Distribution Parameter Actor-Critic: Shifting the Agent-Environment Boundary for Diverse Action Spaces

Jiamin He, A. Rupam Mahmood[†], Martha White[†]
{jiamin12,armahmood,whitem}@ualberta.ca

Department of Computing Science, University of Alberta, Canada
Alberta Machine Intelligence Institute (Amii)
[†]Canada CIFAR AI Chair

Abstract

We introduce a novel reinforcement learning (RL) framework that treats distribution parameters as actions, redefining the boundary between agent and environment. This reparameterization makes the new action space continuous, regardless of the original action type (discrete, continuous, mixed, etc.). Under this new parameterization, we develop a generalized deterministic policy gradient estimator, *Distribution Parameter Policy Gradient* (DPPG), which has lower variance than the gradient in the original action space. Although learning the critic over distribution parameters poses new challenges, we introduce *interpolated critic learning* (ICL), a simple yet effective strategy to enhance learning, supported by insights from bandit settings. Building on TD3, a strong baseline for continuous control, we propose a practical DPPG-based actor-critic algorithm, *Distribution Parameter Actor-Critic* (DPAC). Empirically, DPAC outperforms TD3 in MuJoCo continuous control tasks from OpenAI Gym and DeepMind Control Suite, and demonstrates competitive performance on the same environments with discretized action spaces.

1 Introduction

Reinforcement learning (RL) algorithms are commonly categorized into value-based and policy-based methods. Value-based methods, such as Q-learning (Watkins & Dayan, 1992) and its variants like DQN (Mnih et al., 2015), are particularly effective in discrete action spaces due to the feasibility of enumerating and comparing action values. In contrast, policy-based methods are typically used for continuous actions, though they can be used for both discrete and continuous action spaces (Williams, 1992; Sutton et al., 1999).

Policy-based methods are typically built around the policy gradient theorem (Sutton et al., 1999), with different approaches to estimate this gradient. The likelihood-ratio (LR) estimator can be applied to arbitrary action distributions, including discrete ones. In continuous action spaces, one can alternatively compute gradients via the action-value function (the critic), leveraging its differentiability with respect to actions. This idea underlies the deterministic policy gradient (DPG) algorithms (Silver et al., 2014) and the use of the reparameterization (RP) trick for stochastic policies (Heess et al., 2015; Haarnoja et al., 2018). These approaches can produce lower-variance gradient estimates by backpropagating through the critic and the policy (Xu et al., 2019).

Despite the flexibility of policy gradient methods, current algorithms remain tightly coupled to the structure of the action space. In particular, different estimators and architectures are often required

for discrete versus continuous actions, making it difficult to design unified algorithms that generalize across domains. Although the LR estimator is always applicable, it often requires different critic architectures for different action spaces and carefully designed baselines to manage high variance, especially in continuous or high dimensional action spaces.

In this paper, we introduce the *parameter-as-action framework*, an alternative to the classical RL formulation that treats *distribution parameters* as actions. For a Gaussian policy, for example, the distribution parameters are the mean and variance, and for a softmax policy, the distribution parameters are the probability values. The RL agent outputs these distribution parameters to the environment, and the sampling of the action is now part of the stochastic transition in the environment. Distribution parameters are typically continuous, even if the actions are discrete, mixed or structured. By shifting this agent-environment boundary, therefore, we can develop one continuous-action algorithm for a diverse class of action spaces.

We first propose the *Distribution Parameter Policy Gradient* (DPPG) estimator, and prove it has lower variance than common estimators in the original action space. This reduction in variance can increase the bias, as the critic can be harder to learn. We develop an augmentation approach, called *interpolated critic learning* (ICL), to improve this critic learning. We then introduce a deep RL algorithm based on TD3 (Fujimoto et al., 2018), called *Distribution Parameter Actor-Critic* (DPAC), that incorporates the DPPG estimator and ICL. We evaluate DPAC empirically to assess the viability of this new framework. Although our goal is not surpassing the performance of existing algorithms, DPAC outperforms TD3 on 20 MuJoCo continuous control tasks from OpenAI Gym and DeepMind Control Suite, and achieves competitive performance on the same environments with discretized action spaces—without hyperparameter tuning. We also provide targeted experiments to understand the bias-variance trade-off in DPAC, and show the utility of ICL for improving critic learning.

2 Problem formulation

We consider a Markov decision process (MDP) $\langle \mathcal{S}, \mathcal{A}, p, d_0, r, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition function, $d_0 \in \Delta(\mathcal{S})$ is the initial state distribution, $r : \mathcal{S} \times \mathcal{A} \to \Delta(\mathbb{R})$ is the reward function, and γ is the discount factor. Here, $\Delta(\mathcal{X})$ denotes the set of distributions over a set $\mathcal{X}$. In this paper, we consider $\mathcal{A}$ to be either discrete or continuous.[1] We use $\pi(a|s)$ to represent the probability of taking action $a \in \mathcal{A}$ under state $s \in \mathcal{S}$ for policy π. The goal of the agent is to find a policy π under which the below objective is maximized:

$$J(\pi) \doteq \sum_{t=0}^{\infty} \mathbb{E}_{S_0 \sim d_0, A_t \sim \pi(\cdot|S_t), S_{t+1} \sim p(\cdot|S_t, A_t)} \left[\gamma^t R_{t+1}\right] = \sum_{t=0}^{\infty} \mathbb{E}_{\pi} \left[\gamma^t R_{t+1}\right], \tag{1}$$

where the second formula uses simplified notation that we follow in the rest of the paper. The *(state-)value function* and *action-value function* of the policy are defined as follows:

$$v_\pi(s) \doteq \sum_{t=0}^{\infty} \mathbb{E}_{\pi} \left[\gamma^t R_{t+1} | S_0 = s\right], \quad q_\pi(s,a) \doteq \mathbb{E}_{\pi} \left[R_1 + \gamma v_\pi(S_1) | S_0 = s, A_0 = a\right]. \tag{2}$$

In this paper, we consider actor-critic methods that learns a parameterized policy (the actor), denoted by $\pi_{\boldsymbol{\theta}}$, and a parameterized action-value function (the critic), denoted by $Q_{\mathbf{w}}$. Given a learned critic $Q_{\mathbf{w}}$, the policy is typically optimized using a surrogate of Equation (1):

$$\hat{J}(\pi_{\boldsymbol{\theta}}) = \mathbb{E}_{S_t \sim d, A_t \sim \pi_{\boldsymbol{\theta}}(\cdot|S_t)} \left[Q_{\mathbf{w}}(S_t, A_t)\right], \tag{3}$$

where $d \in \Delta(\mathcal{S})$ is some distribution over states. Given a sampled state S_t, we outline three typical stochastic estimators for the gradient of this objective below.

The likelihood-ratio (LR) policy gradient estimator uses $\hat{\nabla}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, A) = \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A|S_t) Q_{\mathbf{w}}(S_t, A)$, where $A \sim \pi_{\boldsymbol{\theta}}(\cdot|S_t)$. Since the LR estimator suffers from high variance, it is often used with the value function as a baseline:

$$\hat{\nabla}^{\text{LR}}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, A) = \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A|S_t)(Q_{\mathbf{w}}(S_t, A) - V(S_t)), \tag{4}$$

[1] Note that the framework and methods proposed in this paper also apply to other complex types of action spaces. We focus on discrete and continuous action spaces in our presentation for simplicity.

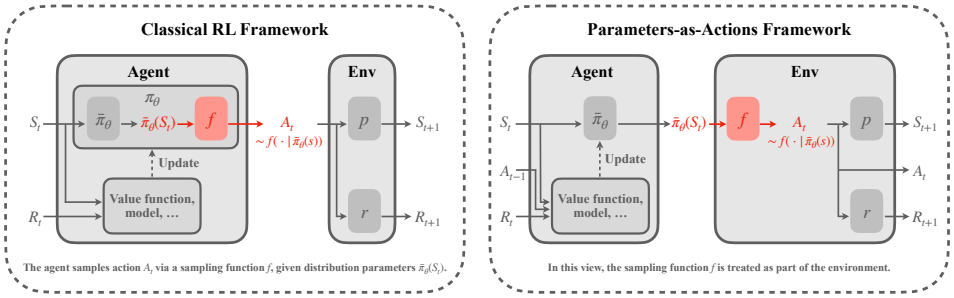

Figure 1: **Comparison between the classical reinforcement learning (RL) framework and the proposed parameters-as-actions framework.** In the classical RL setting (col 1), the agent's policy $\pi_{\boldsymbol{\theta}}$ consists of $\bar{\pi}_{\boldsymbol{\theta}}$, which produces the *distribution parameters*, and a sampling function f that returns an action given these parameters. In the parameters-as-actions framework (col 2), the sampling function f is considered part of the environment, and the agent outputs the distribution parameters $\bar{\pi}_{\boldsymbol{\theta}}(S_t)$ as its action. This shift redefines the interface between agent and environment, potentially simplifying learning and enabling new algorithmic perspectives.

where $V(S_t)$ could either be parameterized and learned or be calculated analytically from $Q_{\mathbf{w}}$ when the action space is discrete and low dimensional.

The deterministic policy gradient (DPG) estimator (Silver et al., 2014) is used when the action space is continuous and the policy is deterministic ($\pi_{\boldsymbol{\theta}} : \mathcal{S} \to \mathcal{A}$), and uses the gradient of $Q_{\mathbf{w}}$ with respect to the action:

$$\hat{\nabla}^{\text{DPG}}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t) = \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(S_t)^\top \nabla_A Q_{\mathbf{w}}(S_t, A)|_{A=\pi_{\boldsymbol{\theta}}(S_t)}. \tag{5}$$

The reparameterization (RP) policy gradient estimator (Heess et al., 2015) can be used if the policy can be reparameterized (i.e., $A = g_{\boldsymbol{\theta}}(\epsilon; S_t)$, $\epsilon \sim p(\cdot)$, where $p(\cdot)$ is a prior distribution):

$$\hat{\nabla}^{\text{RP}}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, \epsilon) = \nabla_{\boldsymbol{\theta}} g_{\boldsymbol{\theta}}(\epsilon; S_t)^\top \nabla_A Q_{\mathbf{w}}(S_t, A)|_{A=g_{\boldsymbol{\theta}}(\epsilon; S_t)}. \tag{6}$$

3 Parameters-as-actions framework

The action space is typically defined by the environment designer based on domain-specific knowledge. Depending on the problem, it may be more natural to model the action space as either discrete or continuous. In both cases, the agent's policy at a given state s can often be interpreted as first producing distribution parameters $\bar{\pi}_{\boldsymbol{\theta}}(s)$, followed by sampling an action $A \sim f(\cdot|\bar{\pi}_{\boldsymbol{\theta}}(s))$ from the resulting distribution. With a slight abuse of notation, we denote $\bar{\pi}_{\boldsymbol{\theta}} : \mathcal{S} \to \mathcal{U}$ as the part of the policy $\pi_{\boldsymbol{\theta}}$ that maps states to distribution parameters, and by $f(\cdot|u)$ the distribution over actions defined by parameters $u \in \mathcal{U}$.

In the classical RL framework, both $\bar{\pi}_{\boldsymbol{\theta}}$ and f are considered part of the agent, as in the left of Figure 1. In this work, we introduce the *parameters-as-actions framework*: the agent outputs distribution parameters $\bar{\pi}_{\boldsymbol{\theta}}(s)$ as its action, while the sampling process $A \sim f(\cdot|\bar{\pi}_{\boldsymbol{\theta}}(s))$ is treated as part of the environment, as in the right of Figure 1. This reformulation leads to a new MDP in which the action space is the parameter space $\mathcal{U}$. The reward and transition functions in this MDP become:

$$\bar{p}(s'|s,u) \doteq \mathbb{E}_{a\sim f(a|u)}[p(s'|s,a)], \quad \bar{r}(s,u) \doteq \mathbb{E}_{a\sim f(a|u)}[r(s,a)]. \tag{7}$$

This gives rise to the *parameter-space MDP* $\langle \mathcal{S}, \mathcal{U}, \bar{p}, d_0, \bar{r}, \gamma \rangle$. As in classical RL, we can define the corresponding value functions, which are connected to their classical counterparts:

$$\bar{v}_{\bar{\pi}}(s) \doteq \sum_{t=0}^{\infty} \mathbb{E}_{\bar{\pi}} \left[\gamma^t R_{t+1} | S_0 = s\right], \quad \bar{q}_{\bar{\pi}}(s,u) \doteq \mathbb{E}_{\bar{\pi}} \left[R_1 + \gamma \bar{v}_{\bar{\pi}}(S_1) | S_0 = s, U_0 = u\right]. \tag{8}$$

Assumption 3.1. The set $\mathcal{U}$ is compact. Moreover, when $\mathcal{S}$ or $\mathcal{A}$ is continuous, the corresponding set is also assumed to be compact.

Proposition 3.2. *Under Assumption 3.1,* $\bar{v}_{\bar{\pi}}(s) = v_\pi(s)$ *and* $\bar{q}_{\bar{\pi}}(s,u) = \mathbb{E}_{A\sim f(\cdot|u)}[q_\pi(s,A)]$.

The proofs of Proposition 3.2 and all other theoretical results are presented in Appendix A.

The main advantage of this framework is that it transforms the original action space into a continuous parameter space $\mathcal{U}$, regardless of whether the underlying action space $\mathcal{A}$ is discrete, continuous, or structured. This unification allows us to develop generic RL algorithms that operate over a continuous transformed action space, enabling a single framework to accommodate a wide variety of settings, including discrete-continuous hybrid action spaces (Masson et al., 2016). For example, we can apply DPG methods even in discrete action domains, where they were not previously inapplicable. We explore this direction in detail in Sections 4 and 5.

4 Distribution parameter policy gradient algorithms

In this section, we introduce the *Distribution Parameter Policy Gradient* (DPPG), a generalization of DPG for the parameters-as-actions framework. We show this estimator has lower variance, and then present a practical DPPG algorithm for deep RL based on TD3.

4.1 Distribution parameter policy gradient estimator

DPPG is the application of DPG to the parameter-space MDP. We need to slightly modify the assumptions to reason about both the distribution parameter space and the original action space.

Assumption 4.1. The functions $\bar{\pi}_{\boldsymbol{\theta}}(s)$, $f(a|u)$, and their derivatives are continuous with respect to the variables u and $\boldsymbol{\theta}$. Moreover, when $\mathcal{S}$ or $\mathcal{A}$ is continuous, the functions $p(s'|s,a)$, $d_0(s)$, $r(s,a)$, $\bar{\pi}_{\boldsymbol{\theta}}(s)$, $f(a|u)$, and their derivatives are also continuous with respect to s, s', or a, respectively.

Theorem 4.2 (Distribution parameter policy gradient theorem)**.** *Under Assumptions 3.1 and 4.1, the gradient of the objective function* $J(\bar{\pi}_{\boldsymbol{\theta}}) = \sum_{t=0}^{\infty} \mathbb{E}_{\bar{\pi}}\left[\gamma^t R_{t+1}\right]$ *with respect to* $\boldsymbol{\theta}$ *can be expressed as*

$$\nabla_{\boldsymbol{\theta}} J(\bar{\pi}_{\boldsymbol{\theta}}) = \mathbb{E}_{s\sim d_{\bar{\pi}_{\boldsymbol{\theta}}}}\left[\nabla_{\boldsymbol{\theta}} \bar{\pi}_{\boldsymbol{\theta}}(s)^\top \nabla_u \bar{q}_{\bar{\pi}_{\boldsymbol{\theta}}}(s,u)|_{u=\bar{\pi}_{\boldsymbol{\theta}}(s)}\right],$$

where $d_{\bar{\pi}_{\boldsymbol{\theta}}}(s) \doteq \sum_{t=0}^{\infty} \mathbb{E}_{\bar{\pi}_{\boldsymbol{\theta}}}[\gamma^t \mathbb{I}(S_t = s)]$ *is the (discounted) occupancy measure under* $\bar{\pi}_{\boldsymbol{\theta}}$.

The resulting gradient estimator of the surrogate objective $\hat{J}(\bar{\pi}_{\boldsymbol{\theta}}) = \mathbb{E}_{S_t\sim d}\left[\bar{Q}_{\mathbf{w}}(S_t, \bar{\pi}_{\boldsymbol{\theta}}(S_t))\right]$ is

$$\hat{\nabla}^{\text{DPPG}}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t) = \nabla_{\boldsymbol{\theta}} \bar{\pi}_{\boldsymbol{\theta}}(S_t)^\top \nabla_U \bar{Q}_{\mathbf{w}}(S_t, U)|_{U=\bar{\pi}_{\boldsymbol{\theta}}(S_t)}, \tag{9}$$

where $\bar{Q}_{\mathbf{w}}$ is a learned parameterized critic function. Note that the DPPG estimator shares the same mathematical form as the DPG estimator (see Equation (5)). However, the roles of the components differ: In DPPG, the policy outputs distribution parameters rather than a single action, and the critic estimates the expected return over the entire action distribution, rather than for a specific action. In fact, DPPG is a strict generalization of DPG. When the policy is restricted to be deterministic, the distribution parameters effectively become the action, and the parameter-space critic reduces to the classical action-value critic.

Proposition 4.3. *If* $\mathcal{U} = \mathcal{A}$ *and* $f(\cdot|u)$ *is the Dirac delta distribution centered at* u*, then* $\bar{\pi}_{\boldsymbol{\theta}}$ *and* $\bar{Q}_{\mathbf{w}}$ *are equivalent to* $\pi_{\boldsymbol{\theta}}$ *and* $Q_{\mathbf{w}}$*, respectively. Consequently, the DPPG gradient estimator becomes equivalent to DPG:* $\hat{\nabla}^{DPPG}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t) = \hat{\nabla}^{DPG}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t)$.

Moreover, DPG's theoretical analysis can also be extended to the parameter-as-action framework. In Appendix A, we generalize the convergence analysis of DPG to DPPG, establishing a theoretical guarantee that holds for MDPs with arbitrary action space types.

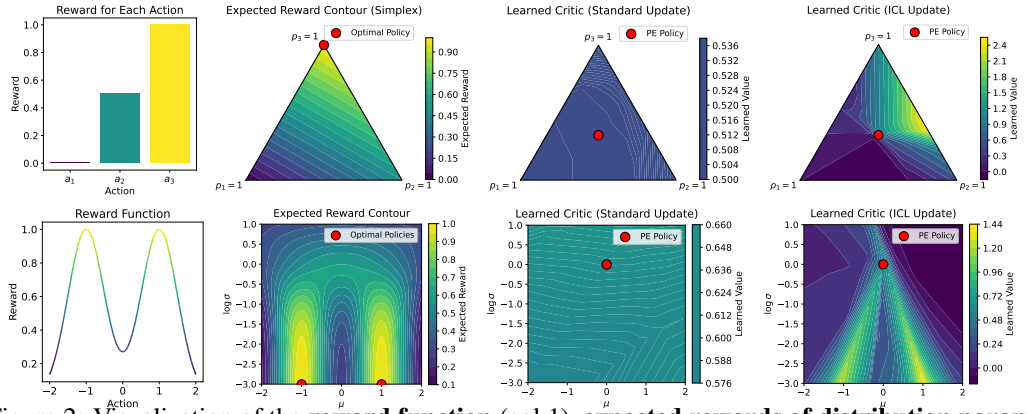

Figure 2: Visualization of the **reward function** (col 1), **expected rewards of distribution parameters** (col 2), and **learned critic functions** using the *standard* update in Equation (10) (col 3) and the *interpolated critic learning* (ICL) update in Equation (12) (col 4) in policy evaluation (PE). **Top:** K-Armed Bandit. **Bottom:** Bimodal Continuous Bandit. With access only to samples from the PE policy, the standard update estimates values accurately at the target policy but fails to generalize. In contrast, the ICL update learns a critic that captures curvature useful for policy optimization.

4.2 Comparison to other estimators for stochastic policies

We now compare the proposed DPPG estimator with classical stochastic policy gradient (PG) methods, highlighting its variance and bias characteristics across action spaces.

DPPG can be seen as the conditional expectation of both the LR (see Equation (4)) and RP (see Equation (6)) estimators. This leads to strictly lower variance.

Proposition 4.4. *Assume* $Q_{\mathbf{w}} = q_{\pi_{\boldsymbol{\theta}}}$ *in* $\hat{\nabla}^{LR}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, A)$ *and* $\bar{Q}_{\mathbf{w}} = \bar{q}_{\bar{\pi}_{\boldsymbol{\theta}}}$ *in* $\hat{\nabla}^{DPPG}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t)$. *Then,* $\hat{\nabla}^{DPPG}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t) = \mathbb{E}_{A\sim\pi_{\boldsymbol{\theta}}(\cdot|S_t)}[\hat{\nabla}^{LR}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, A)]$. *Further, if the expectation of the action-conditioned variance is greater than zero, then* $\mathbb{V}(\hat{\nabla}^{DPPG}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t)) < \mathbb{V}(\hat{\nabla}^{LR}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, A))$.

Proposition 4.5. *Assume* $\mathcal{A}$ *is continuous,* $Q_{\mathbf{w}} = q_{\pi_{\boldsymbol{\theta}}}$ *in* $\hat{\nabla}^{RP}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, \epsilon)$, *and* $\bar{Q}_{\mathbf{w}} = \bar{q}_{\bar{\pi}_{\boldsymbol{\theta}}}$ *in* $\hat{\nabla}^{DPPG}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t)$. *Then,* $\hat{\nabla}^{DPPG}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t) = \mathbb{E}_{\epsilon\sim p}[\hat{\nabla}^{RP}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, \epsilon)]$. *Further, if the expectation of the noise-induced variance is greater than zero, then* $\mathbb{V}(\hat{\nabla}^{DPPG}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t)) < \mathbb{V}(\hat{\nabla}^{RP}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, \epsilon))$.

Another direction to reduce variance is *expected policy gradient* (EPG; Ciosek & Whiteson, 2018; Allen et al., 2017). The idea is to integrate (or sum) over actions, yielding zero-variance gradients conditioned on a state: $\hat{\nabla}^{\text{EPG}}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{A_t\sim\pi_{\boldsymbol{\theta}}(\cdot|S_t)} [Q_{\mathbf{w}}(S_t, A_t)]$. However, this estimator is only practical in low-dimensional discrete action spaces (Allen et al., 2017) or in special cases within continuous settings—such as Gaussian policies with quadratic critics (Ciosek & Whiteson, 2020). In contrast, our estimator $\hat{\nabla}^{\text{DPPG}}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t)$ generalizes to a wider range of settings, including high-dimensional discrete, general continuous, and even hybrid action spaces.

Despite its lower variance, DPPG may suffer from increased bias due to the increased complexity of the critic's input space. For discrete actions, the critic $\bar{Q}_{\mathbf{w}}$ inputs a vector of probabilities corresponding to discrete outcomes. For continuous actions, with Gaussian policies, the critic $\bar{Q}_{\mathbf{w}}$ inputs both the mean and standard deviation. This increased input dimensionality makes it harder to approximate the true value function, and if the critic is inaccurate, the overall benefit of lower gradient variance may be diminished—an effect we examine empirically in Section 5.2.

4.3 Interpolated parameter-space critic learning

In this section, we propose a method to improve learning the parameter-space critic $\bar{Q}_{\mathbf{w}}$. Given a transition $\langle S_t, U_t, A_t, R_{t+1}, S_{t+1}\rangle$, the standard TD update for $\bar{Q}_{\mathbf{w}}$ is

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha(R_{t+1} + \gamma \bar{Q}_{\mathbf{w}}(S_{t+1}, \bar{\pi}_{\boldsymbol{\theta}}(S_{t+1})) - \bar{Q}_{\mathbf{w}}(S_t, U_t))\nabla \bar{Q}_{\mathbf{w}}(S_t, U_t). \tag{10}$$

This update, however, does not make use of the sampled action A_t, and its relationship to the outcome state and reward. One direction to leverage this knowledge is to recognize that the transition can also be used to update the value at alternative parameters $\hat{U}_t$. This is possible because the action A_t could have been sampled from distributions parameterized by many other $\hat{U}_t$. As a result, the value at $\hat{U}_t$ can be learned off-policy.

What, then, should we choose for $\hat{U}_t$? To answer this, we ask: *what properties should the critic have to support effective policy optimization in parameter space?* Our answer is that the critic should provide informative gradient directions that guide the policy toward optimality. For MDPs, there always exists a deterministic optimal policy (Puterman, 2014). Therefore, we assume the existence of some $U_{A_t^*} \in \mathcal{U}$, a deterministic distribution corresponding to the optimal action A_t^* for state S_t. Ideally, the critic should exhibit curvature that points toward such optimal parameters U_t^*.

One candidate for $\hat{U}_t$ is U_{A_t}, the deterministic distribution parameters associated with the sampled action A_t. However, merely learning accurate values at U_{A_t} does not ensure that the critic has smooth curvature from U_t toward these potentially high-value points. To encourage the critic to provide smoother gradients, we propose using a linearly interpolated point between U_t and U_{A_t}:

$$\hat{U}_t = \omega_t U_t + (1 - \omega_t) U_{A_t}, \quad \omega_t \sim \text{Uniform}[0, 1]. \tag{11}$$

The critic is then trained to predict the value at $\hat{U}_t$ using the following update:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha(R_{t+1} + \gamma \bar{Q}_{\mathbf{w}}(S_{t+1}, \bar{\pi}_{\boldsymbol{\theta}}(S_{t+1})) - \bar{Q}_{\mathbf{w}}(S_t, \hat{U}_t)) \nabla \bar{Q}_{\mathbf{w}}(S_t, \hat{U}_t). \tag{12}$$

We refer to this approach as *interpolated critic learning* (ICL). To further provide intuition on ICL, we conduct a policy evaluation experiment in bandit problems, shown in Figure 2 (column 1). Figure 2 (column 3) and (column 4) show the learned critic functions using the standard update in Equation (10) and the ICL update in Equation (12), respectively. The critic learned by ICL has more informative curvature. In the continuous setting, the learned critic is sufficient to identify the optimal distribution parameters. More experiment details are in Appendix B.2.

4.4 Distribution parameter actor-critic

Since the DPPG estimator is derived from DPG, we base our practical algorithm on TD3 (Fujimoto et al., 2018), a strong DPG-based off-policy actor-critic algorithm for continuous control. We replace the classical actor and critic with their parameter-space counterparts and use the DPPG gradient estimator (Equation (9)) and the ICL critic loss (Equation (12)) to update them, respectively. We omit the actor target network, as it does not improve performance (see Appendix B.4). The pseudocode for the algorithm, which we call *Distribution Parameter Actor-Critic* (DPAC), is in Appendix D.

5 Experiments

In this section, we empirically investigate DPAC in both continuous and discrete action settings.

5.1 Continuous and discrete control on common benchmarks

We use OpenAI Gym MuJoCo (Brockman et al., 2016) and the DeepMind Control (DMC) Suite (Tunyasuvunakool et al., 2020) for our experiments. From MuJoCo, we use the most commonly used 5 environments; from DMC, we use the same 15 environments as D'Oro et al. (2023). Details about them are in Appendix B.4. While these *continuous control* environments are defined with continuous action spaces, we also discretize the action spaces to test DPAC's performance for *discrete control*. Specifically, we discretize each action dimension into 7 bins with uniform spacing. For example, the original action space in Humanoid-v4 is $[-0.4, 0.4]^{17}$, which is discretized to $0.4 \times \{-1, -\frac{2}{3}, -\frac{1}{3}, 0, \frac{1}{3}, \frac{2}{3}, 1\}^{17}$. We run each environment for $1M$ steps and 10 seeds.

Baselines for continuous control We use TD3 (Fujimoto et al., 2018) as our primary baseline, as DPAC is based on it. We also include an off-policy actor-critic baseline that uses the reparameterization (RP) estimator. This AC-RP algorithm closely resembles DPAC but learns in the original

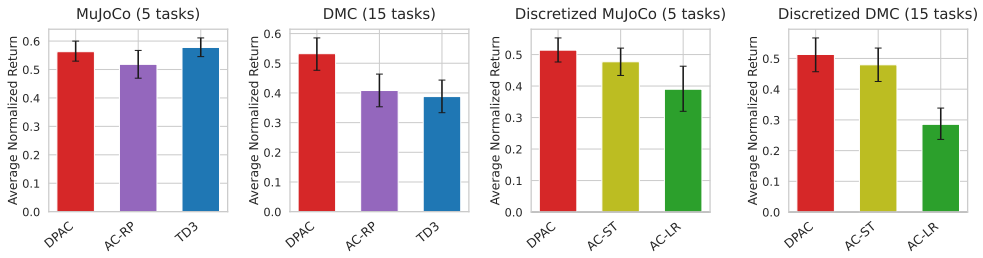

Figure 3: **Average normalized returns of DPAC and baselines** on 5 MuJoCo tasks (col 1), 15 DMC tasks (col 2), and their discretized variants (col 3–4). Values are averaged over 10 seeds and 5 (MuJoCo) or 15 (DMC) tasks. Error bars show 95% bootstrapped CIs. DPAC demonstrates robust and competitive performance in both benchmarks under both settings.

action space and updates the policy using the RP estimator. For consistency, all algorithms use the default hyperparameters of TD3 and a Gaussian policy parameterization.

Baselines for discrete control Since the likelihood-ratio (LR) estimator is the most common for discrete actions, we include an off-policy actor-critic baseline, AC-LR, similar to AC-RP but with the LR estimator. We learn a separate value function for the baseline, because analytically computing it from the action-value function is prohibitive for these high-dimensional action spaces. Additionally, although not commonly used in prior work, we include a variant that replaces the LR estimator in AC-LR with the straight-through (ST) estimator (Bengio et al., 2013), denoted as AC-ST. We use the same hyperparameters as the TD3 defaults and a categorical policy parameterization.

More details and pseudocode of these algorithms can be found in Appendices B.5 and D.

Results Figure 3 shows the average normalized returns of DPAC and the baselines in both continuous and discrete settings. For the continuous case (columns 1–2), DPAC achieves better overall performance, outperforming AC-RP and TD3 significantly in the DMC Suite. For the discrete setting (columns 3–4), DPAC's average performance is higher than AC-ST, despite overlapping confidence intervals, while AC-LR exhibits poor performance across both benchmarks. Learning curves in individual environments are in Appendix C.

5.2 Effectiveness of interpolated critic learning

We investigate the impact of using ICL (Equation (12)). We compare DPAC and DPAC w/o ICL, an ablated version that uses the standard critic update (Equation (10)). In Figure 5, we can see that DPAC w/o ICL is generally worse than DPAC for both continuous and discrete control.

To provide further insights into ICL's effectiveness, we move to a bandit setting where visualization and analysis are intuitive. In addition to DPAC and DPAC w/o ICL, we also include AC-LR and AC-RP as a reference, as they should be quite effective in these settings because of a much simpler critic function. Note that our goal is not to show that DPAC can outperform other baselines in these toy settings, but rather to illustrate how ICL substantially improves critic learning in DPAC. We also include PG-LR, PG-RP, and DPPG, variants of AC-LR, AC-RP, and DPAC that have access to their corresponding true value functions to remove the confounding factor of learning the critic. We use the same bandits from Figure 2. See Appendix B.3 for hyperparameters and other details.

As shown in Figure 6, DPPG exhibits slightly faster convergence than PG-LR and PG-RP—albeit marginal—highlighting the advantage of using a lower-variance estimator when no bias is present. However, when the critic must be learned, DPAC w/o ICL performs significantly worse than both AC-LR and AC-RP, achieving highly suboptimal returns even by the end of training. This reflects the difficulty of learning an effective critic using the standard update, as discussed in Section 4.2. In contrast, although DPAC initially learns more slowly than AC-LR and AC-RP, it substantially outperforms DPAC w/o ICL and eventually matches the performance of AC-LR and AC-RP in the

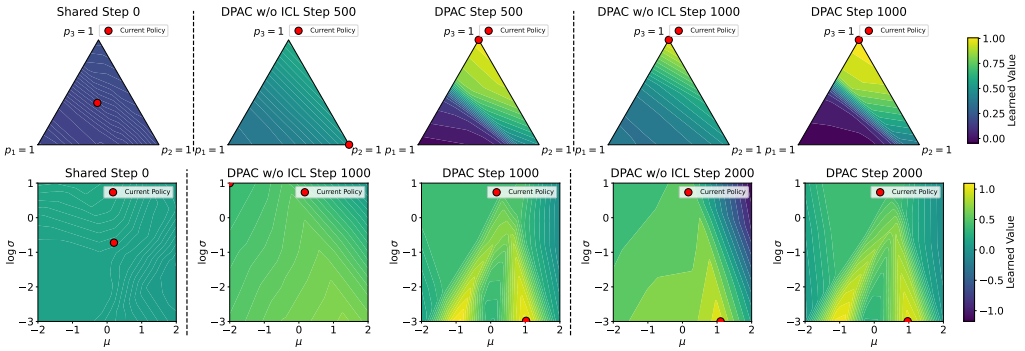

Figure 4: **Initial critic** (col 1) and **learned critic functions and policies** at different training stages using DPAC w/o ICL (cols 2 and 4) and DPAC (cols 3 and 5). **Top:** K-Armed Bandit. **Bottom:** Bimodal Continuous Bandit. DPAC produces more accurate value estimates at deterministic distribution parameters—corresponding to the vertices in the discrete case and the x-axis in the continuous case—and offers stronger gradient signals for policy optimization.

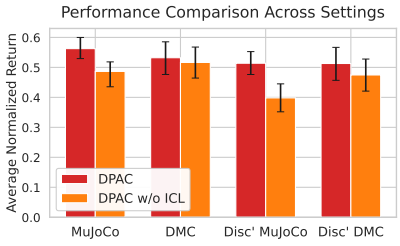

Figure 5: **Comparison between DPAC and DPAC w/o ICL** in different settings. Value are averaged over 10 seeds and 5 (MuJoCo) or 10 (DMC) tasks. Error bars show 95% bootstrapped CIs.

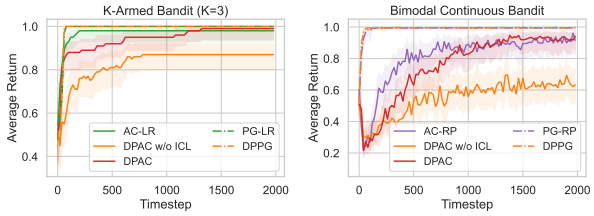

Figure 6: **Learning curves of DPAC, DPAC w/o ICL, and baselines** on the K-Armed Bandit (col 1) and Bimodal Continuous Bandit (col 2) tasks. Results are averaged over 50 seeds. Shaded regions show 95% bootstrapped CIs. ICL substantially improves DPAC's performance, enabling it to match AC-LR and AC-RP in these simple settings.

later stages of training. To assess the impact of ICL on critic quality, we visualize the learned critics from a representative training run of DPAC and DPAC w/o ICL in Figure 4. In both discrete and continuous action settings, DPAC yields a significantly improved critic landscape early in training.

6 Conclusions

We introduced the *parameters-as-actions framework*, redefining the agent-environment boundary to treat distribution parameters as actions. We showed that the policy gradient update has theoretically lower variance, and developed a practical deep RL algorithm called *Distribution Parameter Actor-Critic* (DPAC) based on this estimator and an improved critic learning update, ICL, tailored to this new setting. We showed better or comparable performance to TD3 across 20 environments.

This reframing allowed us to develop a continuous action algorithm that applies to diverse underlying action types: discrete, continuous, or otherwise structured. A key next step is test the algorithm in mixed spaces, and further exploit this reframing for new algorithmic avenues, including model-based methods, hierarchical control, or novel hybrid approaches. There are also key open questions around critic learning in this new framework. More advanced strategies for training the parameter-space critic could be explored, including off-policy updates at diverse regions of the parameter space or using a learned action-value function $Q_{\mathbf{w}}(s, a)$ to guide updates of $\bar{Q}_{\mathbf{w}'}(s, u)$. This will also open up new questions about convergence properties for these new variants.

References

Cameron Allen, Kavosh Asadi, Melrose Roderick, Abdel-rahman Mohamed, George Konidaris, and Michael Littman. Mean actor-critic. *arXiv preprint arXiv:1709.00503*, 2017.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Aditya Bhatt, Daniel Palenicek, Boris Belousov, Max Argus, Artemij Amiranashvili, Thomas Brox, and Jan Peters. CrossQ: Batch normalization in deep reinforcement learning for greater sample efficiency and simplicity. In *The Twelfth International Conference on Learning Representations*, 2024.

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

Kamil Ciosek and Shimon Whiteson. Expected policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Kamil Ciosek and Shimon Whiteson. Expected policy gradients for reinforcement learning. *Journal of Machine Learning Research*, 21(52):1–51, 2020.

Pierluca D'Oro, Max Schwarzer, Evgenii Nikishin, Pierre-Luc Bacon, Marc G Bellemare, and Aaron Courville. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *The Eleventh International Conference on Learning Representations*, 2023.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pp. 1587–1596. PMLR, 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. PMLR, 2018.

Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. *Advances in Neural Information Processing Systems*, 28, 2015.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. CleanRL: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022.

Warwick Masson, Pravesh Ranchod, and George Konidaris. Reinforcement learning with parameterized actions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

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.

A Paszke. PyTorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

Andrew Patterson, Samuel Neumann, Martha White, and Adam White. Empirical design in reinforcement learning. *Journal of Machine Learning Research*, 25(318):1–63, 2024.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

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. PMLR, 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.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

Gautham Vasan, Mohamed Elsayed, Seyed Alireza Azimi, Jiamin He, Fahim Shahriar, Colin Bellinger, Martha White, and Rupam Mahmood. Deep policy gradient methods without batch updates, target networks, or replay buffers. *Advances in Neural Information Processing Systems*, 37:845–891, 2024.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

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

Huaqing Xiong, Tengyu Xu, Lin Zhao, Yingbin Liang, and Wei Zhang. Deterministic policy gradient: Convergence analysis. In *Uncertainty in Artificial Intelligence*, pp. 2159–2169. PMLR, 2022.

Ming Xu, Matias Quiroz, Robert Kohn, and Scott A Sisson. Variance reduction properties of the reparameterization trick. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2711–2720. PMLR, 2019.

Supplementary Materials

The following content was not necessarily subject to peer review.

A Theoretical analysis of DPPG

We provide the proofs of the theoretical results for the parameters-as-actions framework and distribution parameter policy gradient (DPPG) in the main text in Appendices A.1 and A.2. In addition, we also extend a convergence proof of DPG from Xiong et al. (2022) to DPPG in Appendix A.3.

A.1 Proofs of theoretical results in Section 3

Assumption 3.1. The set $\mathcal{U}$ is compact. Moreover, when $\mathcal{S}$ or $\mathcal{A}$ is continuous, the corresponding set is also assumed to be compact.

Proposition 3.2. Under Assumption 3.1, $\bar{v}_{\bar{\pi}}(s) = v_\pi(s)$ and $\bar{q}_{\bar{\pi}}(s,u) = \mathbb{E}_{A\sim f(\cdot|u)}[q_\pi(s,A)]$.

Proof. Let π be the policy in the original MDP that first maps s to $u = \bar{\pi}(s)$ and then samples $A \sim f(\cdot|u)$. The state-value function $\bar{v}_{\bar{\pi}}(s)$ in the parameter-space MDP is defined as:

$$\bar{v}_{\bar{\pi}}(s) = \sum_{k=0}^{\infty} \mathbb{E}_{\bar{\pi}}\left[\gamma^k \bar{r}(S_k, U_k) \mid S_0 = s\right],$$

where $U_k = \bar{\pi}(S_k)$. From Equation (7), $\bar{r}(s,u) = \mathbb{E}_{A\sim f(\cdot|u)}[r(s,A)]$. Also, the transition $\bar{p}(s'|s,u) = \mathbb{E}_{A\sim f(\cdot|u)}[p(s'|s,A)]$. Consider a trajectory $S_0, U_0, S_1, U_1, \ldots$ in the parameter-space MDP. This corresponds to a trajectory $S_0, A_0, S_1, A_1, \ldots$ in the original MDP where $A_k \sim f(\cdot|U_k)$. The expected reward at time k in the parameter-space MDP, given S_k and $U_k = \bar{\pi}(S_k)$, is $\bar{r}(S_k, \bar{\pi}(S_k)) = \mathbb{E}_{A_k\sim f(\cdot|\bar{\pi}(S_k))}[r(S_k, A_k)]$. The dynamics are also equivalent in expectation: $\mathbb{E}[S_{k+1}|S_k, U_k] = \mathbb{E}_{S'\sim\bar{p}(\cdot|S_k,U_k)}[S'] = \mathbb{E}_{A_k\sim f(\cdot|U_k)}[\mathbb{E}_{S'\sim p(\cdot|S_k,A_k)}[S']]$. Thus, the sequence of states and expected rewards generated under $\bar{\pi}$ in the parameter-space MDP is identical in distribution to the sequence of states and rewards under π in the original MDP. Therefore, $\bar{v}_{\bar{\pi}}(s) = v_\pi(s)$.

For the action-value function $\bar{q}_{\bar{\pi}}(s,u)$:

$$\begin{aligned}
\bar{q}_{\bar{\pi}}(s,u) &= \mathbb{E}_{\bar{\pi}}\left[\bar{r}(S_0,U_0) + \gamma \bar{v}_{\bar{\pi}}(S_1) \mid S_0 = s, U_0 = u\right] \\
&= \bar{r}(s,u) + \gamma \mathbb{E}_{S_1\sim\bar{p}(\cdot|s,u)}[\bar{v}_{\bar{\pi}}(S_1)] \\
&= \mathbb{E}_{A\sim f(\cdot|u)}[r(s,A)] + \gamma \mathbb{E}_{A\sim f(\cdot|u)}\left[\mathbb{E}_{S_1\sim p(\cdot|s,A)}[v_\pi(S_1)]\right] \quad \text{(using } \bar{v}_{\bar{\pi}} = v_\pi\text{)} \\
&= \mathbb{E}_{A\sim f(\cdot|u)}\left[r(s,A) + \gamma \mathbb{E}_{S_1\sim p(\cdot|s,A)}[v_\pi(S_1)]\right] \\
&= \mathbb{E}_{A\sim f(\cdot|u)}\left[\mathbb{E}_\pi[R_1 + \gamma v_\pi(S_1)|S_0 = s, A_0 = A]\right] \\
&= \mathbb{E}_{A\sim f(\cdot|u)}[q_\pi(s,A)].
\end{aligned}$$

The compactness assumption in Assumption 3.1 along with continuity from Assumption 4.1 ensures these expectations and value functions are well-defined. □

A.2 Proofs of theoretical results in Section 4

Assumption 4.1. The functions $\bar{\pi}_{\boldsymbol{\theta}}(s)$, $f(a|u)$, and their derivatives are continuous with respect to the variables u and $\boldsymbol{\theta}$. Moreover, when $\mathcal{S}$ or $\mathcal{A}$ is continuous, the functions $p(s'|s,a)$, $d_0(s)$, $r(s,a)$, $\bar{\pi}_{\boldsymbol{\theta}}(s)$, $f(a|u)$, and their derivatives are also continuous with respect to s, s', or a, respectively.

Theorem 4.2 (Distribution parameter policy gradient theorem)**.** *Under Assumptions 3.1 and 4.1, the gradient of the objective function $J(\bar{\pi}_{\boldsymbol{\theta}}) = \sum_{t=0}^{\infty}\mathbb{E}_{\bar{\pi}}\left[\gamma^t R_{t+1}\right]$ with respect to $\boldsymbol{\theta}$ can be expressed as*

$$\nabla_{\boldsymbol{\theta}} J(\bar{\pi}_{\boldsymbol{\theta}}) = \mathbb{E}_{s\sim d_{\bar{\pi}_{\boldsymbol{\theta}}}}\left[\nabla_{\boldsymbol{\theta}}\bar{\pi}_{\boldsymbol{\theta}}(s)^\top \nabla_u \bar{Q}_{\bar{\pi}_{\boldsymbol{\theta}}}(s,u)|_{u=\bar{\pi}_{\boldsymbol{\theta}}(s)}\right],$$

where $d_{\bar{\pi}_{\boldsymbol{\theta}}}(s) \doteq \sum_{t=0}^{\infty}\mathbb{E}_{\bar{\pi}_{\boldsymbol{\theta}}}[\gamma^t \mathbb{I}(S_t = s)]$ is the (discounted) occupancy measure under $\bar{\pi}_{\boldsymbol{\theta}}$.

Proof. This theorem results from applying the deterministic policy gradient (DPG) theorem to the parameter-space MDP $\langle \mathcal{S}, \mathcal{U}, \bar{p}, d_0, \bar{r}, \gamma \rangle$, where $\bar{\pi}_{\boldsymbol{\theta}} : \mathcal{S} \to \mathcal{U}$ acts as a deterministic policy. The objective function is $J(\bar{\pi}_{\boldsymbol{\theta}}) = \mathbb{E}_{S_0 \sim d_0}[\bar{v}_{\bar{\pi}_{\boldsymbol{\theta}}}(S_0)]$.

Following the DPG theorem derivation (Silver et al. (2014), Theorem 1), for a general deterministic policy $\mu_{\boldsymbol{\theta}} : \mathcal{S} \to \mathcal{A}$, the policy gradient is:

$$\nabla_{\boldsymbol{\theta}} J(\mu_{\boldsymbol{\theta}}) = \mathbb{E}_{s \sim d_{\mu_{\boldsymbol{\theta}}}} \left[\nabla_{\boldsymbol{\theta}} \mu_{\boldsymbol{\theta}}(s)^\top \nabla_a q_{\mu_{\boldsymbol{\theta}}}(s, a) \big|_{a = \mu_{\boldsymbol{\theta}}(s)} \right].$$

In our context:

- The policy in the parameter-space MDP is $\bar{\pi}_{\boldsymbol{\theta}}(s)$.
- The action space is $\mathcal{U}$, and actions are denoted by u.
- The critic $\bar{q}_{\bar{\pi}_{\boldsymbol{\theta}}}(s, u)$ is the action-value function in this parameter-space MDP.
- The state distribution $d_{\bar{\pi}_{\boldsymbol{\theta}}}(s)$ is the discounted state occupancy measure under policy $\bar{\pi}_{\boldsymbol{\theta}}$.

Assumptions 3.1 and 4.1 ensure that $\bar{\pi}_{\boldsymbol{\theta}}(s)$ and $\bar{q}_{\bar{\pi}_{\boldsymbol{\theta}}}(s, u)$ are appropriately differentiable and that the interchange of expectation and differentiation is valid. Substituting $\bar{\pi}_{\boldsymbol{\theta}}$ for $\mu_{\boldsymbol{\theta}}$ and $\bar{q}_{\bar{\pi}_{\boldsymbol{\theta}}}$ for $q_{\mu_{\boldsymbol{\theta}}}$ yields the theorem's result:

$$\nabla_{\boldsymbol{\theta}} J(\bar{\pi}_{\boldsymbol{\theta}}) = \mathbb{E}_{s \sim d_{\bar{\pi}_{\boldsymbol{\theta}}}} \left[\nabla_{\boldsymbol{\theta}} \bar{\pi}_{\boldsymbol{\theta}}(s)^\top \nabla_u \bar{q}_{\bar{\pi}_{\boldsymbol{\theta}}}(s, u)|_{u = \bar{\pi}_{\boldsymbol{\theta}}(s)} \right].$$

The notation $\nabla_{\boldsymbol{\theta}} \bar{\pi}_{\boldsymbol{\theta}}(s)^\top \nabla_u \bar{q}_{\bar{\pi}_{\boldsymbol{\theta}}}$ in the theorem statement implies the appropriate vector or matrix product. If $\boldsymbol{\theta} \in \mathbb{R}^k$ and $u \in \mathbb{R}^m$, then $\nabla_{\boldsymbol{\theta}} \bar{\pi}_{\boldsymbol{\theta}}(s)$ is an $m \times k$ Jacobian, $\nabla_u \bar{q}_{\bar{\pi}_{\boldsymbol{\theta}}}$ is an $m \times 1$ vector, and the product $(\nabla_{\boldsymbol{\theta}} \bar{\pi}_{\boldsymbol{\theta}}(s))^\top \nabla_u \bar{q}_{\bar{\pi}_{\boldsymbol{\theta}}}$ results in the $k \times 1$ gradient vector for $J(\bar{\pi}_{\boldsymbol{\theta}})$. □

Proposition 4.3. If $\mathcal{U} = \mathcal{A}$ and $f(\cdot \mid u)$ is the Dirac delta distribution centered at u, then $\bar{\pi}_{\boldsymbol{\theta}}$ and $\bar{Q}_{\mathbf{w}}$ are equivalent to $\pi_{\boldsymbol{\theta}}$ and $Q_{\mathbf{w}}$, respectively. Consequently, the DPPG gradient estimator becomes equivalent to DPG:

$$\hat{\nabla}^{\text{DPPG}}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t) = \hat{\nabla}^{\text{DPG}}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t).$$

Proof. The DPPG gradient estimator is given by Equation (9):

$$\hat{\nabla}^{\text{DPPG}}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t) = \nabla_{\boldsymbol{\theta}} \bar{\pi}_{\boldsymbol{\theta}}(S_t)^\top \nabla_U \bar{Q}_{\mathbf{w}}(S_t, U)|_{U = \bar{\pi}_{\boldsymbol{\theta}}(S_t)}.$$

Given the conditions:

1. $\mathcal{U} = \mathcal{A}$: The parameter space is the action space.
2. $f(\cdot|u) = \delta(\cdot - u)$: Sampling $A \sim f(\cdot|u)$ yields $A = u$.

Under these conditions, $\bar{\pi}_{\boldsymbol{\theta}}(S_t)$ outputs parameters $U \in \mathcal{U}$, which are directly actions in $\mathcal{A}$. Thus, we can write $\pi_{\boldsymbol{\theta}}(S_t) = \bar{\pi}_{\boldsymbol{\theta}}(S_t)$, where $\pi_{\boldsymbol{\theta}}(S_t) \in \mathcal{A}$.

Next, consider the parameter-space value function $\bar{q}_{\bar{\pi}_{\boldsymbol{\theta}}}(S_t, U)$. From Proposition 3.2, $\bar{q}_{\bar{\pi}_{\boldsymbol{\theta}}}(S_t, U) = \mathbb{E}_{A \sim f(\cdot|U)}[q_{\pi_{\boldsymbol{\theta}}}(S_t, A)]$. Since $f(A|U) = \delta(A - U)$, the expectation becomes $q_{\pi_{\boldsymbol{\theta}}}(S_t, U)$. So, $\bar{q}_{\bar{\pi}_{\boldsymbol{\theta}}}(S_t, U) = q_{\pi_{\boldsymbol{\theta}}}(S_t, U)$, where $U \in \mathcal{U} = \mathcal{A}$.

This means the parameter-space critic $\bar{Q}_{\mathbf{w}}(S_t, U)$ is estimating the action-value function $q_{\pi_{\boldsymbol{\theta}}}(S_t, U)$. Thus, we can write $\bar{Q}_{\mathbf{w}}(S_t, U) = Q_{\mathbf{w}}(S_t, U)$, where $U \in \mathcal{A}$.

Substituting these equivalences into the DPPG gradient estimator:

$$\hat{\nabla}^{\text{DPPG}}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t) = \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(S_t)^\top \nabla_A Q_{\mathbf{w}}(S_t, A)|_{A = \pi_{\boldsymbol{\theta}}(S_t)}.$$

This is precisely the DPG gradient estimator (Equation (5)). Thus, $\hat{\nabla}^{\text{DPPG}}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t) = \hat{\nabla}^{\text{DPG}}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t)$. □

Proposition 4.4. Assume $Q_{\mathbf{w}} = q_{\pi_{\boldsymbol{\theta}}}$ in $\hat{\nabla}^{\text{LR}}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, A)$ and $\bar{Q}_{\mathbf{w}} = \bar{q}_{\bar{\pi}_{\boldsymbol{\theta}}}$ in $\hat{\nabla}^{\text{DPPG}}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t)$. Then,

$$\hat{\nabla}^{\text{DPPG}}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t) = \mathbb{E}_{A \sim \pi_{\boldsymbol{\theta}}(\cdot|S_t)} \left[\hat{\nabla}^{\text{LR}}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, A) \right].$$

Further, if the expectation of the action-conditioned variance is greater than zero, then

$$\mathbb{V}\left(\hat{\nabla}^{\text{DPPG}}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t) \right) < \mathbb{V}\left(\hat{\nabla}^{\text{LR}}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, A) \right).$$

Proof. Proposition 3.2 states $\bar{q}_{\bar{\pi}_{\boldsymbol{\theta}}}(S_t, U) = \mathbb{E}_{A \sim f(\cdot|U)}[q_{\pi_{\boldsymbol{\theta}}}(S_t, A)]$. Given $\bar{Q}_{\mathbf{w}} = \bar{q}_{\bar{\pi}_{\boldsymbol{\theta}}}$ and $Q_{\mathbf{w}} = q_{\pi_{\boldsymbol{\theta}}}$, this becomes $\bar{Q}_{\mathbf{w}}(S_t, U) = \mathbb{E}_{A \sim f(\cdot|U)}[Q_{\mathbf{w}}(S_t, A)]$. Note that $Q_{\mathbf{w}}(S_t, A)$ and $\bar{Q}_{\mathbf{w}}(S_t, U)$ are distinct critic functions. The use of $\mathbf{w}$ for both signifies that they are learned approximators. In the context of this proof, we can think of $Q_{\mathbf{w}}$ and $\bar{Q}_{\mathbf{w}}$ as separate approximators, each utilizing a corresponding subset of $\mathbf{w}$.

Starting with the DPPG estimator (assuming continuous $\mathcal{A}$; discrete case is analogous with sums):

$$\begin{aligned}
\hat{\nabla}^{\text{DPPG}}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t) &= \nabla_{\boldsymbol{\theta}} \bar{\pi}_{\boldsymbol{\theta}}(S_t)^\top \nabla_U \bar{Q}_{\mathbf{w}}(S_t, U)|_{U=\bar{\pi}_{\boldsymbol{\theta}}(S_t)} \\
&= \nabla_{\boldsymbol{\theta}} \bar{\pi}_{\boldsymbol{\theta}}(S_t)^\top \nabla_U \mathbb{E}_{A \sim f(\cdot|U)}[Q_{\mathbf{w}}(S_t, A)]|_{U=\bar{\pi}_{\boldsymbol{\theta}}(S_t)} \\
&= \nabla_{\boldsymbol{\theta}} \bar{\pi}_{\boldsymbol{\theta}}(S_t)^\top \left(\nabla_U \int_{\mathcal{A}} f(A|U) Q_{\mathbf{w}}(S_t, A)\, dA \right)\Big|_{U=\bar{\pi}_{\boldsymbol{\theta}}(S_t)} \\
&= \nabla_{\boldsymbol{\theta}} \bar{\pi}_{\boldsymbol{\theta}}(S_t)^\top \left(\int_{\mathcal{A}} \nabla_U f(A|U) Q_{\mathbf{w}}(S_t, A)\, dA \right)\Big|_{U=\bar{\pi}_{\boldsymbol{\theta}}(S_t)} \\
&= \nabla_{\boldsymbol{\theta}} \bar{\pi}_{\boldsymbol{\theta}}(S_t)^\top \int_{\mathcal{A}} \nabla_U f(A|U)|_{U=\bar{\pi}_{\boldsymbol{\theta}}(S_t)} Q_{\mathbf{w}}(S_t, A)\, dA \\
&= \int_{\mathcal{A}} \nabla_{\boldsymbol{\theta}} \bar{\pi}_{\boldsymbol{\theta}}(S_t)^\top \nabla_U f(A|U)|_{U=\bar{\pi}_{\boldsymbol{\theta}}(S_t)} Q_{\mathbf{w}}(S_t, A)\, dA \\
&= \int_{\mathcal{A}} \nabla_{\boldsymbol{\theta}} f(A|\bar{\pi}_{\boldsymbol{\theta}}(S_t)) Q_{\mathbf{w}}(S_t, A)\, dA.
\end{aligned}$$

The differentiability under the integral sign is justified by Assumption 4.1. The last line follows from the chain rule, where $\nabla_{\boldsymbol{\theta}} f(A|\bar{\pi}_{\boldsymbol{\theta}}(S_t)) = \nabla_{\boldsymbol{\theta}} \bar{\pi}_{\boldsymbol{\theta}}(S_t)^\top \nabla_U f(A|U)|_{U=\bar{\pi}_{\boldsymbol{\theta}}(S_t)}$.

Using $\pi_{\boldsymbol{\theta}}(A|S_t) = f(A|\bar{\pi}_{\boldsymbol{\theta}}(S_t))$ and the log-derivative trick, we can express the DPPG estimator as:

$$\begin{aligned}
\hat{\nabla}^{\text{DPPG}}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t) &= \int_{\mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(A|S_t) Q_{\mathbf{w}}(S_t, A)\, dA \\
&= \int_{\mathcal{A}} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A|S_t) \pi_{\boldsymbol{\theta}}(A|S_t) Q_{\mathbf{w}}(S_t, A)\, dA \\
&= \mathbb{E}_{A \sim \pi_{\boldsymbol{\theta}}(\cdot|S_t)} \left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A|S_t) Q_{\mathbf{w}}(S_t, A) \right].
\end{aligned}$$

The LR estimator is $\hat{\nabla}^{\text{LR}}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, A) = \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A|S_t)(Q_{\mathbf{w}}(S_t, A) - V(S_t))$. Its expectation is $\mathbb{E}_{A \sim \pi_{\boldsymbol{\theta}}(\cdot|S_t)} \left[\hat{\nabla}^{\text{LR}}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, A) \right]$. The term involving the baseline $V(S_t)$ vanishes in expectation:

$$\begin{aligned}
\mathbb{E}_{A \sim \pi_{\boldsymbol{\theta}}(\cdot|S_t)} \left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A|S_t) V(S_t) \right] &= V(S_t) \mathbb{E}_{A \sim \pi_{\boldsymbol{\theta}}(\cdot|S_t)} \left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A|S_t) \right] \\
&= V(S_t) \int_{\mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(A|S_t)\, dA \\
&= V(S_t) \nabla_{\boldsymbol{\theta}} \int_{\mathcal{A}} \pi_{\boldsymbol{\theta}}(A|S_t)\, dA = V(S_t) \nabla_{\boldsymbol{\theta}}(1) = 0.
\end{aligned}$$

Thus, $\mathbb{E}_{A \sim \pi_{\boldsymbol{\theta}}(\cdot|S_t)} \left[\hat{\nabla}^{\text{LR}}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, A) \right] = \mathbb{E}_{A \sim \pi_{\boldsymbol{\theta}}(\cdot|S_t)} \left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A|S_t) Q_{\mathbf{w}}(S_t, A) \right]$. This shows $\hat{\nabla}^{\text{DPPG}}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t) = \mathbb{E}_{A \sim \pi_{\boldsymbol{\theta}}(\cdot|S_t)} \left[\hat{\nabla}^{\text{LR}}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, A) \right]$.

For variance reduction, let $X = \hat{\nabla}^{\text{LR}}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, A)$ and $Y = \hat{\nabla}^{\text{DPPG}}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t)$. We have $Y = \mathbb{E}[X|S_t, \bar{\pi}_{\boldsymbol{\theta}}(S_t)]$ (expectation over A). By the law of total variance: $\mathbb{V}(X) = \mathbb{E}[\mathbb{V}(X|S_t, \bar{\pi}_{\boldsymbol{\theta}}(S_t))] + \mathbb{V}(\mathbb{E}[X|S_t, \bar{\pi}_{\boldsymbol{\theta}}(S_t)])$. This translates to

$$\mathbb{V}\left(\hat{\nabla}^{\text{LR}}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, A)\right) = \mathbb{E}_{S_t}\left[\mathbb{V}_A\left(\hat{\nabla}^{\text{LR}}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, A)\middle| S_t\right)\right] + \mathbb{V}\left(\hat{\nabla}^{\text{DPPG}}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t)\right).$$

If $\mathbb{E}_{S_t}\left[\mathbb{V}_A\left(\hat{\nabla}^{\text{LR}}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, A)\middle| S_t\right)\right] > 0$ (i.e., the action-conditioned variance is positive on average), then $\mathbb{V}\left(\hat{\nabla}^{\text{DPPG}}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t)\right) < \mathbb{V}\left(\hat{\nabla}^{\text{LR}}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, A)\right)$. □

Proposition 4.5. Assume $\mathcal{A}$ is continuous, $Q_{\mathbf{w}} = q_{\pi_{\boldsymbol{\theta}}}$ in $\hat{\nabla}^{\text{RP}}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, \epsilon)$, and $\bar{Q}_{\mathbf{w}} = \bar{q}_{\bar{\pi}_{\boldsymbol{\theta}}}$ in $\hat{\nabla}^{\text{DPPG}}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t)$. Then,

$$\hat{\nabla}^{\text{DPPG}}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t) = \mathbb{E}_{\epsilon \sim p}\left[\hat{\nabla}^{\text{RP}}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, \epsilon)\right].$$

Further, if the expectation of the noise-induced variance is greater than zero, then

$$\mathbb{V}\left(\hat{\nabla}^{\text{DPPG}}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t)\right) < \mathbb{V}\left(\hat{\nabla}^{\text{RP}}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, \epsilon)\right).$$

Proof. For the RP estimator, the action is generated as $A = g_{\boldsymbol{\theta}}(\epsilon; S_t)$, where $\epsilon \sim p(\cdot)$. For consistency with DPPG notation, we can write $A = g(\epsilon; U)$, where $U = \bar{\pi}_{\boldsymbol{\theta}}(S_t) \in \mathcal{U}$ represents all relevant learnable distribution parameters. Thus, the distribution $f(\cdot|U)$ of the random variable A is induced by $g(\epsilon; U)$ with $\epsilon \sim p(\cdot)$.

Similar to the proof of Proposition 4.4, given the critics are the corresponding true action-value functions, we have:

$$\bar{Q}_{\mathbf{w}}(S_t, U) = \mathbb{E}_{A \sim f(\cdot|U)}[Q_{\mathbf{w}}(S_t, A)] = \mathbb{E}_{\epsilon \sim p}\left[Q_{\mathbf{w}}(S_t, g(\epsilon; \bar{\pi}_{\boldsymbol{\theta}}(S_t)))\right],$$

where we use a change of variables to express the expectation in terms of the noise ϵ.

Now, we can express the DPPG gradient as:

$$\begin{aligned}
\hat{\nabla}^{\text{DPPG}}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t) &= \nabla_{\boldsymbol{\theta}} \bar{\pi}_{\boldsymbol{\theta}}(S_t)^\top \nabla_U \bar{Q}_{\mathbf{w}}(S_t, U)|_{U=\bar{\pi}_{\boldsymbol{\theta}}(S_t)} \\
&= \nabla_{\boldsymbol{\theta}} \bar{\pi}_{\boldsymbol{\theta}}(S_t)^\top \nabla_U \mathbb{E}_{\epsilon \sim p}[Q_{\mathbf{w}}(S_t, g(\epsilon; U))]|_{U=\bar{\pi}_{\boldsymbol{\theta}}(S_t)} \\
&= \nabla_{\boldsymbol{\theta}} \bar{\pi}_{\boldsymbol{\theta}}(S_t)^\top \mathbb{E}_{\epsilon \sim p}\left[\nabla_U Q_{\mathbf{w}}(S_t, g(\epsilon; U))|_{U=\bar{\pi}_{\boldsymbol{\theta}}(S_t)}\right] \\
&= \mathbb{E}_{\epsilon \sim p}\left[\nabla_{\boldsymbol{\theta}} \bar{\pi}_{\boldsymbol{\theta}}(S_t)^\top \nabla_U Q_{\mathbf{w}}(S_t, g(\epsilon; U))|_{U=\bar{\pi}_{\boldsymbol{\theta}}(S_t)}\right] \\
&= \mathbb{E}_{\epsilon \sim p}\left[\nabla_{\boldsymbol{\theta}} Q_{\mathbf{w}}(S_t, g(\epsilon; \bar{\pi}_{\boldsymbol{\theta}}(S_t)))\right].
\end{aligned}$$

The differentiability under the integral sign is justified by Assumption 4.1. The last line follows from the chain rule, where $\nabla_{\boldsymbol{\theta}} Q_{\mathbf{w}}(S_t, g(\epsilon; \bar{\pi}_{\boldsymbol{\theta}}(S_t))) = \nabla_{\boldsymbol{\theta}} \bar{\pi}_{\boldsymbol{\theta}}(S_t)^\top \nabla_U Q_{\mathbf{w}}(S_t, g(\epsilon; U))|_{U=\bar{\pi}_{\boldsymbol{\theta}}(S_t)}$.

On the other hand, the RP gradient is:

$$\begin{aligned}
\hat{\nabla}^{\text{RP}}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, \epsilon) &= \nabla_{\boldsymbol{\theta}} g_{\boldsymbol{\theta}}(\epsilon; S_t)^\top \nabla_A Q_{\mathbf{w}}(S_t, A)|_{A=g_{\boldsymbol{\theta}}(\epsilon; S_t)} \\
&= \nabla_{\boldsymbol{\theta}} g(\epsilon; \bar{\pi}_{\boldsymbol{\theta}}(S_t))^\top \nabla_A Q_{\mathbf{w}}(S_t, A)|_{A=g(\epsilon; \bar{\pi}_{\boldsymbol{\theta}}(S_t))} \\
&= \nabla_{\boldsymbol{\theta}} Q_{\mathbf{w}}(S_t, g(\epsilon; \bar{\pi}_{\boldsymbol{\theta}}(S_t))),
\end{aligned}$$

where we use the chain rule again in the last equation: $\nabla_{\boldsymbol{\theta}} Q_{\mathbf{w}}(S_t, g(\epsilon; \bar{\pi}_{\boldsymbol{\theta}}(S_t))) = \nabla_{\boldsymbol{\theta}} g(\epsilon; \bar{\pi}_{\boldsymbol{\theta}}(S_t))^\top \nabla_A Q_{\mathbf{w}}(S_t, A)|_{A=g(\epsilon; \bar{\pi}_{\boldsymbol{\theta}}(S_t))}$. Thus, we have:

$$\hat{\nabla}^{\text{DPPG}}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t) = \mathbb{E}_{\epsilon \sim p}\left[\hat{\nabla}^{\text{RP}}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, \epsilon)\right].$$

The variance reduction argument is similar to that in Proposition 4.4. Let $X = \hat{\nabla}^{\text{RP}}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, \epsilon)$ and $Y = \hat{\nabla}^{\text{DPPG}}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t)$. We have $Y = \mathbb{E}[X|S_t, \epsilon]$ (expectation over ϵ). By the law of total variance: $\mathbb{V}(X) = \mathbb{E}[\mathbb{V}(X|S_t, \epsilon)] + \mathbb{V}(\mathbb{E}[X|S_t, \epsilon])$. This translates to

$$\mathbb{V}\left(\hat{\nabla}^{\text{RP}}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, \epsilon)\right) = \mathbb{E}_{S_t}\left[\mathbb{V}_\epsilon\left(\hat{\nabla}^{\text{RP}}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, \epsilon)\middle| S_t\right)\right] + \mathbb{V}\left(\hat{\nabla}^{\text{DPPG}}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t)\right).$$

If $\mathbb{E}_{S_t}\left[\mathbb{V}_\epsilon\left(\hat{\nabla}^{\text{RP}}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, \epsilon)\middle| S_t\right)\right] > 0$ (i.e., the noise-induced variance is positive on average), then $\mathbb{V}\left(\hat{\nabla}^{\text{DPPG}}_{\boldsymbol{\theta}} \hat{J}(\bar{\pi}_{\boldsymbol{\theta}}; S_t)\right) < \mathbb{V}\left(\hat{\nabla}^{\text{RP}}_{\boldsymbol{\theta}} \hat{J}(\pi_{\boldsymbol{\theta}}; S_t, \epsilon)\right)$. □

A.3 Convergence analysis for DPPG

We present a convergence result for the distribution parameter policy gradient (DPPG), which is a direct application of the convergence of the deterministic policy gradient (DPG; Xiong et al., 2022). We assume an on-policy linear function approximation setting and use TD learning to learn the critic. See Algorithm 1 for the analyzed DPPG-TD algorithm. We follow the notation of Xiong et al. as much as possible for comparison with their results.

Algorithm 1 DPPG-TD

Input: $\alpha_w, \alpha_\theta, w_0, \theta_0$, batch size M.
for $t = 0, 1, \ldots, T$ **do**
  for $j = 0, 1, \ldots, M-1$ **do**
    Sample $s_{t,j} \sim d_{\theta_t}$.
    Generate $u_{t,j} = \bar{\pi}_{\theta_t}(s_{t,j})$.
    Sample $s_{t+1,j} \sim \bar{p}(\cdot|s_{t,j}, u_{t,j})$ and $r_{t,j}$.
    Generate $u_{t+1,j} = \bar{\pi}_{\theta_t}(s_{t+1,j})$.
    Denote $x_{t,j} = (s_{t,j}, u_{t,j})$.
    $\delta_{t,j} = r_{t,j} + \gamma\phi(x_{t+1,j})^\top w_t - \phi(x_{t,j})^\top w_t$.
  end for
  $w_{t+1} = w_t + \frac{\alpha_w}{M}\sum_{j=0}^{M-1} \delta_{t,j}\phi(x_{t,j})$.
  for $j = 0, 1, \ldots, M-1$ **do**
    Sample $s'_{t,j} \sim \nu_{\theta_t}$.
  end for
  $\theta_{t+1} = \theta_t + \frac{\alpha_\theta}{M}\sum_{j=0}^{M-1} \nabla_\theta \bar{\pi}_{\theta_t}(s'_{t,j}) \nabla_\theta \bar{\pi}_{\theta_t}(s'_{t,j})^\top w_t$.
end for

Following their notation, the parameterized policy is denoted as $\bar{\pi}_\theta$ and the objective function $J(\bar{\pi}_\theta)$ (Equation (1)) is denoted as $J(\theta)$. The distribution parameter policy gradient is

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim\nu_\theta}\left[\nabla_\theta \bar{\pi}_\theta(s) \nabla_u \bar{q}_{\bar{\pi}_\theta}(s, u)|_{u=\bar{\pi}_\theta(s)}\right], \tag{13}$$

where $\nu_\theta(s) \doteq \sum_{t=0}^{\infty} \mathbb{E}_{\bar{\pi}_\theta}[\gamma^t \mathbb{I}(S_t = s)]$ is the discounted occupancy measure under $\bar{\pi}_\theta$. We also define the stationary distribution of $\bar{\pi}_\theta$ to be $d_\theta(s) = \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}_{\bar{\pi}_\theta}[\mathbb{I}(S_t = s)]$. Under linear function approximation for the critic function, the parameterized critic can be expressed as $\bar{Q}_w(s, u) = \phi(s, u)^\top w$, where $\phi : \mathcal{S} \times \mathcal{U} \to \mathbb{R}^d$ is the feature function.

We will first list the full set of assumptions needed for the convergence result, followed by the convergence theorem. In addition, we incorporate the corrections to the result of Xiong et al. from Vasan et al. (2024), which extends the result to the reparameterization policy gradient. Following Vasan et al., the corrections are highlighted in red.

Assumption A.1. For any $\theta_1, \theta_2, \theta \in \mathbb{R}^d$, there exist positive constants $L_{\bar{\pi}}, L_\phi$ and λ_Φ, such that (1) $\|\bar{\pi}_{\theta_1}(s) - \bar{\pi}_{\theta_2}(s)\| \le L_{\bar{\pi}}\|\theta_1 - \theta_2\|, \forall s \in \mathcal{S}$; (2) $\|\nabla_\theta \bar{\pi}_{\theta_1}(s) - \nabla_\theta \bar{\pi}_{\theta_2}(s)\| \le L_\psi\|\theta_1 - \theta_2\|, \forall s \in \mathcal{S}$; (3) the matrix $\Psi_\theta := \mathbb{E}_{\nu_\theta}\left[\nabla_\theta \bar{\pi}_\theta(s) \nabla_\theta \bar{\pi}_\theta(s)^\top\right]$ is non-singular with the minimal eigenvalue uniformly lower-bounded as $\sigma_{\min}(\Psi_\theta) \ge \lambda_\Psi$.

Assumption A.2. For any $u_1, u_2 \in \mathcal{U}$, there exist positive constants $L_{\bar{p}}, L_{\bar{r}}$, such that (1) the parameter-space transition kernel satisfies $|\bar{p}(s'|s,u_1) - \bar{p}(s'|s,u_2)| \leq L_{\bar{p}}\|u_1 - u_2\|, \forall s, s' \in \mathcal{S}$; (2) the parameter-space reward function satisfies $|\bar{r}(s,u_1) - \bar{r}(s,u_2)| \leq L_{\bar{r}}\|u_1 - u_2\|, \forall s, s' \in \mathcal{S}$.

Assumption A.3. For any $u_1, u_2 \in \mathcal{U}$, there exists a positive constant $L_{\bar{q}}$, such that $\|\nabla_u \bar{q}_{\bar{\pi}_\theta}(s,u_1) - \nabla_u \bar{q}_{\bar{\pi}_\theta}(s,u_2)\| \leq L_{\bar{q}}\|u_1 - u_2\|, \forall \theta \in \mathbb{R}^d, s \in \mathcal{S}$.

Assumption A.4. The feature function $\phi : \mathcal{S} \times \mathcal{U} \to \mathbb{R}^d$ is uniformly bounded, i.e., $\|\phi(\cdot,\cdot)\| \leq C_\phi$ for some positive constant C_ϕ. In addition, we define $A = \mathbb{E}_{d_\theta}\left[\phi(x)(\gamma\phi(x') - \phi(x))^\top\right]$ and $D = \mathbb{E}_{d_\theta}\left[\phi(x)\phi(x)^\top\right]$, and assume that A and D are non-singular. We further assume that the absolute value of the eigenvalues of A are uniformly lower bounded, i.e., $|\sigma(A)| \geq \lambda_A$ for some positive constant λ_A.

Proposition A.5 (Compatible function approximation)**.** *A function estimator $\bar{Q}_w(s,u)$ is compatible with a policy $\bar{\pi}_\theta$, i.e., $\nabla J(\theta) = \mathbb{E}_{\nu_\theta}\left[\nabla_\theta \bar{\pi}_\theta(s) \nabla_u \bar{Q}_w(s,u)|_{u=\bar{\pi}_\theta(s)}\right]$, if it satisfies the following two conditions:*

1. $\nabla_u \bar{Q}_w(s,u)|_{u=\bar{\pi}_\theta(s)} = \nabla_\theta \bar{\pi}_\theta(s)^\top w$;
2. $w = w^*_{\xi_\theta}$ *minimizes the mean square error* $\mathbb{E}_{\nu_\theta}\left[\xi(s;\theta,w)^\top \xi(s;\theta,w)\right]$, *where* $\xi(s;\theta,w) = \nabla_u \bar{Q}_w(s,u)|_{u=\bar{\pi}_\theta(s)} - \nabla_u \bar{q}_{\bar{\pi}_\theta}(s,u)|_{u=\bar{\pi}_\theta(s)}$.

Given the above assumption, one can show that the distribution parameter policy gradient is smooth (Lemma A.6), and that Algorithm 1 converges (Theorem A.7).

Lemma A.6. *Suppose Assumptions A.1-A.3 hold. Then the distribution parameter policy gradient $\nabla J(\theta)$ defined in Equation (13) is Lipschitz continuous with the parameter L_J, i.e., $\forall \theta_1, \theta_2 \in \mathbb{R}^d$,*

$$\|\nabla J(\theta_1) - \nabla J(\theta_2)\| \leq L_J \|\theta_1 - \theta_2\|, \tag{14}$$

where $L_J = \left(\frac{1}{2}L_{\bar{p}} L_{\bar{\pi}}^2 L_\nu C_\nu + \frac{L_\psi}{1-\gamma}\right)\left(L_{\bar{r}} + \frac{\gamma R_{\max} L_{\bar{p}}}{1-\gamma}\right) + \frac{L_{\bar{\pi}}}{1-\gamma}\left(L_{\bar{q}} L_{\bar{\pi}} + \frac{\gamma}{2} L_{\bar{p}}^2 R_{\max} L_{\bar{\pi}} C_\nu + \frac{\gamma L_{\bar{p}} L_{\bar{r}} L_{\bar{\pi}}}{1-\gamma}\right)$.

Theorem A.7. *Suppose that Assumptions A.1-A.4 hold. Let* $\alpha_w \leq \frac{\lambda}{2C_A^2}; M \geq \frac{48\alpha_w C_A^2}{\lambda}; \alpha_\theta \leq \min\left\{\frac{1}{4L_J}, \frac{\lambda\alpha_w}{24\sqrt{6}L_h L_w}\right\}$. *Then the output of DPPG-TD in Algorithm 1 satisfies*

$$\min_{t\in[T]} \mathbb{E}\|\nabla J(\theta_t)\|^2 \leq \frac{c_1}{T} + \frac{c_2}{M} + c_3\kappa^2,$$

where $c_1 = \frac{8R_{\max}}{\alpha_\theta(1-\gamma)} + \frac{144L_h^2}{\lambda\alpha_w}\|w_0 - w^*_{\theta_0}\|^2, c_2 = \left[48\alpha_w^2(C_A^2 C_w^2 + C_b^2) + \frac{96 L_w^2 L_{\bar{\pi}}^4 C_{w_\xi}^2 \alpha_\theta^2}{\lambda\alpha_w}\right] \cdot \frac{144L_h^2}{\lambda\alpha_w} + 72L_{\bar{\pi}}^4 C_{w_\xi}^2, c_3 = 18L_h^2 + \left[\frac{24L_w^2 L_h^2 \alpha_\theta^2}{\lambda\alpha_w} + \frac{24}{\lambda\alpha_w}\right] \cdot \frac{144L_h^2}{\lambda\alpha_w}$ *with* $C_A = 2C_\phi^2, C_b = R_{\max} C_\phi, C_w = \frac{R_{\max} C_\phi}{\lambda_A}, C_{w_\xi} = \frac{L_{\bar{\pi}} C_{\bar{q}}}{\lambda_\Psi(1-\gamma)}, L_w = \frac{L_J}{\lambda_\Psi} + \frac{L_{\bar{\pi}} C_{\bar{q}}}{\lambda_\Psi^2(1-\gamma)}\left(L_{\bar{\pi}}^2 L_\nu + \frac{2L_{\bar{\pi}} L_\psi}{1-\gamma}\right), L_h = L_{\bar{\pi}}^2, C_{\bar{q}} = L_{\bar{r}} + L_{\bar{p}} \cdot \frac{\gamma R_{\max}}{1-\gamma}, L_\nu = \frac{1}{2} C_\nu L_{\bar{p}} L_{\bar{\pi}}$, *and L_J defined in Lemma A.6, and we define*

$$\kappa := \max_\theta \|w^*_\theta - w^*_{\xi_\theta}\|. \tag{15}$$

Remark A.8. Apart from the corrections highlighted in red, the convergence result retains the same mathematical form as the DPG convergence result (see Theorem 1 of Xiong et al. (2022)). However, the associated constants differ, as they are defined with respect to the parameter-space formulations of the MDP, policy, and critic. Notably, the parameter-space policy class strictly generalizes the deterministic policy class. Consequently, this convergence result constitutes a strict generalization of the DPG convergence result.

The proofs of Lemma A.6 and Theorem A.7 follow the same lines as that of Lemma 1 and Theorem 1 of Xiong et al.. We refer the reader to Xiong et al. for proofs and discussion and Vasan et al. for details about the corrections.

B Experimental details

Our implementation builds upon a PyTorch (Paszke, 2019) implementation of TD3 from CleanRL (Huang et al., 2022), distributed under the MIT license.

Since the performance distribution in reinforcement learning (RL) is often not Gaussian, we use 95% bootstrapped confidence intervals (CIs) for reporting the statistical significance whenever applicable, as recommended by Patterson et al. (2024). We use `scipy.stats.bootstrap` with $10,000$ resamples from SciPy to calculate the bootstrapped CIs. For all bar plots, we plot the final performance, which is computed using the average of the return collected during the final 10% training steps.

B.1 Policy parameterization and action sampling

When the action space is multidimensional, we treat each dimension independently. For simplicity, our exposition will focus on the unidimensional case in the remaining of the paper.

Discrete action spaces We use the categorical policy parameterization: $A \sim f(\cdot|[p_1, \cdots, p_N]^\top)$, where $f(x|[p_1, \cdots, p_N]^\top) = \prod_{i=1}^{N} p_i^{\mathbb{I}(x=i)}$ is the probability mass function for the categorical distribution. For DPAC, we choose the probability vector $u = [p_1, \cdots, p_N]^\top$ as the distribution parameters. We define the distribution parameters corresponding to an action A to be the one-hot vector $U_A = \text{one\_hot}(A)$.

Continuous action spaces Assume the action space is $[a_{\min}, a_{\max}]$. We use the Gaussian policy parameterization that is used in TD3: $A = \text{clip}(\mu + \epsilon, a_{\min}, a_{\max}), \epsilon \sim \mathcal{N}(0, \sigma)$. Same as TD3, we restrict the mean μ to be within $[a_{\min}, a_{\max}]$ using a squashing function:

$$\mu = \frac{u_\mu + 1}{2}(a_{\max} - a_{\min}) + a_{\min}, \quad u_\mu = \tanh(\text{logit}_\mu),$$

where $\text{logit}_\mu \in \mathbb{R}$ is the actor network's output for μ. While TD3 uses a fixed $\sigma_{\text{TD3}} = 0.1 * \frac{a_{\max} - a_{\min}}{2}$, we allow the learnable standard deviation to be within a range $\sigma \in [\sigma_{\min}, \sigma_{\max}]$:

$$\log \sigma = \frac{u_\sigma + 1}{2} * (\log \sigma_{\max} - \log \sigma_{\min}) + \log \sigma_{\min}, \quad u_\sigma = \tanh(\text{logit}_\sigma),$$

where $\text{logit}_\sigma \in \mathbb{R}$ is the actor network's output for σ. For AC-RP, the reparameterization function is $g_{\boldsymbol{\theta}}(\epsilon; S_t) = \text{clip}(\mu_{\boldsymbol{\theta}}(S_t) + \sigma_{\boldsymbol{\theta}}(S_t)\epsilon, a_{\min}, a_{\max}), \epsilon \sim \mathcal{N}(0, 1)$. This setup may slightly disadvantage AC-RP, as the policy gradient becomes zero when sampled actions fall outside the support of the action space. For DPAC, we choose the distribution parameters to be $u = [u_\mu, u_\sigma]^\top \in [-1, 1]^2$ so that the parameter space is consistent across the mean and standard deviation dimensions. Since we lower bound the standard deviation space to encourage exploration, we define the distribution parameters corresponding to an action A to be $U_A = [\frac{2A}{a_{\max} - a_{\min}}, -1]^\top$ to approximate the Dirac delta distribution, which corresponds to $\mu = A$ and $\sigma = \sigma_{\min}$.

B.2 Policy evaluation in bandits

K-Armed Bandit We use a K-armed bandit with $K = 3$ and a deterministic reward function:

$$r(a_1) = 0, \quad r(a_2) = 0.5, \quad r(a_3) = 1.$$

Bimodal Continuous Bandit We use a continuous bandit with a deterministic bimodal reward function. Specifically, the reward function is the normalized summation of two Gaussians' density functions whose standard deviations are both 0.5 and whose means are -1 and 1, respectively:

$$r(a) = e^{-\frac{(a+1)^2}{0.5}} + e^{-\frac{(a-1)^2}{0.5}}.$$

We restrict the action space to be $[a_{\min}, a_{\max}] = [-2, 2]$. The standard standard deviation is constrained to $[\sigma_{\min}, \sigma_{\max}] = [e^{-3}, e]$.

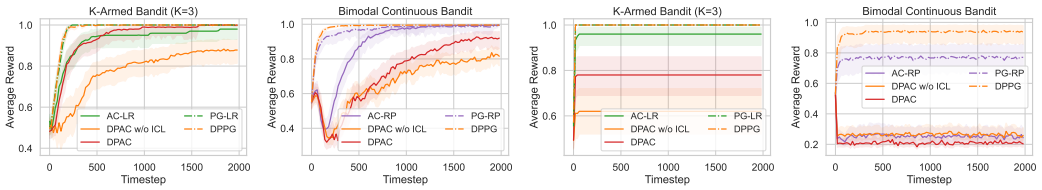

Figure 7: **Learning curves of DPAC, DPAC w/o ICL, and baselines** using learning rates 0.001 (cols 1–2) and 0.1 (cols 3–4). Results are averaged over 50 seeds. Shaded regions show 95% bootstrapped CIs. An aggressive learning rate of 0.1 often leads to premature convergence to suboptimal points for most algorithms. Consistent with Figure 6, ICL demonstrates improved performance for DPAC when a more conservative learning rate is employed.

Critic network architecture To be consistent with the RL settings, we use the same critic network architecture as those in Appendices B.4 and B.5. Specifically, we use a two-layer MLP network with the concatenated state and action vector as input. We reduce the hidden size from 256 to 16 and use a dummy state vector with a value of 1.

Experimental details We keep the policy evaluation (PE) policy fixed and update the parameter-space critic function for 2000 steps using either Equation (10) or Equation (12). In K-Armed Bandit, the PE policy is $\bar{\pi}_{\text{PE}} = u_{\text{PE}} = [1/3, 1/3, 1/3]$; in Bimodal Continuous Bandit, the PE policy is $\bar{\pi}_{\text{PE}} = u_{\text{PE}} = [0, 0.5]$ (corresponding to $\mu = 0$ and $\log \sigma = 0.0$). The hyperparameters are the same as those of DPAC in Table 3, except that the batch size is 32, and the actor is kept fixed to the corresponding PE policy.

B.3 Policy optimization in bandits

Details We use the same K-Armed Bandit and Bimodal Continuous Bandit environments as Appendix B.2. We use the same critic network architecture as in Appendix B.2. Similarly, we use the same actor network architecture as those in Appendices B.4 and B.5. Specifically, we use a two-layer MLP network with the state vector as input. We reduce the hidden size from 256 to 16 and use a dummy state tensor with a value of 1. The hyperparameters are in Table 3. For PG-LR, PG-RP, and DPPG, the critic function is calculated analytically; otherwise, their hyperparameters are the same as their counterparts with a learned critic function.

Results with alternative learning rates While we choose a fixed learning rate for all algorithms for a more controlled comparison in Section 5.2, we note that interpolated critic learning (ICL) also improves the performance of DPAC under other learning rates. Apart from 0.01, we report the results with learning rates 0.001 and 0.1 in Figure 7.

B.4 Continuous control

Environments From OpenAI Gym MuJoCo, we use the most commonly used 5 environments (see Table 1). From DeepMind Control Suite, we use the same 15 environments as D'Oro et al. (2023), which are mentioned to be neither immediately solvable nor unsolvable by common deep RL algorithms. The full list of environments and their corresponding observation and action space dimensions are in Table 2. Returns for bar plots are normalized by dividing the episodic return by the maximum possible return for a given task. In DMC environments, the maximum return is 1000 (Tunyasuvunakool et al., 2020). For MuJoCo environments, we establish maximum returns based on the highest values observed from proficient RL algorithms (Bhatt et al., 2024): 4000 for Hopper-v4, 7000 for Walker2d-v4, 8000 for Ant-v4, 16000 for HalfCheetah-v4, and 12000 for Humanoid-v4.

Experimental details Similar to TD3, DPAC and AC-RP also adopt a uniform exploration phase. During the uniform exploration phase, the distribution parameters $u = [u_\mu, u_\sigma]^\top$ are uni-

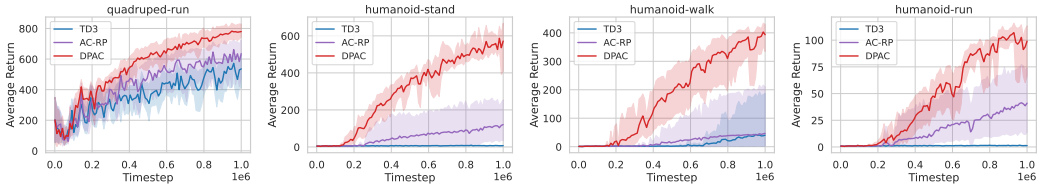

Figure 8: **Learning curves in four DeepMind Control tasks with high-dimensional action spaces.** Results are averaged over 10 seeds. shaded regions show 95% bootstrapped CIs.

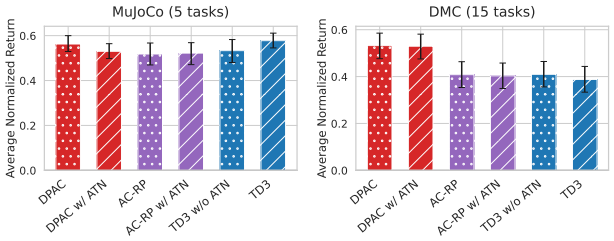

Figure 9: **Average normalized returns with and without actor target network (ATN)** on MuJoCo (col 1) and DMC (col 2) tasks. Values are averaged over 10 seeds and 5 (MuJoCo) or 10 (DMC) tasks. Error bars show 95% bootstrapped CIs.

formly sampled from $[-1, 1]^2$. All algorithms use the default hyperparameters of TD3 (see Table 4). See Figure 11 for learning curves in each individual environment.

Impact of the actor target network We also investigate the impact of using an actor target network (ATN) in DPAC and the baselines. While TD3 already employs an ATN, both DPAC and AC-RP do not. We additionally test DPAC w/ ATN and AC-RP w/ ATN and TD3 w/o ATN. From Figure 9, we can see that the actor target network does not have a significant impact in general.

B.5 Discrete control

Details We use the same 20 environments as Appendix B.4. Similar to the continuous control case, we also include a uninform exploration phase for all discrete control algorithms. For AC-LR and AC-ST, the action is randomly sampled from a uniform categorical distribution. For DPAC, the logits of the distribution parameters (in this case, the probability vector) are sampled from $\mathcal{N}(0, 1)^N$, where N is the number of possible discrete outcomes. All algorithms use the default hyperparameters of TD3 (see Table 4). See Figure 12 for learning curves in each individual environment.

Comparison to continuous control We plot the relative final performance of DPAC with continuous actions versus with discrete actions in Figure 10. We can see that the performance of DPAC with discrete actions can often compete with DPAC with continuous actions.

B.6 Computational resource requirement

All training for bandits was conducted on a local machine with AMD Ryzen 9 5900X 12-Core Processor. Each training run was executed using a single CPU core and consumed less than 256MB of RAM. Most runs completed 2000 training steps within 10 seconds.

All training for the MuJoCo simulation tasks was conducted on CPU servers. These servers were equipped with a diverse range of Intel Xeon processors, including Intel E5-2683 v4 Broadwell @ 2.1GHz, Intel Platinum 8160F Skylake @ 2.1GHz, and Intel Platinum 8260 Cascade Lake @ 2.4GHz. Each training run was executed using a single CPU core and consumed less than 2GB of RAM. The training duration varied considerably across environments, primarily influenced by the

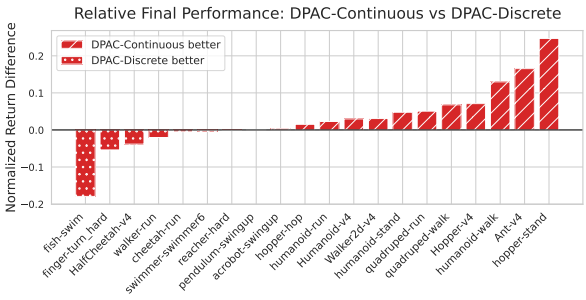

Figure 10: **Relative final performance of DPAC with continuous actions versus with discrete actions** across 20 individual control tasks. Results are averaged over 10 seeds per task.

dimensionality of the observation space, the complexity of the physics simulation, and, in the discrete action case, the dimensionality of the action space. Most algorithms typically completed $1M$ training steps in approximately 7 hours per run. However, AC-LR required a longer training period of roughly 9 hours due to the additional computational overhead of learning an extra neural network.

Table 1: Observation and action dimensions of OpenAI Gym MuJoCo environments.

Environment	**Observation dimension**	**Action dimension**
Hopper-v3	11	3
Walker2d-v3	17	6
HalfCheetah-v3	17	6
Ant-v3	111	8
Humanoid-v3	376	17

Table 2: Observation and action dimensions of DeepMind Control Suite environments.

Domain	**Task(s)**	**Observation dimension**	**Action dimension**
pendulum	swingup	3	1
acrobot	swingup	6	1
reacher	hard	6	2
finger	turn_hard	12	2
hopper	stand, hop	15	4
fish	swim	24	5
swimmer	swimmer6	25	5
cheetah	run	17	6
walker	run	24	6
quadruped	walk, run	58	12
humanoid	stand, walk, run	67	24

Table 3: Hyperparameters of actor-critic algorithms for both continuous (col 2) and discrete (col 3) bandits. DPAC is applied to both settings, denoted as DPAC-C and DPAC-D, respectively.

Hyperparameter	**AC-RP / DPAC-C**	**AC-LR / DPAC-D**
Batch size	8	
Optimizer	Adam	
Learning rate (actor & critic)	0.01	
Target network update rate (τ)	0.005	
Gradient steps per env step	1	
Number of hidden layers	2	
Neurons per hidden layer	16	
Activation function	ReLU	
Discount factor (γ)	N/A	
Replay buffer size	2000	
Uniform exploration steps	N/A	
Policy update delay (N_d)	1	
Learnable σ range ($[\sigma_{\min}, \sigma_{\max}]$)	$[e^{-3}, e]$	N/A

Table 4: Hyperparameters of actor-critic algorithms for both continuous (cols 2–3) and discrete (col 4) control RL environments. DPAC is applied to both settings, denoted as DPAC-C and DPAC-D, respectively. For simplicity, we assume $[a_{\min}, a_{\max}] = [-1, 1]$ for continuous control algorithms.

Hyperparameter	**TD3**	**AC-RP / DPAC-C**	**AC-LR / AC-ST / DPAC-D**
Batch size	256		
Optimizer	Adam		
Learning rate (actor & critic)	3×10^{-4}		
Target network update rate (τ)	0.005		
Gradient steps per env step	1		
Number of hidden layers	2		
Neurons per hidden layer	256		
Activation function	ReLU		
Discount factor (γ)	0.99		
Replay buffer size	1×10^6		
Uniform exploration steps	$25,000$		
Policy update delay (N_d)	2		
Target policy noise clip (c)	0.5	N/A	
Target policy noise ($\tilde{\sigma}_{\text{TD3}}$)	0.2	N/A	
Exploration policy noise (σ_{TD3})	0.1	N/A	
Learnable σ range ($[\sigma_{\min}, \sigma_{\max}]$)	N/A	$[0.05, 0.2]$	N/A

C Additional plots

We present the learning curves for individual continuous and discrete tasks in Figures 11 and 12, respectively.

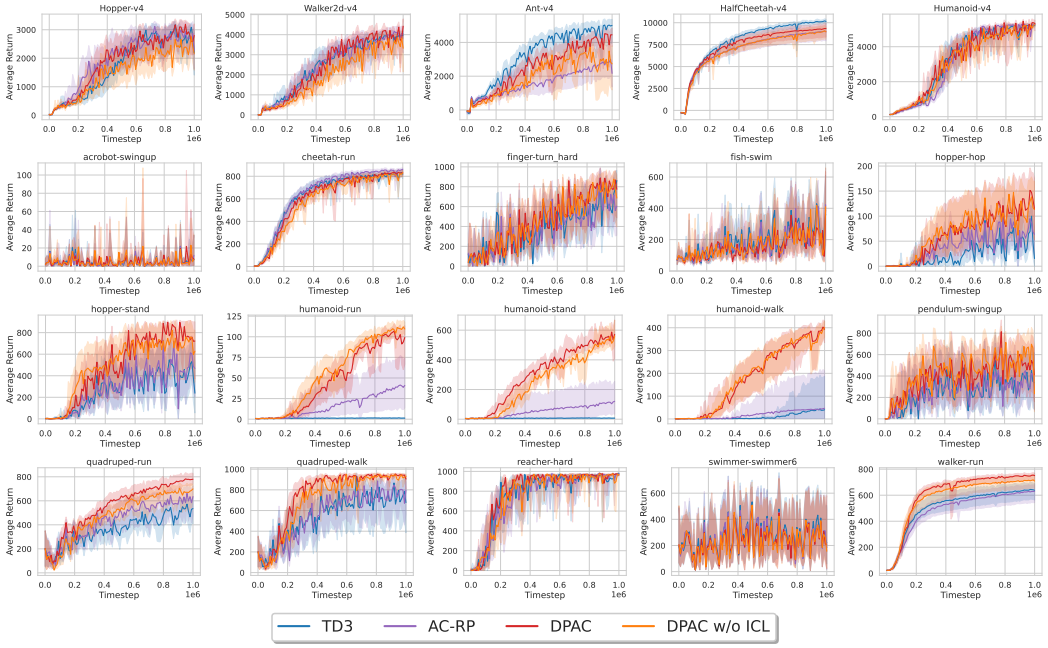

Figure 11: **Learning curves of DPAC, DPAC w/o ICL, and baselines** in 20 individual *continuous* control tasks. Results are averaged over 10 seeds. Shaded regions show 95% bootstrapped CIs.

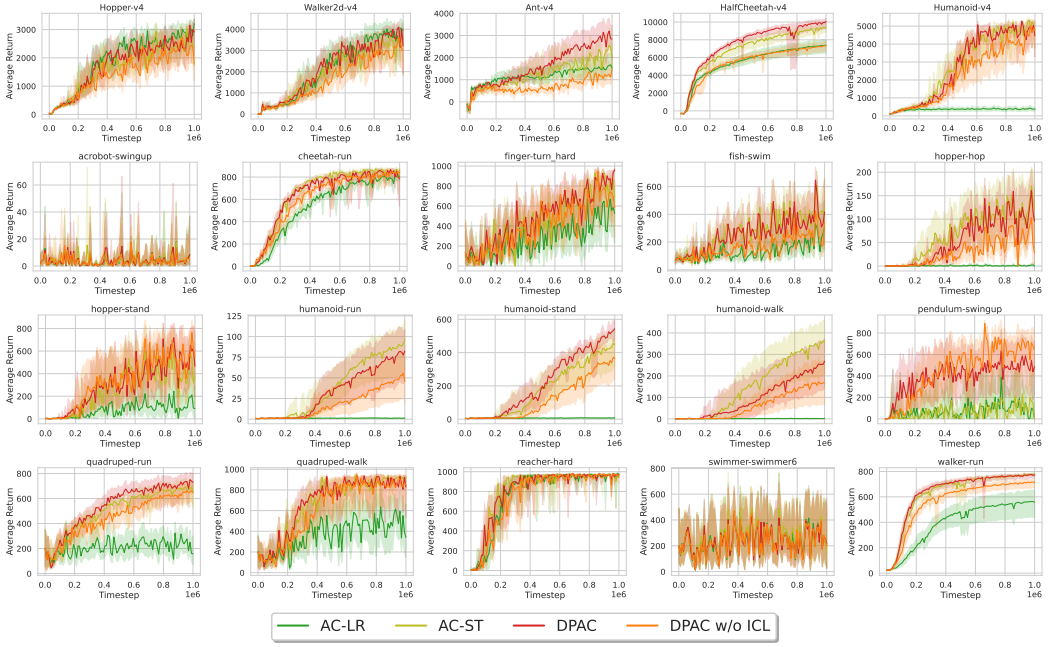

Figure 12: **Learning curves of DPAC, DPAC w/o ICL, and baselines** in 20 individual *discrete* control tasks. Results are averaged over 10 seeds. Shaded regions show 95% bootstrapped CIs.

D Pseudocode

Algorithm 2 DPAC for diverse action spaces

Input action sampling function $f : \mathcal{U} \to \Delta(\mathcal{A})$ (see Appendix B.1 for f in different action spaces)

Initialize parameters $\mathbf{w}_1, \mathbf{w}_2, \boldsymbol{\theta}$, $\bar{\mathbf{w}}_1 \leftarrow \mathbf{w}_1$, $\bar{\mathbf{w}}_2 \leftarrow \mathbf{w}_2$, replay buffer $\mathcal{B}$
Obtain initial state S_0
for $t = 1$ to T **do**
  Take action $A_t \sim f(\cdot|U_t)$ with $U_t = \bar{\pi}_{\boldsymbol{\theta}}(S_t)$, observe R_{t+1}, S_{t+1}
  Add $\langle S_t, U_t, A_t, S_{t+1}, R_{t+1}\rangle$ to the buffer $\mathcal{B}$
  Sample a mini-batch B from buffer $\mathcal{B}$
  Sample $\hat{U} = \omega U + (1-\omega)U_A$, $\omega \sim \mathrm{Uniform}[0,1]$, for each transition $\langle S, U, A, S', R\rangle$ in B
  Update critics on B:

$$\mathbf{w}_i \leftarrow \mathbf{w}_i + \alpha_t (R + \gamma \min_{j\in\{1,2\}} Q_{\bar{\mathbf{w}}_j}(S', \bar{\pi}_{\boldsymbol{\theta}}(S')) - Q_{\mathbf{w}_i}(S, \hat{U})) \nabla Q_{\mathbf{w}_i}(S, \hat{U})$$

  if $t \equiv 0 \pmod{N_d}$ **then**
    Update policy on B:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha_t \nabla_{\boldsymbol{\theta}} \bar{\pi}_{\boldsymbol{\theta}}(S)^\top \nabla_{\tilde{U}} Q_{\mathbf{w}_1}(S, \tilde{U})|_{\tilde{U}=\bar{\pi}_{\boldsymbol{\theta}}(S)}$$

    Update target network weights:

$$\bar{\mathbf{w}}_i \leftarrow \tau \mathbf{w}_i + (1-\tau)\bar{\mathbf{w}}_i$$

  end if
end for

Algorithm 3 TD3 for continuous action spaces

Input exploration noise σ_{TD3}, target policy noise $\tilde{\sigma}_{\text{TD3}}$, target noise clipping c
Initialize parameters $\mathbf{w}_1, \mathbf{w}_2, \boldsymbol{\theta}$, $\bar{\mathbf{w}}_1 \leftarrow \mathbf{w}_1$, $\bar{\mathbf{w}}_2 \leftarrow \mathbf{w}_2$, $\bar{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta}$, replay buffer $\mathcal{B}$
Obtain initial state S_0
for $t = 1$ to T **do**
  Take action $A_t = \pi_{\boldsymbol{\theta}}(S_t) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma_{\text{TD3}})$, and observe R_{t+1}, S_{t+1}
  Add $\langle S_t, A_t, S_{t+1}, R_{t+1}\rangle$ to the buffer $\mathcal{B}$
  Sample a mini-batch B from buffer $\mathcal{B}$
  Sample $A' = \pi_{\bar{\boldsymbol{\theta}}}(S') + \epsilon$, $\epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}_{\text{TD3}}), -c, c)$, for each transition $\langle S, A, S', R\rangle$ in B
  Update critics on B:

$$\mathbf{w}_i \leftarrow \mathbf{w}_i + \alpha_t (R + \gamma \min_{j\in\{1,2\}} Q_{\bar{\mathbf{w}}_j}(S', A') - Q_{\mathbf{w}_i}(S, A)) \nabla Q_{\mathbf{w}_i}(S, A)$$

  if $t \equiv 0 \pmod{N_d}$ **then**
    Update policy on B:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha_t \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(S)^\top \nabla_{\tilde{A}} Q_{\mathbf{w}_1}(S, \tilde{A})|_{\tilde{A}=\pi_{\boldsymbol{\theta}}(S)}$$

    Update target network weights:

$$\bar{\mathbf{w}}_i \leftarrow \tau \mathbf{w}_i + (1-\tau)\bar{\mathbf{w}}_i$$
$$\bar{\boldsymbol{\theta}} \leftarrow \tau \boldsymbol{\theta} + (1-\tau)\bar{\boldsymbol{\theta}}$$

  end if
end for

Algorithm 4 AC-RP for continuous action spaces

Input reparameterization function $g_{\boldsymbol{\theta}} : \mathcal{S} \times \mathbb{R} \to \mathcal{A}$ (for Gaussian policies, see Appendix B.1)
Initialize parameters $\mathbf{w}_1, \mathbf{w}_2, \boldsymbol{\theta}$, $\bar{\mathbf{w}}_1 \leftarrow \mathbf{w}_1$, $\bar{\mathbf{w}}_2 \leftarrow \mathbf{w}_2$, replay buffer $\mathcal{B}$
Obtain initial state S_0
for $t = 1$ to T **do**
  Take action $A_t = g_{\boldsymbol{\theta}}(\epsilon; S_t)$, $\epsilon \sim \mathcal{N}(0, 1)$, and observe R_{t+1}, S_{t+1}
  Add $\langle S_t, A_t, S_{t+1}, R_{t+1}\rangle$ to the buffer $\mathcal{B}$
  Sample a mini-batch B from buffer $\mathcal{B}$
  Sample $A' = g_{\boldsymbol{\theta}}(\epsilon; S')$, $\epsilon \sim \mathcal{N}(0, 1)$, for each transition $\langle S, A, S', R\rangle$ in B
  Update critics on B:

$$\mathbf{w}_i \leftarrow \mathbf{w}_i + \alpha_t (R + \gamma \min_{j\in\{1,2\}} Q_{\bar{\mathbf{w}}_j}(S', A') - Q_{\mathbf{w}_i}(S, A)) \nabla Q_{\mathbf{w}_i}(S, A)$$

  if $t \equiv 0 \pmod{N_d}$ **then**
    Sample $\epsilon \sim \mathcal{N}(0, 1)$ for each transition $\langle S, A, S', R\rangle$ in B
    Update policy on B:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha_t \nabla_{\boldsymbol{\theta}} g_{\boldsymbol{\theta}}(\epsilon; S)^\top \nabla_{\tilde{A}} Q_{\mathbf{w}_1}(S, \tilde{A})|_{\tilde{A}=g_{\boldsymbol{\theta}}(\epsilon;S)}$$

    Update target network weights:

$$\bar{\mathbf{w}}_i \leftarrow \tau \mathbf{w}_i + (1-\tau)\bar{\mathbf{w}}_i$$

  end if
end for

Algorithm 5 AC-LR for discrete action spaces

Initialize parameters $\mathbf{w}_1, \mathbf{w}_2, \boldsymbol{\theta}, \mathbf{v}$, $\bar{\mathbf{w}}_1 \leftarrow \mathbf{w}_1$, $\bar{\mathbf{w}}_2 \leftarrow \mathbf{w}_2$, replay buffer $\mathcal{B}$
Obtain initial state S_0
for $t = 1$ to T **do**
  Take action $A_t \sim \pi_{\boldsymbol{\theta}}(\cdot|S_t)$, and observe R_{t+1}, S_{t+1}
  Add $\langle S_t, A_t, S_{t+1}, R_{t+1}\rangle$ to the buffer $\mathcal{B}$
  Sample a mini-batch B from buffer $\mathcal{B}$
  Sample $\tilde{A} \sim \pi_{\boldsymbol{\theta}}(\cdot|S)$ and $A' \sim \pi_{\boldsymbol{\theta}}(\cdot|S')$ for each transition $\langle S, A, S', R\rangle$ in B
  Update critics on B:

$$\mathbf{w}_i \leftarrow \mathbf{w}_i + \alpha_t(R + \gamma \min_{j\in\{1,2\}} Q_{\bar{\mathbf{w}}_j}(S', A') - Q_{\mathbf{w}_i}(S, A))\nabla Q_{\mathbf{w}_i}(S, A)$$
$$\mathbf{v} \leftarrow \mathbf{v} + \alpha_t(Q_{\mathbf{w}_1}(S, \tilde{A}) - V_{\mathbf{v}}(S))\nabla V_{\mathbf{v}}(S)$$

  if $t \equiv 0 \pmod{N_d}$ **then**
    Sample $\tilde{A} \sim \pi_{\boldsymbol{\theta}}(\cdot|S)$, for each transition $\langle S, A, S', R\rangle$ in B
    Update policy on B:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha_t \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\tilde{A}|S)(Q_{\mathbf{w}_1}(S, \tilde{A}) - V_{\mathbf{v}}(S))$$

    Update target network weights:

$$\bar{\mathbf{w}}_i \leftarrow \tau \mathbf{w}_i + (1 - \tau)\bar{\mathbf{w}}_i$$

  end if
end for

Algorithm 6 AC-ST for discrete action spaces

Initialize parameters $\mathbf{w}_1, \mathbf{w}_2, \boldsymbol{\theta}$, $\bar{\mathbf{w}}_1 \leftarrow \mathbf{w}_1$, $\bar{\mathbf{w}}_2 \leftarrow \mathbf{w}_2$, replay buffer $\mathcal{B}$
Obtain initial state S_0
for $t = 1$ to T **do**
  Take action $A_t \sim \pi_{\boldsymbol{\theta}}(\cdot|S_t)$, and observe R_{t+1}, S_{t+1}
  Add $\langle S_t, A_t, S_{t+1}, R_{t+1}\rangle$ to the buffer $\mathcal{B}$
  Sample a mini-batch B from buffer $\mathcal{B}$
  Sample $A' \sim \pi_{\boldsymbol{\theta}}(\cdot|S')$ for each transition $\langle S, A, S', R\rangle$ in B
  Update critics on B:

$$\mathbf{w}_i \leftarrow \mathbf{w}_i + \alpha_t(R + \gamma \min_{j\in\{1,2\}} Q_{\bar{\mathbf{w}}_j}(S', A') - Q_{\mathbf{w}_i}(S, A))\nabla Q_{\mathbf{w}_i}(S, A)$$

  if $t \equiv 0 \pmod{N_d}$ **then**
    Sample $\tilde{A} \sim \pi_{\boldsymbol{\theta}}(\cdot|S)$, for each transition $\langle S, A, S', R\rangle$ in B
    Use the straight-through trick to compute $\tilde{A}_{\boldsymbol{\theta}} = \text{one\_hot}(\tilde{A}) + \pi_{\boldsymbol{\theta}}(\cdot|S) - \pi_{\boldsymbol{\phi}}(\cdot|S)|_{\boldsymbol{\phi}=\boldsymbol{\theta}}$
    Update policy on B:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha_t \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(\cdot|S)^\top \nabla_{\tilde{A}} Q_{\mathbf{w}_1}(S, \tilde{A})|_{\tilde{A}=\tilde{A}_{\boldsymbol{\theta}}}$$

    Update target network weights:

$$\bar{\mathbf{w}}_i \leftarrow \tau \mathbf{w}_i + (1 - \tau)\bar{\mathbf{w}}_i$$

  end if
end for