$$Q_\sigma^\pi(s,a) = R(s,a) + \gamma \int_{\mathcal{S}} V_\sigma^\pi(s') P(s' \mid s,a) ds'.$$

For any state $s \in \mathcal{S}$ and two actions $a, a' \in \mathcal{A}$, we can bound the difference as follows:

$$\begin{aligned} |Q_\sigma^\pi(s,a) - Q_\sigma^\pi(s,a')| \leq{}& |R(s,a) - R(s,a')| \\ &+ \gamma \int_{\mathcal{S}} V_\sigma^\pi(s') \left|P(s' \mid s,a) - P(s' \mid s,a')\right| ds'. \end{aligned}$$

By the assumptions in Lemma 5.3, the reward function is L_R-Lipschitz continuous, which gives:

$$|R(s,a) - R(s,a')| \leq L_R \|a - a'\|_2. \tag{17}$$

For the second term, we know that the value function is bounded by $V_{\max}$, i.e., $V_\sigma^\pi(s') \leq V_{\max} = R_{\max}/(1-\gamma)$. Furthermore, since the transition kernel is assumed to be L_P-Lipschitz continuous with respect to the action in the L_1 sense, we have:

$$\begin{aligned} \int_{\mathcal{S}} V_\sigma^\pi(s') \left|P(s' \mid s,a) - P(s' \mid s,a')\right| ds' &\leq V_{\max} \int_{\mathcal{S}} \left|P(s' \mid s,a) - P(s' \mid s,a')\right| ds' \\ &\leq V_{\max} L_P \|a - a'\|_2. \end{aligned} \tag{18}$$

Combining (17) and (18) yields:

$$\begin{aligned} |Q_\sigma^\pi(s,a) - Q_\sigma^\pi(s,a')| &\leq L_R \|a - a'\|_2 + \gamma V_{\max} L_P \|a - a'\|_2 \\ &= \left(L_R + \gamma L_P V_{\max}\right) \|a - a'\|_2. \end{aligned}$$

This demonstrates that for any given state s, the smoothed action-value function $Q_\sigma^\pi(s,\cdot)$ is L_Q-Lipschitz continuous with respect to the action, where the Lipschitz constant is given by $L_Q = L_R + \gamma L_P V_{\max}$. This completes the proof.

B.5 Proof of Lemma 5.5

Let m denote the dimension of the action space. Since $Q^\pi = T^\pi Q^\pi$ and $Q_\sigma^\pi = T_\sigma^\pi Q_\sigma^\pi$,

$$\|Q^\pi - Q_\sigma^\pi\|_\infty = \|T^\pi Q^\pi - T_\sigma^\pi Q_\sigma^\pi\|_\infty \leq \|T^\pi Q^\pi - T^\pi Q_\sigma^\pi\|_\infty + \|T^\pi Q_\sigma^\pi - T_\sigma^\pi Q_\sigma^\pi\|_\infty.$$

The first term is bounded by the contraction of T^π:

$$\|T^\pi Q^\pi - T^\pi Q_\sigma^\pi\|_\infty \leq \gamma \|Q^\pi - Q_\sigma^\pi\|_\infty. \tag{19}$$

For the second term, note that the reward and transition kernel are not smoothed; the only difference is in the value of the next action:

$$\begin{aligned} (T^\pi Q)(s,a) &= R(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[Q(s', \pi(s'))\right], \\ (T_\sigma^\pi Q)(s,a) &= R(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a), w \sim \mathcal{N}(0,I)} \left[Q(s', \pi(s') + \sigma w)\right]. \end{aligned}$$

By assumption, for every $s \in \mathcal{S}$ and $a, a' \in \mathcal{A}$,

$$|Q_\sigma^\pi(s,a) - Q_\sigma^\pi(s,a')| \leq L_Q \|a - a'\|_2.$$

Thus, taking $a = \pi(s')$ and $a' = \pi(s') + \sigma w$, we obtain

$$|Q_\sigma^\pi(s', \pi(s')) - Q_\sigma^\pi(s', \pi(s') + \sigma w)| \leq L_Q \|\sigma w\|_2 = L_Q \sigma \|w\|_2.$$

Therefore, for any $(s,a) \in \mathcal{S} \times \mathcal{A}$,

$$\begin{aligned}
&|(T^\pi Q_\sigma^\pi)(s,a) - (T_\sigma^\pi Q_\sigma^\pi)(s,a)| \\
&= \gamma \left| \mathbb{E}_{s' \sim P(\cdot|s,a)} \mathbb{E}_{w \sim \mathcal{N}(0,I)} \left[Q_\sigma^\pi(s', \pi(s')) - Q_\sigma^\pi(s', \pi(s') + \sigma w) \right] \right| \\
&\leq \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \mathbb{E}_{w \sim \mathcal{N}(0,I)} \left[|Q_\sigma^\pi(s', \pi(s')) - Q_\sigma^\pi(s', \pi(s') + \sigma w)| \right] \\
&\leq \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \mathbb{E}_{w \sim \mathcal{N}(0,I)} \left[L_Q \sigma \|w\|_2 \right] \\
&\leq \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \mathbb{E}_{w \sim \mathcal{N}(0,I)} \left[L_Q \sigma \|w\|_2 \right] \\
&= \gamma L_Q \sigma \mathbb{E}_{w \sim \mathcal{N}(0,I)} [\|w\|_2] \\
&\leq \gamma L_Q \sigma \sqrt{m}. \qquad (20)
\end{aligned}$$

The last inequality follows from Jensen's inequality and the properties of the standard multivariate normal distribution (Bishop & Nasrabadi, 2006): $\mathbb{E}_{w \sim \mathcal{N}(0,I)}[\|w\|_2] \leq \sqrt{\mathbb{E}_{w \sim \mathcal{N}(0,I)}[\|w\|_2^2]} = \sqrt{m}$. Since this bound holds for every (s,a), taking the supremum over (s,a) yields

$$\|T^\pi Q_\sigma^\pi - T_\sigma^\pi Q_\sigma^\pi\|_\infty \leq \gamma L_Q \sigma \sqrt{m}. \qquad (21)$$

Combining (19) and (21), we obtain

$$\|Q^\pi - Q_\sigma^\pi\|_\infty \leq \gamma \|Q^\pi - Q_\sigma^\pi\|_\infty + \gamma L_Q \sigma \sqrt{m}.$$

Rearranging gives

$$\|Q^\pi - Q_\sigma^\pi\|_\infty \leq \frac{\gamma L_Q \sigma \sqrt{m}}{1 - \gamma}.$$

This completes the proof.

B.6 Proof of Lemma 5.6

Using the chain rule, we have

$$\begin{aligned}
\nabla_\theta V_\sigma^{\pi_\theta}(s) =& \nabla_\theta E_{w \sim N(0,I)}[Q_\sigma^{\pi_\theta}(s, \pi_\theta(s) + \sigma w)] \\
=& \nabla_\theta E_{w \sim N(0,I)}[Q_\sigma^{\pi_\theta}(s, a + \sigma w)]\big|_{a = \pi_\theta(s)} + \nabla_\theta E_{w \sim N(0,I)}[Q_\sigma^{\pi}(s, \pi_\theta(s) + \sigma w)]\big|_{\pi = \pi_\theta},
\end{aligned}$$

where

$$\begin{aligned}
\nabla_\theta E_{w \sim N(0,I)}[Q_\sigma^{\pi_\theta}(s, a + \sigma w)]\big|_{a = \pi_\theta(s)} =& \nabla_\theta \left\{ R_\sigma(s,a) + \gamma \int_{s' \in S} V_\sigma^{\pi_\theta}(s') P_\sigma(s'|s,a) ds' \right\} \Big|_{a = \pi_\theta(s)} \\
=& \gamma \int_{s' \in S} \nabla_\theta V_\sigma^{\pi_\theta}(s') P_\sigma(s'|s, \pi_\theta(s)) ds'.
\end{aligned}$$

This completes the proof.

B.7 Proof of Theorem 5.7

The objective function is defined as the expected return from the initial state distribution ρ_0:

$$J_\sigma^{\pi_\theta} = \mathbb{E}_{s_0 \sim \rho_0} \left[V_\sigma^{\pi_\theta}(s_0) \right].$$

Taking the gradient with respect to θ yields:

$$\nabla_\theta J_\sigma^{\pi_\theta} = \int_{\mathcal{S}} \rho_0(s_0) \nabla_\theta V_\sigma^{\pi_\theta}(s_0) ds_0.$$

By substituting the gradient Bellman equation (Lemma 5.6) into the above expression, we obtain the first unrolling step:

$$\begin{aligned}\nabla_\theta J_\sigma^{\pi_\theta} &= \int_{\mathcal{S}} \rho_0(s_0)\Big(\nabla_\theta \mathbb{E}_{w\sim\mathcal{N}(0,I)}\big[Q_\sigma^{\pi_\theta}(s_0,\pi_\theta(s_0)+\sigma w)\big]\big|_{\pi=\pi\theta} \\ &\quad + \gamma \int_{\mathcal{S}} P_\sigma(s_1|s_0,\pi_\theta(s_0))\nabla_\theta V_\sigma^{\pi_\theta}(s_1)ds_1\Big)ds_0 \\ &= \int_{\mathcal{S}} \rho_0(s)\ \nabla_\theta \mathbb{E}_{w\sim\mathcal{N}(0,I)}\big[Q_\sigma^{\pi_\theta}(s,\pi_\theta(s)+\sigma w)\big]\big|_{\pi=\pi\theta}\, ds_0 + \gamma \int_{\mathcal{S}} \rho_1(s_1)\nabla_\theta V_\sigma^{\pi_\theta}(s_1)ds_1,\end{aligned}$$

where $\rho_1(s_1) = \int_{\mathcal{S}} \rho_0(s_0)P_\sigma(s_1|s_0,\pi_\theta(s_0))ds_0$ is the state distribution at time step 1.We can recursively substitute the gradient Bellman equation for $\nabla_\theta V_\sigma^{\pi_\theta}(s_1)$ to reveal the second unrolling step:

$$\begin{aligned}\nabla_\theta J_\sigma^{\pi_\theta} &= \int_{\mathcal{S}} \rho_0(s)\ \nabla_\theta \mathbb{E}_{w\sim\mathcal{N}(0,I)}\big[Q_\sigma^{\pi_\theta}(s,\pi_\theta(s)+\sigma w)\big]\big|_{\pi=\pi\theta}\, ds \\ &\quad + \gamma \int_{\mathcal{S}} \rho_1(s_1)\Big(\nabla_\theta \mathbb{E} w\sim\mathcal{N}(0,I)\big[Q_\sigma^{\pi_\theta}(s_1,\pi_\theta(s_1)+\sigma w)\big]\big|_{\pi=\pi\theta} \\ &\quad + \gamma \int_{\mathcal{S}} P_\sigma(s_2|s_1,\pi_\theta(s_1))\nabla_\theta V_\sigma^{\pi_\theta}(s_2)ds_2\Big)ds_1 \\ &= \int_{\mathcal{S}} \rho_0(s)\ \nabla_\theta \mathbb{E}_{w\sim\mathcal{N}(0,I)}\big[Q_\sigma^{\pi_\theta}(s,\pi_\theta(s)+\sigma w)\big]\big|_{\pi=\pi\theta}\, ds \\ &\quad + \gamma \int_{\mathcal{S}} \rho_1(s)\ \nabla_\theta \mathbb{E}_{w\sim\mathcal{N}(0,I)}\big[Q_\sigma^{\pi_\theta}(s,\pi_\theta(s)+\sigma w)\big]\big|_{\pi=\pi\theta}\, ds + \gamma^2 \int_{\mathcal{S}} \rho_2(s)\nabla_\theta V_\sigma^{\pi_\theta}(s)ds\end{aligned}$$

where $\rho_2(s) = \int_{\mathcal{S}} \rho_1(s_1)P_\sigma(s|s_1,\pi_\theta(s_1))ds_1$.

By continuing this recursive unrolling process for infinite time steps, the terms form a geometric series weighted by the state distributions at each step t:

$$\nabla_\theta J_\sigma^{\pi_\theta} = \int_{\mathcal{S}} \sum_{t=0}^{\infty} \gamma^t \rho_t(s)\ \nabla_\theta \mathbb{E}_{w\sim\mathcal{N}(0,I)}\big[Q_\sigma^{\pi_\theta}(s,\pi_\theta(s)+\sigma w)\big]\big|_{\pi=\pi_\theta}\, ds.$$

Recognizing that the unnormalized discounted state visitation measure is defined as $\rho^{\nu_\theta}(s) = \sum_{t=0}^{\infty}\gamma^t\rho_t(s)$, this simplifies compactly to an expectation over ρ^{ν_θ}:

$$\nabla_\theta J_\sigma^{\pi_\theta} = \mathbb{E}_{s\sim\rho^{\nu_\theta},\, w\sim\mathcal{N}(0,I)}\big[\nabla_\theta Q_\sigma^{\pi_\theta}(s,\pi_\theta(s)+\sigma w)\big]\big|_{\pi=\pi_\theta}.$$

From here, we proceed with the derivation of the policy gradient using the chain rule and the Gaussian log-derivative trick:

Moreover, we have

$$\begin{aligned}\nabla_\theta J_\sigma^{\pi_\theta} &= \mathbb{E}_{w\sim\mathcal{N}(0,I),\, s\sim\rho^{\nu_\theta}}\big[\nabla_\theta Q_\sigma^{\pi_\theta}(s,\pi_\theta(s)+\sigma w)\big]\big|_{\pi=\pi_\theta} \\ &= \mathbb{E}_{w\sim\mathcal{N}(0,I),\, s\sim\rho^{\nu_\theta}}\Big[\nabla_\theta\pi_\theta(s)\ \ \nabla_a Q_\sigma^{\pi_\theta}(s,a+\sigma w)|_{a=\pi_\theta(s)}\Big] \\ &= \mathbb{E}_{s\sim\rho^{\nu_\theta}}\Big[\nabla_\theta\pi_\theta(s)\ \nabla_a \mathbb{E}_{w\sim\mathcal{N}(0,I)}\big[Q_\sigma^{\pi_\theta}(s,a+\sigma w)\big]\big|_{a=\pi_\theta(s)}\Big] \\ &= \mathbb{E}_{w\sim\mathcal{N}(0,I),\, s\sim\rho^{\nu_\theta}}\left[\nabla_\theta\pi_\theta(s)\, w\, \frac{Q_\sigma^{\pi_\theta}(s,\pi_\theta(s)+\sigma w)}{\sigma}\right] \\ &= \mathbb{E}_{w\sim\mathcal{N}(0,I),\, s\sim\rho^{\nu_\theta}}\left[\nabla_\theta\pi_\theta(s)\, \frac{\tilde{a}-\pi_\theta(s)}{\sigma}\, \frac{Q_\sigma^{\pi_\theta}(s,\tilde{a})}{\sigma}\right] \\ &= \mathbb{E}_{s\sim\rho^{\nu_\theta},\, \tilde{a}\sim\nu_\theta}\left[-\frac{1}{2\sigma^2}\nabla_\theta\, \|\tilde{a}-\pi_\theta(s)\|_2^2\ Q_\sigma^{\pi_\theta}(s,\tilde{a})\right],\end{aligned}$$

where the fourth equality follows from the definition of the gradient of Gaussian smoothing. The fifth equality is obtained by applying the change of variable $\tilde{a} = \pi_\theta(s) + \sigma w$. Finally, the last equality uses the gradient identity $\nabla_\theta\|\tilde{a}-\pi_\theta(s)\|_2^2 = -2(\tilde{a}-\pi_\theta(s))\nabla_\theta\pi_\theta(s)$, with $\nu_\theta(\cdot|s) = \mathcal{N}(\pi_\theta(s),\sigma^2 I)$ denoting the Gaussian smoothing distribution. This completes the proof.

B.8 Alternative proof of Theorem 5.7 via standard policy gradient

The result in Theorem 5.7 can also be derived more directly from the standard stochastic policy gradient theorem. Define the Gaussian policy

$$\nu_\theta(\tilde{a} \mid s) = \mathcal{N}(\tilde{a}; \pi_\theta(s), \sigma^2 I).$$

The σ-smoothed Bellman equation can be interpreted as the Bellman equation induced by this Gaussian policy. In particular,

$$Q_\sigma^{\pi_\theta}(s,a) = R(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a), a' \sim \nu_\theta(\cdot|s')} \left[Q_\sigma^{\pi_\theta}(s', \tilde{a}') \right].$$

Thus, $Q_\sigma^{\pi_\theta}$ coincides with the action-value function associated with the Gaussian policy ν_θ.

Let $\rho^{\nu_\theta}(s) := \sum_{t=0}^{\infty} \gamma^t \rho_t(s)$ denote the unnormalized discounted state visitation measure induced by ν_θ, where ρ_t is the state distribution at time step t. Then, using the standard stochastic policy gradient theorem with this unnormalized measure, we obtain

$$\begin{aligned} \nabla_\theta J_\sigma &= \int_{\mathcal{S}} \rho^{\nu_\theta}(s) \int_{\mathbb{R}^m} \nabla_\theta \nu_\theta(\tilde{a} \mid s) Q_\sigma^{\pi_\theta}(s, \tilde{a}) \, d\tilde{a} \, ds \\ &= \mathbb{E}_{s \sim \rho^{\nu_\theta}, \, \tilde{a} \sim \nu_\theta} \left[\nabla_\theta \log \nu_\theta(\tilde{a} \mid s) Q_\sigma^{\pi_\theta}(s, \tilde{a}) \right]. \end{aligned}$$

For the Gaussian policy $\nu_\theta(\tilde{a} \mid s)$, the log-probability is

$$\log \nu_\theta(\tilde{a} \mid s) = -\frac{1}{2\sigma^2} \left\| \tilde{a} - \pi_\theta(s) \right\|_2^2 - \frac{m}{2} \log(2\pi\sigma^2).$$

Taking the gradient with respect to θ yields

$$\nabla_\theta \log \nu_\theta(\tilde{a} \mid s) = -\frac{1}{2\sigma^2} \nabla_\theta \left\| \tilde{a} - \pi_\theta(s) \right\|_2^2 = \frac{1}{\sigma^2} \nabla_\theta \pi_\theta(s) (\tilde{a} - \pi_\theta(s)).$$

Substituting this identity into the policy gradient expression gives

$$\begin{aligned} \nabla_\theta J_\sigma &= \mathbb{E}_{s \sim \rho^{\nu_\theta}, \, \tilde{a} \sim \nu_\theta} \left[-\frac{1}{2\sigma^2} \nabla_\theta \left\| \tilde{a} - \pi_\theta(s) \right\|_2^2 Q_\sigma^{\pi_\theta}(s, \tilde{a}) \right] \\ &= \mathbb{E}_{s \sim \rho^{\nu_\theta}, \, \tilde{a} \sim \nu_\theta} \left[\frac{1}{\sigma^2} \nabla_\theta \pi_\theta(s) (\tilde{a} - \pi_\theta(s)) Q_\sigma^{\pi_\theta}(s, \tilde{a}) \right]. \end{aligned}$$

This recovers the Soft-DPG formulation and completes the alternative proof.

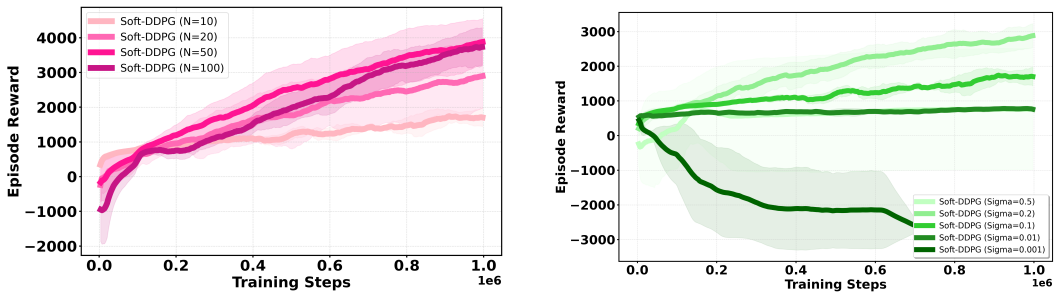
B.9 Sensitivity analysis of smoothing parameters

The proposed Soft DDPG introduces two key hyperparameters: the smoothing scale σ and the number of Monte Carlo samples N used to approximate the Gaussian smoothing expectation. To better understand the influence of these parameters, we conduct a sensitivity analysis.

We perform this analysis on the Ant environment from OpenAI Gym (Brockman et al., 2016) as a representative benchmark. While the optimal hyperparameters may vary across environments due to differences in dynamics and reward structures, our objective is not to perform exhaustive per-environment tuning, but to identify a robust configuration that generalizes well across tasks.

Effect of the number of samples N. We first evaluate the impact of the Monte Carlo sample size by fixing the smoothing scale to $\sigma = 0.2$ and varying $N \in \{10, 20, 50, 100\}$. As shown in Figure 3a, increasing N initially improves performance, as the approximation of the smoothed Bellman expectation becomes more accurate. However, beyond a certain point, the gains saturate and even slightly degrade due to increased variance and computational overhead. Empirically, we observe that $N = 50$ achieves the best overall performance, providing a favorable trade-off between approximation accuracy and efficiency. While the optimal choice of N may vary across environments, we find that $N = 50$ provides a robust and effective setting in practice. Therefore, we use $N = 50$ in all experiments.

Effect of the smoothing scale σ. Next, we analyze the influence of the smoothing scale by fixing the number of samples to $N = 10$ and varying $\sigma \in \{0.001, 0.01, 0.1, 0.2, 0.5\}$. The results in Figure 3b show that excessively small values of σ lead to a significant performance degradation. In particular, when $\sigma = 0.001$, the performance collapses, as the smoothing effect becomes negligible. On the other hand, overly large values of σ (e.g., $\sigma = 0.5$) introduce excessive bias in the smoothed value function, leading to suboptimal performance. Empirically, $\sigma = 0.2$ provides the best performance, striking a balance between stabilizing the learning dynamics and controlling approximation bias. Similarly, although the optimal σ may depend on the specific environment, $\sigma = 0.2$ consistently provides stable and effective performance across tasks. Accordingly, we use $\sigma = 0.2$ in all experiments.



(a) Sensitivity to the number of samples N ($\sigma = 0.1$) (b) Sensitivity to the smoothing scale σ ($N = 10$)

Figure 3: Sensitivity analysis of Soft DDPG hyperparameters on the Ant environment. (Left) Performance as a function of the number of samples N with $\sigma = 0.2$. (Right) Performance as a function of the smoothing scale σ with $N = 10$. The results show that $N = 50$ and $\sigma = 0.2$ provide the best trade-off between performance and stability.

B.10 DISCRETE REWARD EXPERIMENT SETTING

In this section, we provide a detailed description of the discrete reward settings used in the main experiments. Specifically, we first describe the overall experimental setup, including the hyperparameter configurations used for both DDPG and Soft DDPG. We then present the construction of discrete reward environments, where standard dense reward functions are replaced with discontinuous and quantized reward signals. These environments are designed to explicitly violate the smoothness assumptions underlying deterministic policy gradients, thereby providing a controlled setting to evaluate the robustness of the proposed method. All experiments, including those presented in the main paper and appendix, were conducted on a single workstation equipped with an Intel Core i9-10900K CPU, an NVIDIA GeForce RTX 3080 Ti GPU with 12 GB memory, and 32 GB RAM. Each experiment was run on a single GPU. We begin by summarizing the hyperparameter settings used in all experiments. Unless otherwise specified, these configurations are shared across both continuous and discrete reward environments to ensure a fair comparison.

Table 2: Hyperparameter settings for the experiments

Hyperparameter	**Value**
σ (Smoothing noise)	0.2
N (Number of smooth samples)	50
Discount factor (γ)	0.99
Batch size	256
Target update rate (τ)	0.005
Actor learning rate	10^{-4}
Critic learning rate	10^{-4}
Replay buffer size	$1,000,000$
Hidden units per layer	$400, 300$
Number of hidden layers	2
Total timesteps	$1,000,000$

Next, we describe the construction of the discrete reward functions for each environment. In all cases, the original dense reward components are removed and replaced with discontinuous, task-specific reward signals designed to induce non-smooth learning dynamics.

Ant discrete reward setup In the modified Ant environment, the original dense reward components, such as forward-progress shaping and survival incentives, are removed and replaced with a sparse milestone-based reward structure along the forward direction. The modifications are as follows:

- **Milestone-based distance rewards:** The agent receives a discrete bonus only when its horizontal position along the x-axis exceeds predefined distance thresholds for the first time. The milestone positions are set to $x \in \{0.5, 1.5, 3.0, 5.0, 10.0, 20.0\}$, and each milestone grants a reward of 50.0.
- **Sparse curriculum structure:** The milestones are placed more densely in the early stage and more sparsely at longer distances. This design provides an initial learning signal even for short forward movements, while still encouraging the agent to achieve sustained long-horizon locomotion.
- **Flip penalty:** A terminal penalty of -10.0 is applied when the episode ends due to failure, such as the ant flipping over, excluding time-limit truncation. This discourages unstable postures during exploration.
- **Survival reward removal:** The standard per-step survival reward is set to zero. This prevents the agent from exploiting a local optimum in which it simply remains alive without making meaningful forward progress.

HalfCheetah discrete reward setup In the modified HalfCheetah environment, the original dense reward composed of forward velocity and control-related shaping is replaced with a discretized velocity-based reward. The modifications are as follows:

- **Velocity-binned reward:** The agent's forward velocity v_x is partitioned into discrete intervals defined by the thresholds $\{0.0, 1.0, 3.0, 5.0\}$. The reward is assigned according to the velocity bin in which the agent currently lies.
- **Reward assignments:** The step reward is given as
  - 0.0 if $v_x \leq 0.0$,
  - 0.5 if $0.0 < v_x \leq 1.0$,
  - 1.0 if $1.0 < v_x \leq 3.0$,
  - 2.0 if $3.0 < v_x \leq 5.0$,
  - 3.0 if $v_x > 5.0$.
- **Quantized performance signal:** As the forward speed increases, the reward changes only when the velocity crosses one of the bin boundaries, thereby creating a piecewise-constant and discontinuous learning signal.
- **Dense shaping removal:** The original dense shaping terms are discarded so that the learning behavior is driven entirely by the discretized velocity levels.

Hopper discrete reward setup In the modified Hopper environment, the original dense reward is replaced with a sparse milestone-based reward defined on the agent's forward displacement. The modifications are as follows:

- **Forward milestone reward:** The agent receives a discrete reward whenever its horizontal position increases by more than a predefined threshold relative to the last rewarded position. Specifically, if the forward displacement exceeds $\Delta x = 0.5$, a reward of 5.0 is granted for each crossed milestone.
- **Accumulated milestone counting:** If the agent advances by multiple thresholds within a single step, the reward is scaled proportionally to the number of crossed milestones. This preserves the cumulative forward-progress information while maintaining a discretized signal.
- **Minimal survival reward:** A very small survival reward is provided at each non-terminal step. This weak auxiliary signal prevents the agent from immediately collapsing while ensuring that the primary learning signal still comes from forward milestone crossings.
- **Dense reward removal:** All original continuous reward components, including forward-velocity shaping and control-related terms, are discarded so that the agent is trained entirely under the proposed discrete reward structure.

Walker2d discrete reward setup In the modified Walker2d environment, the original dense reward is replaced with a sparse milestone-based forward-progress reward. The modifications are as follows:

- **Forward milestone reward:** The agent receives a discrete reward only when its horizontal displacement from the last rewarded position exceeds a threshold of $\Delta x = 0.2$. Each crossed milestone yields a reward of 2.0.
- **Accumulated milestone counting:** If multiple displacement thresholds are crossed within a single step, the reward is scaled proportionally to the number of crossed milestones. This preserves coarse forward-progress information while maintaining a discontinuous reward signal.
- **Minimal survival reward:** A very small constant reward is added at each step. This prevents immediate degeneration of behavior while ensuring that the dominant learning signal still comes from milestone-based forward progress.
- **Discontinuity emphasis:** Between milestone crossings, the forward reward remains zero, and reward is generated only when the agent passes the threshold. This creates a highly discontinuous signal and makes the task substantially harder than the standard dense-reward setting.
- **Dense reward removal:** The original dense reward components are discarded so that the agent is trained entirely under the proposed discrete locomotion reward structure.

Humanoid discrete reward setup In the modified Humanoid environment, the original dense reward is replaced with an extremely sparse and highly discretized reward function defined by posture and velocity conditions. The modifications are as follows:

- **Posture constraint:** Rewards are provided only when the humanoid maintains an upright and stable posture. Specifically, the torso height must satisfy $z > 1.18$, and the lateral velocity magnitude must remain below 0.18.
- **Low-speed reward:** If the posture constraint is satisfied and the forward velocity is smaller than 0.12, the agent receives a reward of 3.0. This provides a small discrete signal for stable but slow locomotion.
- **High-speed reward:** If the posture constraint is satisfied and the forward velocity exceeds 1.6, the agent becomes eligible for a larger reward. However, the reward of 20.0 is granted only after this high-speed condition is maintained for at least six consecutive steps.
- **Persistence requirement:** The consecutive-step counter is reset whenever the posture constraint is violated or the forward velocity leaves the high-speed regime. This makes the large reward dependent on sustained stable locomotion rather than transient bursts of speed.
- **Dense reward removal:** All original dense reward components are discarded so that the agent is trained solely under the proposed discrete reward mechanism.

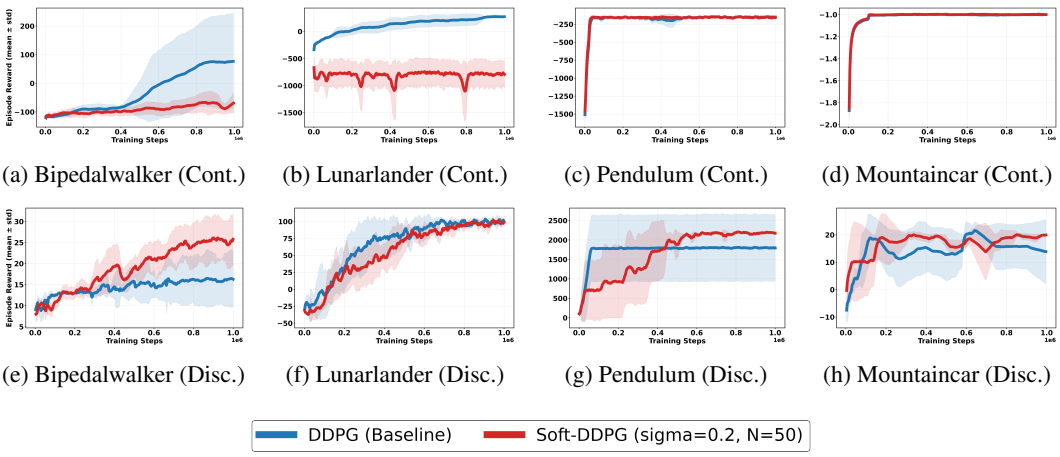
InvertedDoublePendulum discrete reward setup In the modified InvertedDoublePendulum environment, the original dense reward is replaced with a discrete reward structure based on angular stability and long-term survival. The modifications are as follows:

- **Step-wise angular reward:** Let θ_1 and θ_2 denote the two pendulum angles, and define $m = \max(|\theta_1|, |\theta_2|)$. At each step, the agent receives a discrete reward according to the angular region:
  - 20.0 if $m < 0.05$,
  - 5.0 if $0.05 \leq m < 0.15$,
  - 1.0 if $0.15 \leq m < 0.4$,
  - 0.0 otherwise.

  This creates a staircase-shaped reward landscape in which more accurate balancing yields a larger discrete reward.
- **Survival milestone bonus:** Additional sparse bonuses are given when the agent survives for a sufficiently long duration. Specifically, rewards of 500.0 are granted once when the step count reaches 500 and 1000, respectively.
- **Long-horizon credit assignment:** The survival milestones introduce large delayed rewards, making the task suitable for testing whether the algorithm can propagate long-term value information under sparse feedback.
- **Failure penalty:** If the episode terminates due to failure, excluding time-limit truncation, the reward is overwritten to -10.0. This penalizes collapse while keeping the penalty moderate enough to avoid completely suppressing exploration.
- **Dense reward removal:** All original dense reward components are discarded so that the agent is trained entirely under the proposed discrete balancing reward structure.

InvertedPendulum discrete reward setup In the modified InvertedPendulum environment, the original dense reward is replaced with a highly sparse reward structure based on angular stability and long-term survival. The modifications are as follows:

- **Angular reward:** Let θ denote the pendulum angle from the upright position. At each step, the agent receives a reward of 10.0 only when $|\theta| < 0.02$, and receives 0.0 otherwise. This creates a narrow success region in which reward is available only for highly accurate balancing.
- **Survival milestone bonus:** Additional sparse bonuses are granted when the agent survives for a sufficiently long duration. Specifically, rewards of 200.0 are given once when the step count reaches 500 and 1000, respectively.



(a) Bipedalwalker (Cont.) (b) Lunarlander (Cont.) (c) Pendulum (Cont.) (d) Mountaincar (Cont.)

(e) Bipedalwalker (Disc.) (f) Lunarlander (Disc.) (g) Pendulum (Disc.) (h) Mountaincar (Disc.)

Figure 4: Learning curves on various benchmarks: Continuous (top) vs. Discrete (bottom) control. The solid lines represent the mean and the shaded areas represent the standard deviation across multiple seeds.

- **Long-horizon credit assignment:** The survival milestones provide delayed rewards that encourage sustained stabilization, making the task suitable for testing long-term value propagation under sparse feedback.
- **Failure penalty:** If the episode terminates due to failure, excluding time-limit truncation, the reward is overwritten to -20.0. This discourages immediate collapse and distinguishes catastrophic failure from ordinary zero-reward states.
- **Dense reward removal:** All original dense reward components are discarded so that the agent is trained entirely under the proposed discrete balancing reward structure.

C Additional discrete reward experiment

To further validate the robustness of the proposed method, we conduct additional experiments on a diverse set of Gym environments (Brockman et al., 2016), including BipedalWalker, LunarLander, Pendulum, and MountainCar. These environments differ significantly in their dynamics and control objectives, providing a broader evaluation beyond the benchmarks presented in the main text.

All experiments follow the same training protocol and hyperparameter settings as described in Section B.10, ensuring consistency and fair comparison. For each environment, we construct discrete reward variants by replacing the original dense reward functions with quantized or sparse reward signals, following the same design principle used in the main experiments.

The quantitative results are summarized in Table 3. Consistent with the main experiments, Soft DDPG demonstrates clear advantages in the discrete reward settings across most environments. In particular, it outperforms the baseline in BipedalWalker, MountainCar, and Pendulum, while achieving comparable performance in LunarLander.

These results further support our main claim that Soft DDPG is particularly effective in environments with sparse or discrete reward structures, where the action-gradient of the critic becomes unreliable. By leveraging the σ-smoothed Bellman formulation, the proposed method maintains stable and informative policy updates, leading to improved performance under such challenging conditions.

In contrast, under continuous reward settings, vanilla DDPG generally achieves better performance. This behavior is expected, as the smoothing mechanism introduces bias when the underlying value function is already smooth and well-behaved.

Overall, these results demonstrate Soft DDPG provides clear gains in robustness and stability when the reward landscape is discontinuous or non-differentiable.

Table 3: Performance comparison in discrete- and continuous-reward environments (mean $\pm$ std over 5 seeds). Bold values indicate the better result within each reward setting.

Environment	Discrete		Continuous	
	DDPG	Soft DDPG	DDPG	Soft DDPG
Bipedalwalker	16.13 ± 6.22	$\mathbf{25.42 \pm 5.46}$	$\mathbf{76.95 \pm 164.00}$	-80.48 ± 23.73
Lunarlander	$\mathbf{99.03 \pm 3.13}$	98.56 ± 5.17	$\mathbf{273.11 \pm 3.77}$	-787.54 ± 252.02
Pendulum	1797.67 ± 860.55	$\mathbf{2182.19 \pm 19.22}$	$\mathbf{-149.60 \pm 2.84}$	-156.72 ± 3.28
Mountaincar	15.06 ± 9.40	$\mathbf{19.36 \pm 0.93}$	$\mathbf{-1.00 \pm 0.00}$	$\mathbf{-1.00 \pm 0.00}$

The detailed discrete reward formulations for each environment are described below.

BipedalWalker discrete reward setup In the modified BipedalWalker environment, the original dense reward is replaced with a sparse distance-based milestone reward together with a discrete terminal fall penalty. The modifications are as follows:

- **Distance milestone reward:** The agent receives a discrete reward when its forward position crosses a predefined distance threshold for the first time during an episode. The thresholds are set to $x \in \{0.5, 1.0, 5.0, 10.0\}$ with corresponding rewards $\{1.0, 2.0, 10.0, 20.0\}$.
- **One-time reward per threshold:** Each milestone reward is granted only once per episode, based on the maximum forward distance achieved so far. This ensures that the reward reflects genuine progress rather than repeated oscillations around the same position.
- **Fall penalty:** If the episode terminates due to failure rather than time-limit truncation, the agent receives a terminal penalty of -10.0. This penalizes collapse and encourages stable walking behavior.
- **Dense reward removal:** All original dense reward components, including forward-progress shaping and torque-related penalties, are discarded so that the agent is trained entirely under the proposed discrete milestone structure.

LunarLander discrete reward setup In the modified LunarLander environment, the original dense reward is replaced with a staged discrete reward structure that reflects the sequential procedure of successful landing. The modifications are as follows:

- **Approach reward:** A reward of 5.0 is granted once when the lander enters the central horizontal region of the landing pad, i.e., when $|x| < 0.2$.
- **Hover reward:** After the approach milestone has been achieved, an additional reward of 10.0 is granted once when the lander descends below a low-altitude threshold, i.e., when $y < 0.3$.
- **Success reward:** If the episode terminates with both legs in contact with the ground and the lander remains within the landing zone ($|x| < 0.2$), a terminal reward of 100.0 is assigned.
- **Crash penalty:** If the episode terminates without both legs contacting the ground, excluding time-limit truncation, a terminal penalty of -50.0 is applied.
- **One-time milestone reward:** The approach and hover rewards are granted only once per episode, preventing repeated reward collection from oscillatory motion around the same region.
- **Dense reward removal:** All original dense reward components, such as fuel-related penalties and continuous shaping terms, are discarded so that the agent is trained entirely under the proposed discrete landing reward structure.

Pendulum discrete reward setup In the modified Pendulum environment, the original dense reward is replaced with a discrete reward structure designed to encourage swing-up, near-upright stabilization, and sustained balancing. The modifications are as follows:

- **Angle-band reward:** Let θ denote the pendulum angle from the upright position. A discrete reward is assigned according to the angular deviation:

  - 8.0 if $|\theta| < 0.10$,
  - 4.0 if $0.10 \leq |\theta| < 0.25$,
  - 2.0 if $0.25 \leq |\theta| < 0.50$,
  - 0.5 if $0.50 \leq |\theta| < 1.00$,
  - 0.0 otherwise.

  This provides a staircase-shaped reward signal that guides the swing-up process toward the upright region.

- **Near-upright stabilization bonus:** When the pendulum is sufficiently close to upright ($|\theta| < 0.25$), an additional reward is given based on the angular velocity magnitude $|\dot{\theta}|$:
  - 4.0 if $|\dot{\theta}| < 0.5$,
  - 2.0 if $0.5 \leq |\dot{\theta}| < 1.0$.

  This encourages not only reaching the upright region but also reducing oscillatory motion around it.

- **Sustained hold bonus:** If the pendulum remains within a near-upright and low-velocity region defined by $|\theta| < 0.15$ and $|\dot{\theta}| < 0.7$, a hold counter is accumulated. Additional bonuses are granted when this condition is maintained for 10, 30, and 60 consecutive steps, with rewards of 20.0, 40.0, and 80.0, respectively.
- **Hierarchical discrete feedback:** The resulting reward structure combines coarse swing-up guidance, local stabilization incentives, and delayed persistence rewards, thereby forming a hierarchical discrete control objective.
- **Dense reward removal:** All original dense reward components are discarded so that the agent is trained entirely under the proposed discrete reward mechanism.

MountainCar discrete reward setup In the modified MountainCar environment, the original dense reward is replaced with a staged discrete reward structure that reflects the characteristic swing-up behavior required to reach the goal. The modifications are as follows:

- **Backward swing reward:** A reward of 10.0 is granted once when the car first reaches the backward position threshold $x < -0.7$. This encourages the agent to perform the necessary backswing motion to build momentum.
- **Forward milestone reward:** A reward of 20.0 is granted once when the car first reaches the forward position threshold $x > -0.1$. This provides an intermediate sparse signal for successful forward progress after the backswing phase.
- **Goal reward:** If the car reaches the goal position $x \geq 0.5$, an additional reward of 100.0 is assigned and the episode is terminated.
- **One-time milestone reward:** The backward-swing and forward-progress rewards are granted only once per episode, preventing repeated reward collection from oscillation around the same thresholds.
- **Per-step penalty:** A small step penalty of -0.01 is applied at every step. This weakly encourages faster completion while keeping the main learning signal dominated by the discrete milestones.
- **Dense reward removal:** All original dense reward components are discarded so that the task is learned entirely from the proposed sparse staged reward structure.

Algorithm 2 Soft TD3

Initialize $Q_{w_1}, Q_{w_2}, \pi_\theta$, targets $\bar{w}_1, \bar{w}_2, \bar{\theta}$, buffer $\mathcal{D}$
for $t = 1$ to T **do**
  Collect transition (s, a, r, s') and sample minibatch B
  Sample $w_i \sim \mathcal{N}(0, I)$ and set $\tilde{a}'_i = \mathrm{clip}(\pi_{\bar{\theta}}(s') + \sigma w_i, a_{\min}, a_{\max})$
  $y_i = r + \gamma \min_{j\in\{1,2\}} Q_{\bar{w}_j}(s', \tilde{a}'_i)$
  Update each critic Q_{w_j} by minimizing $\frac{1}{2|B|N} \sum_{i=1}^{N} \sum_B (y_i - Q_{w_j}(s, a))^2$
  if $t \bmod d = 0$ **then**
    Sample $w_i \sim \mathcal{N}(0, I)$ and set $a_i = \mathrm{clip}(\pi_\theta(s) + \sigma w_i, a_{\min}, a_{\max})$
    Update actor by minimizing $\frac{1}{2|B|N\sigma^2} \sum_{i=1}^{N} \sum_B \|a_i - \pi_\theta(s)\|^2 Q_{w_1}(s, a_i)$
    Update target networks
  end if
end for

D SOFT TD3

In this section, we investigate the application of the Soft-DPG idea to TD3. We consider two variants. The first, termed Soft-TD3, applies Gaussian smoothing to both the critic target and the actor update, following the Soft-DPG formulation. The second, termed Soft-TD3 (Actor-only), retains the standard TD3 critic update and applies the Soft-DPG update only to the actor. This variant is motivated by the fact that TD3 already incorporates target policy smoothing in its critic update, and allows us to isolate the effect of replacing the conventional action-gradient-based actor update with the Soft-DPG update. The corresponding algorithms are presented in Algorithm 2 and Algorithm 3. The experimental results are presented below.

As shown in Table 4, the Soft-TD3 (Actor-only) variant outperforms standard TD3 in most discrete-reward environments. This demonstrates that the Soft-DPG actor update can improve robustness to non-smooth action-gradients beyond the DDPG setting.

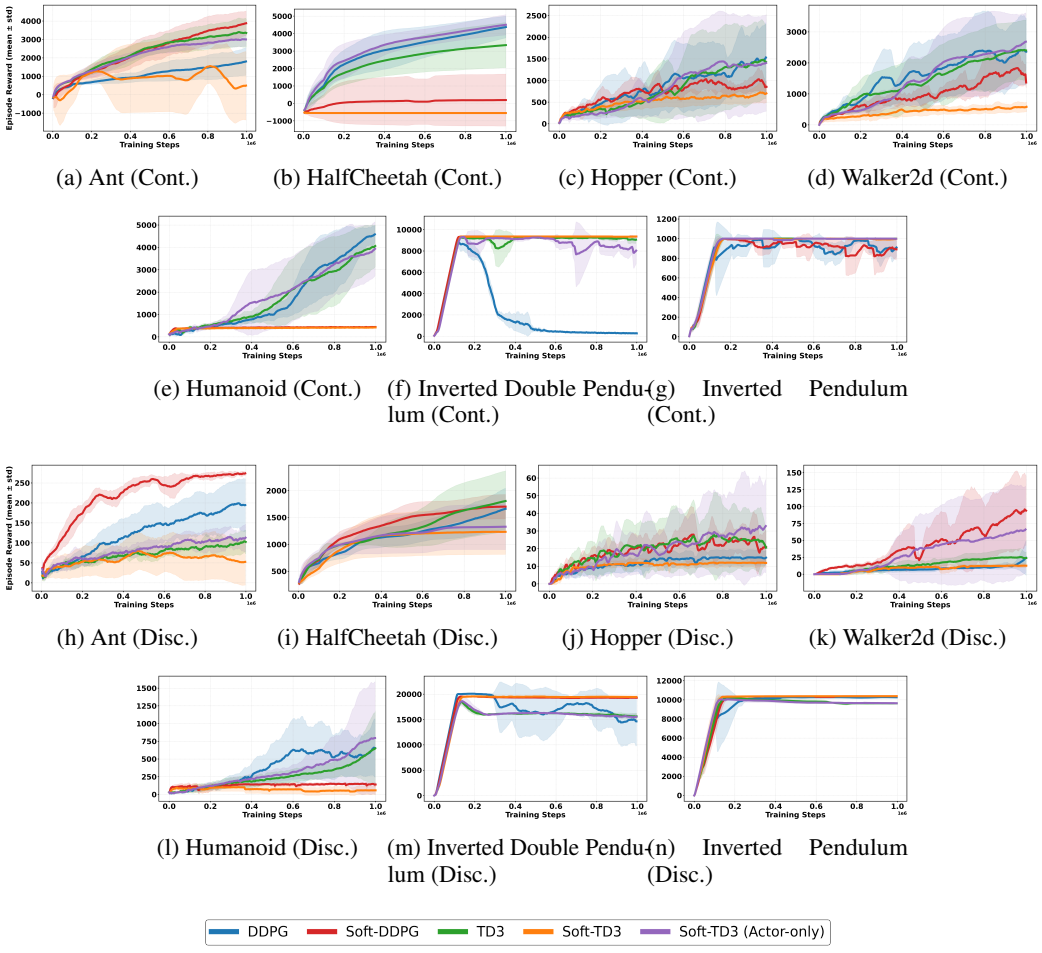
Furthermore, this actor-only approach outperforms the fully smoothed Soft-TD3 baseline in most discrete-reward environments. Because standard TD3 already regularizes the critic via target policy smoothing, applying an additional layer of Gaussian smoothing to the critic may introduce additional approximation bias and degrade value estimation.

Consequently, by restricting the Soft-DPG update to the actor while retaining the standard TD3 critic update, the resulting variant can bypass irregular critic action-gradients without introducing additional smoothing into the critic.

Table 5 provides a comprehensive performance comparison between standard deterministic actor-critic algorithms (DDPG, TD3) and their Soft variants (Soft-DDPG, Soft-TD3 (Actor-only)). Across discrete reward settings, the Soft family of algorithms achieves improved performance over their standard counterparts in many of the evaluated environments. This consistent trend across both DDPG and TD3 suggests that utilizing zeroth-order actor updates to bypass unstable action-gradients is not merely an algorithm-specific trick. Rather, these results indicate that the Soft-DPG principle can extend beyond DDPG and improve the robustness of deterministic actor-critic methods in non-smooth reward landscapes.

Algorithm 3 Soft TD3 (Actor-only)

Initialize $Q_{w_1}, Q_{w_2}, \pi_\theta$, targets $\bar{w}_1, \bar{w}_2, \bar{\theta}$, buffer $\mathcal{D}$
for $t = 1$ to T **do**
  Collect transition (s, a, r, s') and sample minibatch B
  $\epsilon \sim \text{clip}(\mathcal{N}(0, \sigma^2_{\text{TD3}} I), -c, c)$, $\tilde{a}' = \text{clip}(\pi_{\bar{\theta}}(s') + \epsilon, a_{\min}, a_{\max})$
  $y = r + \gamma \min_{j \in \{1,2\}} Q_{\bar{w}_j}(s', \tilde{a}')$
  Update Q_{w_1}, Q_{w_2} using the standard TD3 critic loss
  if $t \bmod d = 0$ **then**
    Sample $w_i \sim \mathcal{N}(0, I)$ and set $a_i = \text{clip}(\pi_\theta(s) + \sigma w_i, a_{\min}, a_{\max})$
    Update actor by minimizing $\frac{1}{2|B|N\sigma^2} \sum_{i=1}^{N} \sum_B \|a_i - \pi_\theta(s)\|^2 Q_{w_1}(s, a_i)$
    Update target networks
  end if
end for



(a) Ant (Cont.) (b) HalfCheetah (Cont.) (c) Hopper (Cont.) (d) Walker2d (Cont.)

(e) Humanoid (Cont.) (f) Inverted Double Pendulum (Cont.) (g) Inverted Pendulum (Cont.)

(h) Ant (Disc.) (i) HalfCheetah (Disc.) (j) Hopper (Disc.) (k) Walker2d (Disc.)

(l) Humanoid (Disc.) (m) Inverted Double Pendulum (Disc.) (n) Inverted Pendulum (Disc.)

Figure 5: The first two rows show results in the continuous-reward environments, while the last two rows show the corresponding discrete-reward variants. Solid lines denote the mean performance over multiple random seeds, and shaded regions indicate one standard deviation.

Table 4: Performance comparison of TD3 variants (mean ± std over 5 seeds).

Environment	TD3	Soft-TD3	Soft-TD3 (Actor-only)
Discrete Reward			
Ant	95.84 ± 24.90	57.10 ± 55.82	$\mathbf{110.38 \pm 26.96}$
HalfCheetah	$\mathbf{1763.52 \pm 550.92}$	1233.44 ± 421.05	1325.27 ± 421.94
Hopper	24.44 ± 16.65	11.95 ± 2.45	$\mathbf{29.74 \pm 25.67}$
Walker2d	23.59 ± 10.76	12.16 ± 2.74	$\mathbf{39.19 \pm 31.22}$
Humanoid	394.61 ± 129.58	51.72 ± 57.85	$\mathbf{453.90 \pm 217.79}$
Inv. Pend.	9634.46 ± 51.66	$\mathbf{10381.91 \pm 2.21}$	9657.26 ± 23.05
Inv. Double Pend.	15805.35 ± 469.98	$\mathbf{19469.32 \pm 55.02}$	15603.30 ± 402.05
Continuous Reward			
Ant	$\mathbf{3332.72 \pm 746.47}$	472.88 ± 1819.06	2985.94 ± 384.02
HalfCheetah	3293.46 ± 1270.93	-551.69 ± 3.08	$\mathbf{4426.51 \pm 560.83}$
Hopper	$\mathbf{1364.84 \pm 909.82}$	660.48 ± 187.24	1356.04 ± 1147.02
Walker2d	2256.53 ± 981.44	549.81 ± 149.20	$\mathbf{2416.49 \pm 1180.99}$
Humanoid	2845.63 ± 1233.66	415.56 ± 4.12	$\mathbf{2874.36 \pm 1258.59}$
Inv. Pend.	$\mathbf{1000.00 \pm 0.00}$	993.44 ± 6.28	999.84 ± 0.32
Inv. Double Pend.	9152.94 ± 140.04	$\mathbf{9344.84 \pm 19.62}$	8271.50 ± 648.20

Table 5: Performance comparison of standard deterministic actor–critic algorithms and their Soft-DPG variants (mean ± std over 5 seeds). Bold indicates the best-performing method within each algorithm family (DDPG or TD3).

Environment	Standard		Soft-DPG Variants		
	DDPG	TD3	Soft-DDPG	Soft-TD3	Soft-TD3 (Actor-only)
Discrete Reward					
Ant	190.56 ± 60.23	95.84 ± 24.90	$\mathbf{271.93 \pm 5.54}$	57.10 ± 55.82	$\mathbf{110.38 \pm 26.96}$
HalfCheetah	1599.42 ± 350.34	$\mathbf{1763.52 \pm 550.92}$	$\mathbf{1698.09 \pm 223.51}$	1233.44 ± 421.05	1325.27 ± 421.94
Hopper	14.85 ± 4.29	24.44 ± 16.65	$\mathbf{22.29 \pm 9.26}$	11.95 ± 2.45	$\mathbf{29.74 \pm 25.67}$
Walker2d	12.52 ± 4.13	23.59 ± 10.76	$\mathbf{79.53 \pm 46.71}$	12.16 ± 2.74	$\mathbf{39.19 \pm 31.22}$
Humanoid	$\mathbf{580.11 \pm 320.87}$	394.61 ± 129.58	143.89 ± 14.73	51.72 ± 57.85	$\mathbf{453.90 \pm 217.79}$
Inv. Pend.	10275.08 ± 53.76	9634.46 ± 51.66	$\mathbf{10383.10 \pm 3.35}$	$\mathbf{10381.91 \pm 2.21}$	9657.26 ± 23.05
Inv. Double Pend.	14678.94 ± 4693.22	15805.35 ± 469.98	$\mathbf{19284.25 \pm 58.15}$	$\mathbf{19469.32 \pm 55.02}$	15603.30 ± 402.05
Continuous Reward					
Ant	1703.54 ± 696.72	$\mathbf{3332.72 \pm 746.47}$	$\mathbf{3742.10 \pm 721.19}$	472.88 ± 1819.06	2985.94 ± 384.02
HalfCheetah	$\mathbf{4275.58 \pm 657.75}$	3293.46 ± 1270.93	188.81 ± 1483.14	-551.69 ± 3.08	$\mathbf{4426.51 \pm 560.83}$
Hopper	$\mathbf{1243.69 \pm 567.00}$	$\mathbf{1364.84 \pm 909.82}$	883.59 ± 144.27	660.48 ± 187.24	1356.04 ± 1147.02
Walker2d	$\mathbf{2218.03 \pm 1036.51}$	2256.53 ± 981.44	1650.24 ± 381.54	549.81 ± 149.20	$\mathbf{2416.49 \pm 1180.99}$
Humanoid	$\mathbf{2496.36 \pm 695.18}$	2845.63 ± 1233.66	437.65 ± 2.81	415.56 ± 4.12	$\mathbf{2874.36 \pm 1258.59}$
Inv. Pend.	$\mathbf{892.73 \pm 81.95}$	$\mathbf{1000.00 \pm 0.00}$	857.69 ± 132.33	993.44 ± 6.28	999.84 ± 0.32
Inv. Double Pend.	280.40 ± 34.71	9152.94 ± 140.04	$\mathbf{9343.24 \pm 22.42}$	$\mathbf{9344.84 \pm 19.62}$	8271.50 ± 648.20