# Retry Policy Gradients in Continuous Action Spaces

**Anonymous authors**
**Paper under double-blind review**

## Abstract

Retry-based objectives such as pass@K and max@K optimize the best return obtained from multiple sampled trajectories, and recent work has shown that they can promote exploration without explicit exploration bonuses. In discrete action spaces, ReMax was shown to do so by adapting to return uncertainty. In this work, we introduce pathwise derivative estimators for retry objectives and use them to extend ReMax to continuous action spaces. We study the resulting learning dynamics and show that, even with deterministic rewards, ReMax can encourage stochastic exploration by reshaping the policy-gradient landscape. In particular, it alters gradients both in direction, biasing updates toward higher policy entropy, and in magnitude, damping gradients and slowing convergence. We further show that Adam's adaptive normalization can mitigate this damping, depending on its numerical stabilization parameter. Empirically, we instantiate this objective as **ReMax Actor-Critic (ReMAC)**, an off-policy actor–critic algorithm that optimizes the ReMax objective using a pathwise derivative estimator. Our experiments show that ReMAC can promote higher policy entropy without entropy regularization and achieves performance comparable to SAC.

## 1 Introduction

Exploration remains a central challenge in reinforcement learning (RL) (Sutton & Barto, 2018). In continuous control, exploration is typically induced by stochastic policies combined with entropy regularization, as popularized by Soft Actor-Critic (SAC) (Haarnoja et al., 2018; Christodoulou, 2019). Alternatively, noise injection—through either action-space or parameter-space perturbations—is widely used to encourage exploratory behavior (Fujimoto et al., 2018; Lillicrap et al., 2016; Plappert et al., 2018). Recent advances have further improved actor–critic methods through architectural refinements, such as the use of ensemble critics (Chen et al., 2021; Hiraoka et al., 2022).

Recently, retry-based objectives, which maximize the highest return achieved over multiple ($M \in \mathbb{N}$) sampled trajectories, have emerged as a compelling alternative formulation for RL in discrete action spaces (Koyamada et al., 2023; Nishimori et al., 2026; Tang et al., 2025; Walder & Karkhanis, 2025). In large language model (LLM) post-training, directly optimizing pass@K, which evaluates the success of the best response among $K$ generations, significantly improves output diversity (Tang et al., 2025; Chen et al., 2025; Walder & Karkhanis, 2025). Within episodic RL settings, the ReMax objective was introduced to explicitly account for the uncertainty of returns (Koyamada et al., 2023; Nishimori et al., 2026). Prior studies have revealed that ReMax naturally encourages exploration as an adaptive response to return uncertainty (Koyamada et al., 2023; Nishimori et al., 2026). Furthermore, Nishimori et al. (2026) demonstrated that ReMax can maintain high policy entropy by slowing down convergence, even in environments with deterministic rewards. However, these existing analyses are largely restricted to discrete action spaces, leaving the detailed theoretical properties of the ReMax objective incompletely understood.

In this study, we investigate the fundamental properties of the ReMax objective in continuous action spaces. Specifically, we identify intrinsic characteristics promoting stochastic exploration in both the *direction* and *magnitude* of the deterministic ReMax policy gradient when the retry budget satisfies $M > 1$. We demonstrate that when a policy is far from the optimum and exhibits low entropy, the gradient direction naturally acts to increase policy entropy. Moreover, as the retry budget $M$ grows, the gradient norm diminishes near the optimum, which inherently slows convergence and sustains exploratory behavior.

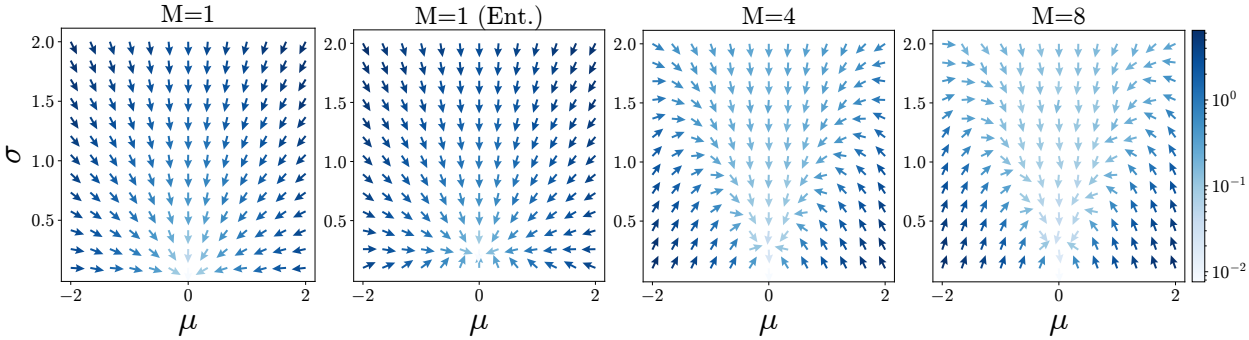



Figure 1: Vector field of normalized ReMax gradients for $M = 1, 4, 8$ over $\mu \in [-2, 2]$ and $\sigma \in [0.1, 2]$. We also plot the entropy-regularized standard RL ($M = 1$ (Ent.)). Arrow color indicates gradient norm (darker is larger). For larger $M$, the gradient increases $\sigma$ when $\mu$ is far from the optimum (bottom edge) and the gradient norm near the optimum shrinks, slowing convergence. Unlike entropy regularization, ReMax always converges to the deterministic optimum $\mu = 0, \sigma = 0$.

Fig. 1 illustrates this effect. It shows the vector field of ReMax gradients for a 1D Gaussian policy $a \sim \mathcal{N}(\mu, \sigma^2)$ with reward $r(a) = -a^2$. For larger $M$, when $\mu$ is far from the optimum and $\sigma$ is small (bottom edges), the gradient points toward increasing $\sigma$, whereas $M = 1$ (standard RL) always decreases $\sigma$. Conversely, when $\mu$ is near the optimum and $\sigma$ is large (top center), the gradient norm is damped (lighter color), slowing convergence. Unlike entropy regularization, ReMax does not change the optimal parameters $\mu = 0$ and $\sigma = 0$, whereas entropy regularization shifts the optimum to $\mu = 0$ and $\sigma > 0$. These properties yield an optimization trajectory in which policy entropy first increases and then gradually decreases, encouraging stochastic exploration before converging to the deterministic optimal policy. A detailed explanation of Fig. 1 is provided in Sec. 5.1, and a theoretical explanation for why these properties emerge is given in Sec. 5.2. Finally, the damping effect can be mitigated by Adam's adaptive gradient normalization (Kingma & Ba, 2015), the de facto standard optimizer in deep learning.

We then instantiate the ReMax objective in an off-policy actor–critic algorithm, which we call **ReM**ax **A**ctor-**C**ritic (**ReMAC**), with a pathwise derivative estimator. Empirically, we show that ReMAC achieves performance comparable to SAC while exhibiting increased entropy for larger $M$.

**Contributions.** Our main contribution is to identify intrinsic properties of the ReMax gradient that encourage stochastic exploration in both direction and magnitude when $M > 1$. We illustrate this effect via a toy vector-field analysis and provide a theoretical explanation in the isotropic Gaussian policy setting. We then instantiate the ReMax objective in an off-policy actor–critic algorithm, ReMAC, which can be implemented with minimal modifications to SAC. Finally, we provide empirical evidence that ReMAC can increase policy entropy with $M > 1$ without entropy regularization and achieves performance comparable to SAC in continuous-control tasks.

## 2 Related Work

Extensive research has addressed the exploration-exploitation tradeoff in reinforcement learning (RL). Existing exploration strategies can be broadly categorized into two paradigms: *stochastic exploration* and *deep exploration*. Stochastic exploration maintains policy diversity to prevent premature convergence to suboptimal behaviors. In contrast, deep exploration seeks temporally coherent behaviors capable of uncovering informative, unvisited regions of the environment. A prevalent method for encouraging stochastic exploration is to introduce an entropy bonus into the policy optimization objective (Schulman et al., 2017; Haarnoja et al., 2018; Christodoulou, 2019; Lee et al., 2025; Nauman et al., 2024). This entropy regularization is a cornerstone of highly successful RL algorithms, such as Soft Actor-Critic (SAC), and remains a standard design choice in modern deep RL architectures (Nauman et al., 2024; Lee et al., 2025).

Conversely, deep exploration is typically driven by intrinsic rewards, uncertainty estimation, or structured noise injection. Standard intrinsic reward techniques include count-based exploration (Bellemare et al., 2016; Ostrovski et al., 2017; Lobel et al., 2023; Tang et al., 2017), prediction-error bonuses (Burda et al., 2019; Pathak et al., 2017), and information gain or curiosity signals (Houthooft et al., 2016; Sukhija et al., 2025; Schmidhuber, 2010). Alternatively, exploration can be promoted by modeling uncertainty in value estimates. For instance, ensemble-based methods approximate posterior sampling using multiple Q-functions (Osband et al., 2016; 2018; 2019; 2023; Zhang & Yao, 2019; Chen et al., 2021; Hiraoka et al., 2022), while related approaches leverage Langevin Monte Carlo updates (Ishfaq et al., 2024). Finally, structured noise injection, whether in the action space or parameter space, offers another mechanism for temporally extended exploration (Fujimoto et al., 2018; Lillicrap et al., 2016; Plappert et al., 2018; Fortunato et al., 2018).

Retry-based objectives offer a distinct paradigm for exploration. Rather than augmenting the reward function or injecting explicit noise into the policy, these objectives evaluate the maximum return achieved across multiple sampled trajectories. Prior work demonstrates that retry objectives successfully induce exploratory behavior in discrete action spaces (Koyamada et al., 2023; Nishimori et al., 2026; Walder & Karkhanis, 2025; Chen et al., 2025; Tang et al., 2025; Hamid et al., 2026). In the context of large language model (LLM) post-training, several recent studies optimize pass@K, or its generalization, max@K, to enhance the diversity of generated outputs (Tang et al., 2025; Chen et al., 2025; Walder & Karkhanis, 2025).

Within the retry-based framework, Koyamada et al. (2023) introduced a policy-gradient estimator for ReMax that relies on a resettable simulator. Later, Nishimori et al. (2026) developed a Q-function-based formulation enabling analytical gradient computation in discrete action spaces, demonstrating that ReMax can foster both stochastic and deep exploration. Specifically, deep exploration emerges when ReMax is paired with posterior sampling; notably, Tong et al. (2026) provided a finite-time regret analysis for this combination under a two-retry setting ($M = 2$). Conversely, stochastic exploration in ReMax has primarily been illustrated through a toy deterministic-reward bandit experiment, where it was observed that ReMax reshapes the gradient landscape and slows convergence, thereby encouraging exploration. However, this prior analysis remains restricted to discrete action spaces and lacks a rigorous, detailed examination of the gradient landscape.

In this paper, we investigate the emergence of *stochastic* exploration from ReMax in continuous action spaces with deterministic rewards, leaving the study of deep exploration for future work. Unlike the pass@K literature, which assumes explicit retries for a given prompt and directly observes outcomes, our setting deals with episodic returns where retries must be emulated via function approximation. Furthermore, while pass@K studies typically rely on sample-based REINFORCE-style estimators, we adopt a pathwise gradient estimator (Kingma & Welling, 2014; Parmas & Sugiyama, 2021; Jankowiak & Obermeyer, 2018; Rezende et al., 2014). This continuous-space pathwise estimator is not directly applicable to standard LLM post-training, which relies on non-differentiable, rule-based reward verifiers. Crucially, unlike traditional entropy regularization, ReMax alters the evaluation of rewards without introducing an auxiliary entropy bonus. Consequently, it naturally avoids bonus-induced bias at the target optimum and eliminates the need to decay bonus coefficients.

## 3 Preliminaries

**Reinforcement learning (RL).** We consider the reinforcement learning (RL) problem (Sutton & Barto, 2018), formulated as a Markov decision process (MDP), $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, p_0, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A} \subseteq \mathbb{R}^d$ is the continuous action space with dimension $d$, $\mathcal{P}$ is the transition dynamics, $p_0$ is the initial state distribution, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. The goal is to train a policy $\pi(\cdot \mid s)$ that maximizes the expected discounted reward $\mathbb{E}_\pi [R_0]$, where $R_0 = \sum_{h=0}^{\infty} \gamma^h r(s_h, a_h)$ is the discounted return and $\mathbb{E}_\pi [\cdot]$ denotes the expectation over the trajectory $(s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$ under policy $\pi$. We define the value function of the policy $\pi$ as $V^\pi(s) = \mathbb{E}_\pi [R_0 \mid s_0 = s]$ and the Q-function as $Q^\pi(s, a) = \mathbb{E}_\pi [R_0 \mid s_0 = s, a_0 = a]$.

## 4 ReMax Objective and its Gradient Estimation

This section introduces the ReMax objective and describes how we estimate its policy gradient.

### 4.1 ReMax Objective

Optimizing returns over multiple trajectories can promote exploration without explicit exploration bonuses (Walder & Karkhanis, 2025; Nishimori et al., 2026; Koyamada et al., 2023). In episodic RL, directly optimizing such retry-based trajectory objectives often assumes the ability to reset the environment to arbitrary states (Koyamada et al., 2023), which is typically infeasible in continuous-control tasks. We therefore adopt the Q-function-based deterministic ReMax formulation (Nishimori et al., 2026). Given a Q-function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, the (deterministic) ReMax objective is defined as

$$\mathcal{J}^M(\pi, Q) := \mathbb{E}_{s\sim p_0}\left[\mathcal{J}^M(\pi, Q, s)\right], \quad \text{where} \quad \mathcal{J}^M(\pi, Q, s) := \mathbb{E}_{a_{1:M} \overset{\text{i.i.d.}}{\sim} \pi(\cdot|s)}\left[\max_{m=1:M} Q(s, a_m)\right] \tag{1}$$

where $M \in \mathbb{N}$ is the retry budget and $\pi$ is the policy. When $M = 1$ and $Q = Q^\pi$, this objective $\mathcal{J}^1(\pi, Q^\pi)$ reduces to the standard RL objective.

Importantly, the ReMax objective preserves the deterministic optimal policy. Since an MDP admits an optimal deterministic policy under standard conditions (Sutton & Barto, 2018), assume that there exists a deterministic optimal policy $\pi^*$ satisfying $Q^{\pi^*} = Q^*$ and $\pi^*(s) \in \arg\max_{a\in\mathcal{A}} Q^*(s, a)$ for every state $s \in \mathcal{S}$. Then $\pi^* \in \arg\max_\pi \mathcal{J}^M(\pi, Q^\pi, s)$ for any finite $M \geq 1$. Indeed, for any state $s$ and policy $\pi$,

$$\mathbb{E}_{a_{1:M} \overset{\text{i.i.d.}}{\sim} \pi(\cdot|s)}\left[\max_{m=1:M} Q^\pi(s, a_m)\right] \leq \sup_a Q^*(s, a) = \mathbb{E}_{a_{1:M} \overset{\text{i.i.d.}}{\sim} \pi^*(\cdot|s)}\left[\max_{m=1:M} Q^*(s, a_m)\right],$$

where $Q^*(s, a) = \sup_\pi Q^\pi(s, a)$. Therefore, we have $\mathcal{J}^M(\pi, Q^\pi) \leq \mathcal{J}^M(\pi^*, Q^{\pi^*})$. In contrast, entropy-regularized objectives shift the optimum toward higher-entropy policies and hence bias the optimum.

### 4.2 Gradient Estimation of the ReMax Objective

We consider optimizing Eq. (1) using policy gradients. Nishimori et al. (2026) showed that in discrete action spaces, both the objective and its gradient can be computed exactly. This approach does not extend to continuous action spaces because it would require sorting Q-values over an uncountable action set. We therefore use sample-based estimators (Parmas et al., 2018; Parmas, 2018). Let $\pi_\theta$ be a policy parameterized by $\theta \in \mathbb{R}^p$. For a state $s$, we sample $a_{1:B} \overset{\text{i.i.d.}}{\sim} \pi_\theta(\cdot \mid s)$ with $B \geq M$ and compute $q_i = Q(s, a_i)$. In practice, $Q$ is approximated by a neural network $Q_\phi$ with parameters $\phi$. We consider two gradient estimators.

**Likelihood ratio (LR) gradient estimator.** The likelihood-ratio estimator approximates the gradient of $\mathbb{E}_{a\sim\pi_\theta}[f(a)]$ ($f : \mathcal{A} \to \mathbb{R}$) as $\mathbb{E}_{a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a \mid s) f(a)]$, also known as the REINFORCE estimator (Williams, 1992). Because Eq. (1) involves a maximum over $M$ coupled action samples, the single-sample LR form above is not directly applicable as written. Nevertheless, Walder & Karkhanis (2025) derived an LR-style estimator for deterministic ReMax: $\hat{g} = \frac{1}{B}\sum_{i=1}^B \nabla_\theta \log \pi_\theta(a_i \mid s)\, \hat{A}_i$, where $\hat{A}_i$ is an advantage estimate. We refer the reader to Walder & Karkhanis (2025) for details.

**Reparameterization (RP) gradient estimator.** Alternatively, we can estimate the gradient via the reparameterization trick (Kingma & Welling, 2014). Assume $\pi_\theta$ is Gaussian with mean $\mu$ and standard deviation $\sigma$ ($\theta = (\mu, \sigma)$). Actions are sampled as $a_i(\mu, \sigma) = \mu + \sigma\varepsilon_i$, where $\varepsilon_i \sim \mathcal{N}(0, 1)$. Thus $\frac{\partial a_i}{\partial \mu} = 1$ and $\frac{\partial a_i}{\partial \sigma} = \varepsilon_i$. For higher-dimensional actions, see Rezende et al. (2014). When $Q$ is differentiable, we can compute $\nabla_{a_i} Q(s, a_i)$ and backpropagate through $q_i$. Walder & Karkhanis (2025) proposed an unbiased estimator of Eq. (1) (for the given $Q$) from $q_{1:B}$ (sorted in ascending order with ties broken randomly):

$$\rho^M(q_{1:B}) := \frac{1}{\binom{B}{M}} \sum_{i=1}^B w_i q_i, \tag{2}$$

where $w_i = \binom{i-1}{M-1}$ counts the number of size-$M$ subsets for which the $i$-th sorted action is the best; the factor $\binom{B}{M}^{-1}$ converts this count into a probability. Because $\rho^M$ is linear in $q_{1:B}$, $\nabla_\theta \rho^M(q_{1:B})$ can be obtained by automatic differentiation. Walder & Karkhanis (2025) did not explore this RP estimator because their focus was LLM post-training, where rewards are typically non-differentiable.

**Our choice.** The two estimators have different trade-offs. LR gradients can remain unbiased with raw returns, which suits on-policy learning, but they often suffer from high variance (Greensmith et al., 2004). RP gradients require a differentiable Q-function but typically (but not always) yield lower-variance estimates (Parmas et al., 2018). In continuous control, off-policy actor–critic methods (Degris et al., 2012; Haarnoja et al., 2018; Fujimoto et al., 2018) are widely used for their sample efficiency. These methods rely on replay buffers that store transitions from past behavior policies and update both the Q-function and policy off-policy. As a result, obtaining raw-return targets for arbitrary state–action pairs is difficult, which weakens the main advantage of LR estimators. Thus, in this setting, LR offers limited benefits while retaining high variance, whereas RP estimators are standard and effective. We therefore adopt the RP estimator.

## 5 Gradient Analysis

We analyze how the ReMax objective promotes exploration through policy gradients. We show that ReMax promotes exploration even with deterministic rewards by reshaping the gradient landscape so that policy entropy remains higher along the optimization path. When the policy is far from the optimum and has low entropy, the gradient increases entropy through its direction; near the optimum, the gradient norm is damped through its magnitude. We provide a theoretical explanation for these properties (Sec. 5.2) and show that Adam's normalization can mitigate the damping effect (Sec. 5.3), offering practical guidance for continuous control. To illustrate this effect, we consider a one-dimensional Gaussian policy with mean $\mu \in \mathbb{R}$ and standard deviation $\sigma > 0$, i.e., $a \sim \mathcal{N}(\mu, \sigma^2)$. We use the quadratic reward $r(a) = -a^2$.

### 5.1 Vector field analysis

In Fig. 1, we visualize the vector field of the RP gradient estimator (Sec. 4.2) with batch size $B = 16$ over $\mu \in [-2, 2]$ and $\sigma \in [0.1, 2]$, for $M \in \{1, 4, 8\}$. For reference, we also plot the entropy-regularized objective for $M = 1$, $r(a) = -a^2 + \alpha \frac{1}{2} \log(2\pi\sigma^2)$ with $\alpha = 0.5$. For $M = 1$, the gradient with respect to $\sigma$ is always negative. For $M > 1$, ReMax exhibits two exploration-promoting properties:

- **Entropy Increase (direction):** When the mean is far from the optimum and $\sigma$ is small, the gradient increases $\sigma$, inducing stochastic exploration.
- **Gradient Damping (magnitude):** As $M$ increases, the gradient norm near the optimal mean (small $\mu$ and large $\sigma$) decreases, slowing convergence and helping sustain larger entropy.

More details are provided in App. B.1. ReMax converges to the deterministic optimum $(\mu, \sigma) = (0, 0)$, whereas entropy regularization converges to $(0, 0.5)$. Together, these effects keep entropy high during optimization: it first increases due to the directional effect and then decreases gradually due to damping.

### 5.2 Theoretical Explanation

We now provide intuition for how ReMax with $M > 1$ shapes exploration dynamics in high-dimensional action spaces. The vector-field analysis in Sec. 5.1 is one-dimensional, but the underlying mechanism is not specific to one dimension. When the policy scale is small, the selected action is determined by the projection of Gaussian perturbations onto the local improvement direction. Thus, ReMax favors samples that move the action toward lower cost, yielding a positive scale gradient when the current policy mean is not stationary. Near the optimum, selecting the best action among $M$ samples makes the selected action close to the optimum, which damps the gradient magnitude. These two effects correspond to the entropy-increase and gradient-damping behaviors observed in the vector field.

**Setup and assumptions.** Let $\mu \in \mathbb{R}^d$ and $\sigma > 0$ denote the mean and scalar scale of an isotropic Gaussian policy. For $m = 1, \ldots, M$, let

$$A_m = \mu + \sigma\varepsilon_m, \qquad \varepsilon_m \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I_d).$$

We consider deterministic rewards of the form $r(a) = -c(a)$ with $c : \mathbb{R}^d \to \mathbb{R}$. For $M \geq 1$, define $J^M(\mu, \sigma) = \mathbb{E}\left[\max_{m=1,\ldots,M} -c(A_m)\right]$. Equivalently, $J^M(\mu, \sigma) = -\mathbb{E}\left[\min_{m=1,\ldots,M} c(A_m)\right]$. Let $m^\star \in$

$\arg\min_{m=1,\dots,M} c(A_m)$ denote the selected action. These assumptions are introduced for analytical tractability rather than to model deep RL environments globally, allowing us to study how ReMax shapes the local gradient landscape.

**Assumption 1** (Smoothness)**.** $c$ is differentiable and $L$-smooth, i.e., $\|\nabla c(a) - \nabla c(b)\| \leq L\|a - b\|$ for all $a, b \in \mathbb{R}^d$.

**Assumption 2** (Non-degenerate selection)**.** There exists $\bar{\sigma} > 0$ such that, for every $\sigma \in (0, \bar{\sigma})$, the random variable $c(\mu + \sigma\varepsilon)$ has no atoms for $\varepsilon \sim \mathcal{N}(0, I_d)$.

This assumption ensures that the selected sample is almost surely unique. It holds, for example, when every level set $\{a : c(a) = t\}$ has Lebesgue measure zero. As shown in App. A, this condition is automatically satisfied under Assumptions 1, 3, and 4.

**Assumption 3** (Centered optimum)**.** The cost is normalized so that $c(0) = 0$ and $\nabla c(0) = 0$.

**Assumption 4** (Strong convexity)**.** $c$ is $\lambda$-strongly convex, i.e.,

$$c(b) \geq c(a) + \nabla c(a)^\top (b - a) + \frac{\lambda}{2}\|b - a\|^2$$

for all $a, b \in \mathbb{R}^d$.

**Assumption 5** (Second-order regularity)**.** $c$ is twice continuously differentiable.

These assumptions are separated to make clear which part of the analysis relies on which regularity condition. The entropy-increase result relies only on local smoothness and non-degenerate sample selection. The entropy-decrease result for $M = 1$ additionally uses second-order regularity and strong convexity. The gradient-damping result uses smoothness, strong convexity, and the normalization that the optimum is centered at the origin. These assumptions are not intended to be the weakest possible conditions. Rather, they provide a clean first setting in which the two mechanisms observed in the vector field can be isolated and proved rigorously. In particular, smoothness and strong convexity are standard assumptions in first-order optimization theory (Nesterov, 2013; Bubeck, 2015), while the non-degenerate selection condition is a technical condition used to handle the best-of-$M$ operator. Thus, although these assumptions do not fully reflect practical deep-RL settings with neural networks, they are appropriate for a first theoretical analysis of the local mechanisms induced by ReMax.

**Entropy increase effect for $M > 1$.** We first show that ReMax can increase policy entropy when the current mean is not stationary. This result does not require global convexity. It only uses the local first-order geometry of the cost around the current policy mean.

**Proposition 1** (Entropy Increase Effect)**.** Assume Assumptions 1 and 2. Let $M \geq 2$ and suppose that $\nabla c(\mu) \neq 0$. Then, there exists $\sigma_0 > 0$ such that

$$\partial_\sigma J^M(\mu, \sigma) > 0 \qquad \text{for all } \sigma \in (0, \sigma_0).$$

The proof is given in App. A.1. The intuition is that ReMax can exploit upward fluctuations in reward across multiple samples. Since $r(a) = -c(a)$, this is equivalent to selecting a sample with a downward fluctuation in cost. Let $g = \nabla c(\mu)$. For small $\sigma$, the cost of each sampled action is locally approximated as $c(\mu+\sigma\varepsilon_m) = c(\mu)+\sigma g^\top \varepsilon_m+O(\sigma^2\|\varepsilon_m\|^2)$. Thus, to first order, the selected action is the one with the smallest projection $g^\top \varepsilon_m$, or equivalently the largest reward fluctuation $-g^\top \varepsilon_m$. For a single sample, this fluctuation has zero mean. With $M \geq 2$, however, taking the maximum over samples creates a positive expected upward fluctuation, since $\mathbb{E}[\max_m(-g^\top \varepsilon_m)] = -\mathbb{E}[\min_m g^\top \varepsilon_m] = \|\nabla c(\mu)\|\mathbb{E}[\max_m Z_m] > 0$. Therefore, when $\sigma$ is small, increasing the policy scale makes these favorable fluctuations more available, yielding a positive scale gradient. This is the high-dimensional analogue of the entropy-increase direction observed in Sec. 5.1.

**Entropy decrease for $M = 1$.** The previous effect contrasts with the standard single-sample objective. When $M = 1$, increasing the policy scale only injects noise into the cost, and the scale gradient is always negative under strong convexity.

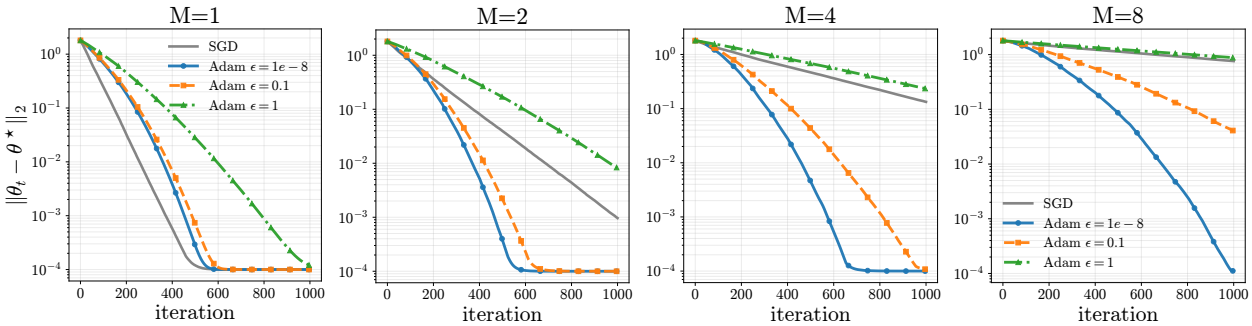



Figure 2: The distance to the optimal parameter $(\mu, \sigma) = (0, 0)$ for Adam with different $\varepsilon$ values and for SGD. As the vector field indicates, the convergence speed slows as $M$ increases in every case. However, with smaller $\varepsilon$ the convergence near the optimum remains fast, whereas larger $\varepsilon$ aligns the trajectory with the SGD path that follows the vector field.

**Proposition 2** (Entropy Decrease for $M = 1$)**.** Assume Assumptions 1, 4, and 5. For $M = 1$, $\mu \in \mathbb{R}^d$, and $\sigma > 0$,

$$\partial_\sigma J^1(\mu, \sigma) \leq -\lambda d\sigma < 0.$$

The proof is given in App. A.2. This result explains why the $M = 1$ vector field consistently shrinks the policy scale. Without best-of-$M$ selection, the objective directly penalizes variance around the current mean.

**Gradient damping effect for $M > 1$.** We next bound the gradient magnitudes by the expected distance of the best sampled action to the optimum.

**Proposition 3** (Gradient Damping)**.** Assume Assumptions 1, 3, and 4. Let $M \geq 1$ and $\sigma > 0$. Then, the gradients exist and satisfy

$$\|\nabla_\mu J^M(\mu, \sigma)\| \leq L\sqrt{\frac{L}{\lambda}}\, \mathbb{E}\left[\min_{m=1,\ldots,M} \|A_m\|\right],$$

and

$$|\partial_\sigma J^M(\mu, \sigma)| \leq L\sqrt{\frac{L}{\lambda}}\, \mathbb{E}\left[\min_{m=1,\ldots,M} \|A_m\| \|\varepsilon_{m^\star}\|\right].$$

The proof is given in App. A.3. These bounds capture the "forgiveness" of the minimum operator. In standard RL ($M = 1$), large $\sigma$ near the optimum produces poor actions and strong negative gradients that rapidly shrink $\sigma$. In contrast, ReMax evaluates only the best action. When $\mu$ is near 0, increasing $M$ raises the probability that at least one action lies close to the optimum, reducing the selected deviation and damping gradients. When the policy mean is far from stationary and $\sigma$ is small, the projection effect in Proposition 1 dominates and increases the policy scale. Together, these effects can keep entropy high during optimization: it first increases due to the directional effect and then decreases gradually due to damping.

### 5.3 Effect of Adam's normalization

Although the vector-field analysis suggests that ReMax with $M > 1$ promotes stochastic exploration, the actual optimization trajectory need not follow the raw vector field. With Adam (Kingma & Ba, 2015), adaptive normalization can counteract gradient damping, leading to faster convergence and faster entropy collapse. Adam updates parameters as $\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}$. If $\varepsilon$ is small, $\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \approx \frac{\hat{m}_t}{\sqrt{\hat{v}_t}}$, effectively normalizing gradients and reducing damping. To illustrate this effect, we optimize $a \sim \mathcal{N}(\mu, \sigma^2)$ with $r(a) = -a^2$ using Adam with $\varepsilon \in \{10^{-8}, 10^{-1}, 1.0\}$. We set $\alpha = 0.01$, $B = 64$, $T = 1000$, and $M \in \{1, 2, 4, 8\}$, starting from $(\mu, \sigma) = (-1.5, 1.0)$. Fig. 2 shows the distance to $(0, 0)$ for Adam and SGD. As expected, convergence slows as $M$ increases. However, smaller $\varepsilon$ mitigates damping: with $M = 4, 8$, $\varepsilon = 10^{-8}$ converges much faster than SGD, whereas $\varepsilon = 1.0$ aligns with SGD. These results suggest a practical guideline. Adam benefits from the directional entropy-increase effect, but with the default $\varepsilon = 10^{-8}$, damping is weakened. Increasing $\varepsilon$

**Algorithm 1** ReMAC

**Require:** Retry parameter $M$, action batch size $B$, discount $\gamma$, learning rates $\alpha, \beta$, target critic update rate $\tau$.

Initialize policy $\pi_\theta$, critics $Q_{\phi_1}, Q_{\phi_2}$, target critics $Q_{\bar{\phi}_1}, Q_{\bar{\phi}_2}$, and replay buffer $\mathcal{D}$
**for** each iteration $t$ **do**
  **for** each environment step **do**
    $a \sim \pi_\theta(\cdot|s),\ (r, s') \sim \mathcal{P}(s, a)$ ▷ interact with environment
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s, a, r, s')\}$ ▷ store transition
  **end for**
  **for** each gradient step **do**
    $(s, a, r, s') \sim \mathcal{D},\ a' \sim \pi_\theta(\cdot|s')$ ▷ sample minibatch and next action
    $y \leftarrow r + \gamma \min(Q_{\bar{\phi}_1}(s', a'), Q_{\bar{\phi}_2}(s', a'))$ ▷ target value
    $\phi_j \leftarrow \phi_j - \alpha \nabla_{\phi_j}(Q_{\phi_j}(s, a) - y)^2,\ j \in \{1, 2\}$ ▷ update critics
    $a_{1:B} \sim \pi_\theta(\cdot|s),\ q_i \leftarrow \min_{j=1,2} Q_{\phi_j}(s, a_i) \forall i \in \{1, 2, \ldots, B\}$ ▷ sample actions and evaluate Q
    $\theta \leftarrow \theta + \beta \nabla_\theta \rho^M(q_{1:B})$ ▷ update policy
    $\bar{\phi}_j \leftarrow \tau \phi_j + (1 - \tau)\bar{\phi}_j,\ j \in \{1, 2\}$ ▷ update target critics
  **end for**
**end for**

restores damping but reduces the effective step size. Jointly tuning $\varepsilon$ and the learning rate may therefore better control optimization dynamics during training. More details are provided in App. B.2.

## 6 ReMax Actor-Critic Algorithm

Having identified the exploration-promoting properties of ReMax, we instantiate it in a simple off-policy actor–critic algorithm with a critic $Q_\phi$ and a stochastic policy $\pi_\theta$. The critic is trained by temporal-difference regression on transitions sampled from a replay buffer $\mathcal{D}$, while the actor is optimized using ReMax computed from batches of action samples.

**Critic update.** Given a transition $(s, a, r, s') \sim \mathcal{D}$, we define the target as $y = r + \gamma Q_{\bar{\phi}}(s', a')$, where $a' \sim \pi_\theta(\cdot \mid s')$ and $Q_{\bar{\phi}}$ is a target critic. The critic is updated by minimizing the squared Bellman error:

$$L_Q(\phi) := \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}}\big[(Q_\phi(s, a) - y)^2\big]. \tag{3}$$

**Actor update.** For policy optimization, let $\pi_\theta(a \mid s) = \mathcal{N}(\mu_\theta(s), \sigma_\theta(s))$, where $\mu_\theta(s)$ and $\sigma_\theta(s)$ are neural-network outputs. We sample $\varepsilon_{1:B} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 1)$ and construct actions as $a_\theta(\varepsilon_{1:B}) = \mu_\theta(s) + \sigma_\theta(s)\varepsilon_{1:B}$. We then evaluate the Q-values as $Q_\phi(s, a_\theta(\varepsilon_{1:B})) := (Q_\phi(s, a_\theta(\varepsilon_1)), \ldots, Q_\phi(s, a_\theta(\varepsilon_B)))$. The actor loss is

$$L^M_{\text{actor}}(\theta) := -\mathbb{E}_{s \sim \mathcal{D}, \varepsilon_{1:B} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1)}\big[\rho^M(Q_\phi(s, a_\theta(\varepsilon_{1:B})))\big]. \tag{4}$$

We optimize the policy parameters by automatic differentiation of $L^M_{\text{actor}}(\theta)$. We call this algorithm **ReM**ax **A**ctor-**C**ritic (**ReMAC**); the full procedure is summarized in Alg. 1.

**Remarks on the implementation.** To reduce overestimation, we use clipped double Q-learning (Haarnoja et al., 2018; Fujimoto et al., 2018) with two critics and take the minimum in both the target and actor loss. In practice, we obtain ReMAC by removing the "soft" components of Soft Actor-Critic (SAC) (Haarnoja et al., 2018)—namely, entropy bonuses in the critic targets and temperature tuning—and replacing the actor loss with our ReMax loss. This is a minimal off-policy actor–critic instantiation of ReMax, and many extensions are possible. One promising direction is to maintain a posterior over the Q-function (Osband et al., 2016; 2018; 2019), making return uncertainty explicit as in ReMax (Koyamada et al., 2023; Nishimori et al., 2026). This may enable *deeper*, more structured exploration, which is crucial in sparse-reward settings.

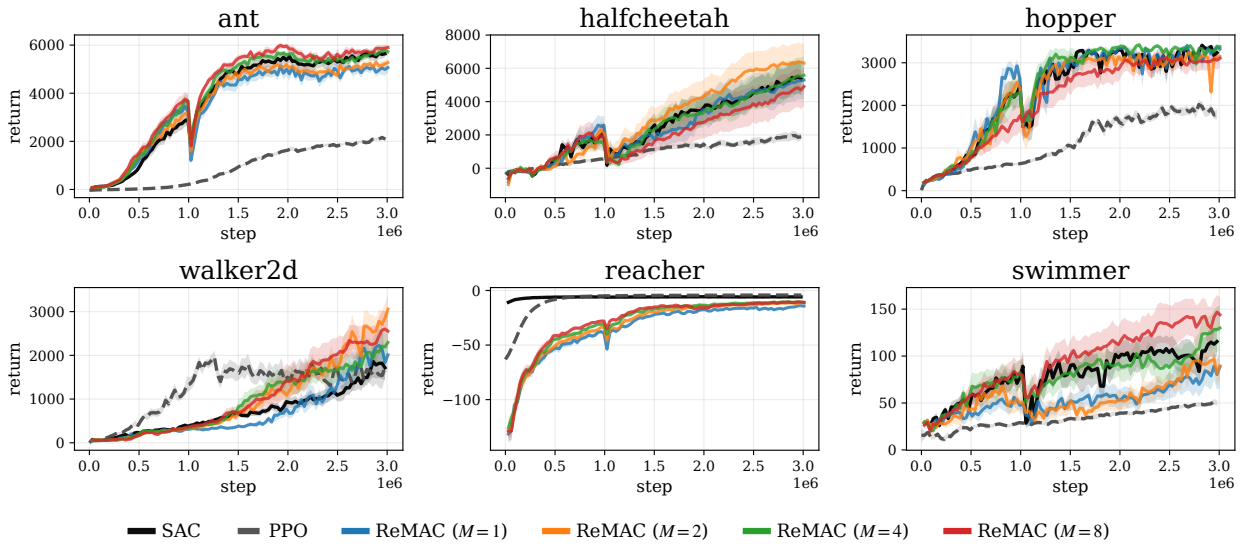



Figure 3: Average return of SAC, PPO, and ReMAC with different $M$ under Adam's default $\varepsilon = 10^{-8}$. The shaded area shows the standard error across 10 random seeds. ReMAC with $M > 1$ achieves performance comparable to SAC.

## 7 Experiments

The primary goal of our empirical evaluation is *not* to establish ReMAC as a new state-of-the-art exploration method for continuous control. Instead, we treat this section as a proof-of-concept evaluation and ask the following two questions.

1. Does ReMAC achieve performance comparable to SAC in continuous-control tasks?
2. Do the entropy-related behaviors predicted by our analysis—namely, higher policy entropy for larger $M$ in ReMax—appear in practical deep RL training?

**Setup.** We considered six continuous-control tasks from Brax (Freeman et al., 2021): Ant, HalfCheetah, Hopper, Reacher, Swimmer, and Walker2d. Agents were trained for 3 million environment steps. We report the average return over 128 evaluation episodes, together with the standard error across 10 random seeds.

**Baselines.** We used SAC (Haarnoja et al., 2018) and PPO (Schulman et al., 2017) as baselines to verify that ReMAC maintains competitive performance. Our implementation built on `rejax` (Liesen et al., 2024), which provides vectorized training pipelines for SAC and PPO with tuned configurations for each Brax task; we adopted these configurations for the baselines. We evaluate ReMAC with retry budgets $M \in \{1, 2, 4, 8\}$. For $M = 1, 2, 4$, we set the action-sample batch size to $B = 8$, and for $M = 8$, we set $B = 16$. For the main comparison, we used Adam's default $\varepsilon = 10^{-8}$. We swept the learning rate over $\{10^{-4}, 2 \times 10^{-4}, 3 \times 10^{-4}, 5 \times 10^{-4}, 10^{-3}\}$ with 3 random seeds and selected a value that performed consistently well across environments and $M$ for fixed $\varepsilon = 10^{-8}$. As discussed in Sec. 5.3, a small value of Adam's $\varepsilon$ can mitigate the gradient-damping effect by normalizing gradient magnitudes. Thus, the main results with the default $\varepsilon = 10^{-8}$ primarily examined whether the directional entropy-increase effect of ReMax appears in practical actor–critic training. To study the role of Adam's $\varepsilon$ in the damping effect, we additionally varied $\varepsilon \in \{10^{-8}, 10^{-2}, 10^{-1}, 1.0\}$ for all $M$; the results are provided in App. C.2. We also tested the sensitivity of the results to the action-sample batch size $B \in \{8, 16\}$, with results reported in App. C.2.

**Main results.** Fig. 3 reports the average return with Adam's default $\varepsilon = 10^{-8}$. ReMAC with $M > 1$ achieved returns comparable to those of SAC and generally performed better than PPO in these experiments.

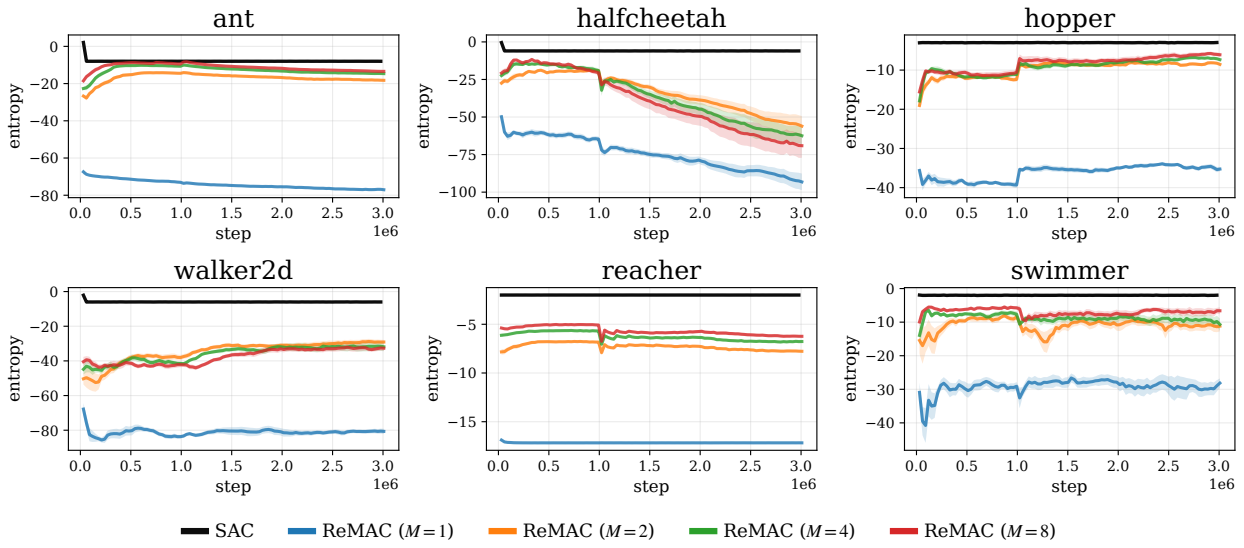



Figure 4: Average policy entropy of SAC and ReMAC with different $M$ under Adam's default $\varepsilon = 10^{-8}$. The shaded area shows the standard error across 10 random seeds. $M > 1$ generally yields higher entropy than $M = 1$, and entropy tends to increase with $M$ (most clearly in Ant).

In several environments, including Ant, Reacher, Swimmer, and Walker2d, larger retry budgets ($M = 4, 8$) achieved higher average return than $M = 1$.

Fig. 4 reports the average policy entropy for different $M$. ReMAC with $M > 1$ yielded higher entropy than $M = 1$, and entropy generally increased with $M$, most clearly in Ant and Swimmer. This behavior is consistent with the ReMax property identified in Sec. 5.1, where larger $M$ maintains higher entropy. In some environments, such as HalfCheetah, larger $M$ led to lower entropy late in training; for example, $M = 8$ became lower than $M = 4$ and $M = 2$. We hypothesized that this behavior was caused by Adam's normalization counteracting the damping effect of ReMax, as discussed in Sec. 5.3. A more detailed analysis of the effect of $\varepsilon$ is provided in App. C.2, where increasing $\varepsilon$ tended to yield higher entropy, consistent with the damping interpretation. Finally, although ReMAC with $M > 1$ increased entropy relative to $M = 1$, its entropy remained lower than SAC, as expected because SAC's critic incorporated future entropy (Haarnoja et al., 2018). From a practical perspective, these environment-step comparisons should be interpreted together with the additional computational cost of ReMAC, which arose from the extra Q-network evaluations for $B$ action samples per state; see App. C for details.

## 8 Conclusion

In this study, we extended the ReMax objective to continuous action spaces. We showed that ReMax reshapes the gradient landscape, guiding the policy toward higher entropy during optimization. We first demonstrated this effect via a toy vector-field analysis and then provided a theoretical explanation. On the practical side, we instantiated ReMax in an off-policy actor–critic algorithm, ReMax Actor-Critic (ReMAC), which can be implemented with minimal modifications to SAC. Empirically, we showed that ReMAC can promote higher policy entropy without entropy regularization and achieves performance comparable to SAC.

**Limitations and future work.** Our study has both theoretical and practical limitations. Theoretically, our analysis assumes strong convexity and smoothness of the reward function, which may not hold in practice. Although the goal of our analysis is to build intuition, extending it to more general settings would help us better understand the properties of ReMax. Practically, our implementation is minimal and focuses on stochastic exploration, as discussed in Sec. 2; it could be improved by incorporating additional exploration mechanisms. A particularly well-aligned direction is to model the posterior over the Q-function (Osband

et al., 2016; 2018), since ReMax explicitly considers return uncertainty. As suggested by Nishimori et al. (2026); Tong et al. (2026), this direction may enable deeper, structured exploration, which is important in sparse-reward settings. We hope that our analysis of ReMax's gradient properties and ReMAC's comparable performance to SAC with minimal modifications will provide a foundation for future work.

## References

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying Count-Based Exploration and Intrinsic Motivation. In *Advances in Neural Information Processing Systems*, 2016.

Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and trends in Machine Learning*, 8(3-4):231–357, 2015.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by Random Network Distillation. In *International Conference on Learning Representations*, 2019.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith W. Ross. Randomized ensembled double Q-learning: Learning fast without a model. In *International Conference on Learning Representations*, 2021.

Zhipeng Chen, Xiaobo Qin, Youbin Wu, Yue Ling, Qinghao Ye, Wayne Xin Zhao, and Guang Shi. Pass@k training for adaptively balancing exploration and exploitation of large reasoning models. *arXiv preprint arXiv:2508.10751*, 2025.

Petros Christodoulou. Soft actor-critic for discrete action settings. *arXiv preprint arXiv:1910.07207*, 2019.

Thomas Degris, Martha White, and Richard S Sutton. Off-policy actor-critic. *International Conference on Machine Learning*, 2012.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Matteo Hessel, Ian Osband, Alex Graves, Volodymyr Mnih, Rémi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy networks for exploration. In *International Conference on Learning Representations*, 2018.

C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - A differentiable physics engine for large scale rigid body simulation. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2021.

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

Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(Nov):1471–1530, 2004.

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*, 2018.

Jubayer Ibn Hamid, Ifdita Hasan Orney, Ellen Xu, Chelsea Finn, and Dorsa Sadigh. Polychromic objectives for reinforcement learning. In *International Conference on Learning Representations*, 2026.

Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout q-functions for doubly efficient reinforcement learning. In *International Conference on Learning Representations*, 2022.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. VIME: Variational Information Maximizing Exploration. In *Advances in Neural Information Processing Systems*, 2016.

Haque Ishfaq, Yixin Tan, Yu Yang, Qingfeng Lan, Jianfeng Lu, A. Rupam Mahmood, Doina Precup, and Pan Xu. More efficient randomized exploration for reinforcement learning via approximate sampling. *Reinforcement Learning Journal*, 3:1211–1235, 2024.

Martin Jankowiak and Fritz Obermeyer. Pathwise derivatives beyond the reparameterization trick. In *International Conference on Machine Learning*, pp. 2235–2244. PMLR, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

Sotetsu Koyamada, Paavo Parmas, Tadashi Kozuno, and Shin Ishii. Emergence of exploration in policy gradient reinforcement learning via resetting, 2023. URL https://openreview.net/forum?id=GKsNIC_mQRG.

Hojoon Lee, Dongyoon Hwang, Donghu Kim, Hyunseung Kim, Jun Jet Tai, Kaushik Subramanian, Peter R Wurman, Jaegul Choo, Peter Stone, and Takuma Seno. Simba: Simplicity bias for scaling up parameters in deep reinforcement learning. In *International Conference on Learning Representations*, 2025.

Jarek Liesen, Chris Lu, and Robert Lange. rejax, 2024. URL https://github.com/keraJLi/rejax.

Timothy Paul Lillicrap, Jonathan James Hunt, Alexander Pritzel, Nicolas Manfred Otto Heess, Tom Erez, Yuval Tassa, David Silver, and Daniel Pieter Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

Sam Lobel, Akhil Bagaria, and George Konidaris. Flipping coins to estimate pseudocounts for exploration in reinforcement learning. In *International Conference on Machine Learning*, pp. 22594–22613. PMLR, 2023.

Michal Nauman, Mateusz Ostaszewski, Krzysztof Jankowski, Piotr Miłoś, and Marek Cygan. Bigger, regularized, optimistic: Scaling for compute and sample efficient continuous control. *Advances in neural information processing systems*, 37:113038–113071, 2024.

Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*, volume 87. Springer Science & Business Media, 2013.

Soichiro Nishimori, Paavo Parmas, Sotetsu Koyamada, Tadashi Kozuno, Toshinori Kitamura, Shin Ishii, and Yutaka Matsuo. Emergence of exploration in policy gradient reinforcement learning via retrying. In *Proceedings of the International Conference on Machine Learning*, 2026.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. In *Advances in Neural Information Processing Systems*, volume 29, pp. 4026–4034, 2016.

Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Ian Osband, Benjamin Van Roy, Daniel J Russo, and Zheng Wen. Deep exploration via randomized value functions. *Journal of Machine Learning Research*, 20(124):1–62, 2019.

Ian Osband, Zheng Wen, Seyed Mohammad Asghari, Vikranth Dwaracherla, Morteza Ibrahimi, Xiuyuan Lu, and Benjamin Van Roy. Approximate Thompson sampling via epistemic neural networks. In *Uncertainty in Artificial Intelligence*, pp. 1586–1595. PMLR, 2023.

Georg Ostrovski, Marc G. Bellemare, Aäron van den Oord, and Rémi Munos. Count-based exploration with neural density models. In *International Conference on Machine Learning*, pp. 2721–2730. PMLR, 2017.

Paavo Parmas. Total stochastic gradient algorithms and applications in reinforcement learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Paavo Parmas and Masashi Sugiyama. A unified view of likelihood ratio and reparameterization gradients. In *International Conference on Artificial Intelligence and Statistics*, pp. 4078–4086. PMLR, 2021.

Paavo Parmas, Carl Edward Rasmussen, Jan Peters, and Kenji Doya. PIPPS: Flexible model-based policy search robust to the curse of chaos. In *International Conference on Machine Learning*, pp. 4065–4074. PMLR, 2018.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-Driven Exploration by Self-Supervised Prediction. In *International Conference on Machine Learning*, 2017.

Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. In *International Conference on Learning Representations*, 2018.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pp. 1278–1286. PMLR, 2014.

Jürgen Schmidhuber. Formal Theory of Creativity, Fun, and Intrinsic Motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010.

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

Bhavya Sukhija, Stelian Coros, Andreas Krause, Pieter Abbeel, and Carmelo Sferrazza. MaxInfoRL: Boosting exploration in reinforcement learning through information gain maximization. In *International Conference on Learning Representations*, 2025.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2nd edition, 2018.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. #Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems*, 2017.

Yunhao Tang, Kunhao Zheng, Gabriel Synnaeve, and Remi Munos. Optimizing language models for inference time objectives using reinforcement learning. In *International Conference on Machine Learning*, pp. 59066–59085, 2025.

Bingkui Tong, Junpei Komiyama, Soichiro Nishimori, and Paavo Parmas. Finite-time regret analysis of retry-aware bandits. *arXiv preprint arXiv:2605.20854*, 2026.

Christian Walder and Deep Tejas Karkhanis. Pass@K policy optimization: Solving harder reinforcement learning problems. *Advances in Neural Information Processing Systems*, 38:152416–152445, 2025.

Ronald J Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8(3):229–256, May 1992.

Shangtong Zhang and Hengshuai Yao. Ace: An actor ensemble algorithm for continuous control with tree search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 5789–5796, 2019.

## Appendix Contents

**A Proofs of Propositions in Sec. 5** **14**

A.1 Proof of Proposition 1 . . . . . 15

A.2 Proof of Proposition 2 . . . . . 16

A.3 Proof of Proposition 3 . . . . . 17

**B Details of the gradient analysis** **18**

B.1 Details for the vector field analysis . . . . . 18

B.2 Details for Adam's normalization . . . . . 19

**C Details of the experiments** **19**

C.1 Setup . . . . . 19

C.2 Additional results . . . . . 19

## A Proofs of Propositions in Sec. 5

In this section, we prove the propositions in Sec. 5. Throughout this section, let

$$A_m = \mu + \sigma\varepsilon_m, \qquad \varepsilon_m \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I_d).$$

**Lemma 1** (No ties from non-atomic cost values)**.** Suppose Assumption 2 holds. Then, for every $\sigma \in (0, \bar{\sigma})$,

$$\mathbb{P}\left(\exists i \neq j \text{ such that } c(\mu + \sigma\varepsilon_i) = c(\mu + \sigma\varepsilon_j)\right) = 0.$$

***Proof.*** Fix $\sigma \in (0, \bar{\sigma})$ and define $Y_i = c(\mu + \sigma\varepsilon_i)$. Then $Y_1, \ldots, Y_M$ are independent and identically distributed. By Assumption 2, each $Y_i$ has no atoms. For any $i \neq j$,

$$\mathbb{P}(Y_i = Y_j) = \int_{\mathbb{R}} \mathbb{P}(Y_i = t)\, d\mathbb{P}_{Y_j}(t) = 0.$$

Taking a union bound over the finitely many pairs $(i, j)$ gives

$$\mathbb{P}(\exists i \neq j : Y_i = Y_j) = 0.$$

□

**Lemma 2** (Differentiability of the scale objective)**.** Assume Assumptions 1 and 2. For every $\sigma \in (0, \bar{\sigma})$, $J^M(\mu, \sigma)$ is differentiable with respect to $\sigma$ and

$$\partial_\sigma J^M(\mu, \sigma) = \mathbb{E}\left[-\nabla c(A_{m^\star})^\top \varepsilon_{m^\star}\right],$$

where $m^\star \in \arg\min_{m=1,\ldots,M} c(A_m)$.

***Proof.*** Fix $\sigma \in (0, \bar{\sigma})$. For a fixed realization of $\varepsilon_{1:M}$, define

$$F(s) = \max_{m=1,\ldots,M} -c(\mu + s\varepsilon_m).$$

By Lemma 1, the maximizer is almost surely unique at $s = \sigma$. Hence, by Danskin's theorem,

$$F'(\sigma) = -\nabla c(\mu + \sigma\varepsilon_{m^\star})^\top \varepsilon_{m^\star}$$

for almost every realization.

It remains to justify interchanging differentiation and expectation. Choose $\delta > 0$ such that $[\sigma - \delta, \sigma + \delta] \subset (0, \bar{\sigma})$. For every $s \in [\sigma - \delta, \sigma + \delta]$ and every $m$, Assumption 1 gives

$$\|\nabla c(\mu + s\varepsilon_m)\| \leq \|\nabla c(\mu)\| + L(\sigma + \delta)\|\varepsilon_m\|.$$

Therefore,

$$|F'(s)| \leq \Big(\|\nabla c(\mu)\| + L(\sigma + \delta) \max_m \|\varepsilon_m\|\Big) \max_m \|\varepsilon_m\|.$$

The right-hand side is integrable because $M < \infty$ and Gaussian random variables have finite moments. Dominated convergence yields

$$\partial_\sigma J^M(\mu, \sigma) = \mathbb{E}[F'(\sigma)] = \mathbb{E}\left[-\nabla c(A_{m^\star})^\top \varepsilon_{m^\star}\right].$$

□

### A.1 Proof of Proposition 1

***Proof.*** Let $g = \nabla c(\mu)$. By assumption, $g \neq 0$. Define $B_m = g^\top \varepsilon_m$. Then $B_m \overset{d}{=} \|g\| Z_m$, where $Z_m \sim \mathcal{N}(0, 1)$. Since $g \neq 0$, the random variables $B_1, \ldots, B_M$ are continuous. Thus,

$$j^\star \in \arg\min_{m=1,\ldots,M} B_m$$

is almost surely unique.

We first show that, for almost every realization, $m^\star = j^\star$ for all sufficiently small $\sigma > 0$. By Assumption 1,

$$c(\mu + \sigma\varepsilon_m) = c(\mu) + \sigma g^\top \varepsilon_m + R_m(\sigma),$$

where

$$|R_m(\sigma)| \leq \frac{L}{2}\sigma^2 \|\varepsilon_m\|^2.$$

Fix a realization for which $j^\star$ is unique, and define

$$\Delta = \min_{m \neq j^\star} (B_m - B_{j^\star}) > 0.$$

For any $m \neq j^\star$,

$$c(\mu + \sigma\varepsilon_m) - c(\mu + \sigma\varepsilon_{j^\star}) = \sigma(B_m - B_{j^\star}) + R_m(\sigma) - R_{j^\star}(\sigma).$$

Since

$$|R_m(\sigma) - R_{j^\star}(\sigma)| \leq \frac{L}{2}\sigma^2 \left(\|\varepsilon_m\|^2 + \|\varepsilon_{j^\star}\|^2\right),$$

there exists $\sigma_\varepsilon > 0$ such that, for all $\sigma \in (0, \sigma_\varepsilon)$,

$$|R_m(\sigma) - R_{j^\star}(\sigma)| \leq \frac{\sigma\Delta}{2} \qquad \text{for all } m \neq j^\star.$$

Thus,

$$c(\mu + \sigma\varepsilon_m) - c(\mu + \sigma\varepsilon_{j^\star}) \geq \frac{\sigma\Delta}{2} > 0$$

for all $m \neq j^\star$. Hence, $m^\star = j^\star$ for all sufficiently small $\sigma > 0$, almost surely.

By Lemma 2,

$$\partial_\sigma J^M(\mu, \sigma) = \mathbb{E}\left[-\nabla c(\mu + \sigma\varepsilon_{m^\star})^\top \varepsilon_{m^\star}\right].$$

Using the eventual equality $m^\star = j^\star$ and the continuity of $\nabla c$,

$$-\nabla c(\mu + \sigma\varepsilon_{m^\star})^\top \varepsilon_{m^\star} \to -g^\top \varepsilon_{j^\star} = -\min_{m=1,\ldots,M} g^\top \varepsilon_m$$

almost surely as $\sigma \downarrow 0$.

We now justify dominated convergence. For $0 < \sigma \leq 1$, Assumption 1 gives

$$\|\nabla c(\mu + \sigma\varepsilon_{m^\star})\| \leq \|g\| + L \max_m \|\varepsilon_m\|.$$

Therefore,

$$\left|\nabla c(\mu + \sigma\varepsilon_{m^\star})^\top \varepsilon_{m^\star}\right| \leq \left(\|g\| + L \max_m \|\varepsilon_m\|\right) \max_m \|\varepsilon_m\|.$$

The right-hand side is integrable. By dominated convergence,

$$\lim_{\sigma\downarrow 0} \partial_\sigma J^M(\mu, \sigma) = -\mathbb{E}\left[\min_{m=1,\ldots,M} g^\top \varepsilon_m\right].$$

Since $g^\top \varepsilon_m \overset{d}{=} \|g\| Z_m$,

$$-\mathbb{E}\left[\min_m g^\top \varepsilon_m\right] = \|g\| \mathbb{E}\left[\max_m Z_m\right] = \|\nabla c(\mu)\| \beta_M,$$

where $\beta_M = \mathbb{E}[\max_{m=1,\ldots,M} Z_m]$. For $M \geq 2$, $\beta_M > 0$ because

$$\beta_M \geq \mathbb{E}[\max\{Z_1, Z_2\}] = \frac{1}{2}\mathbb{E}|Z_1 - Z_2| > 0.$$

Therefore,

$$\lim_{\sigma\downarrow 0} \partial_\sigma J^M(\mu, \sigma) = \|\nabla c(\mu)\| \beta_M > 0.$$

By the definition of a one-sided limit, there exists $\sigma_0 > 0$ such that

$$\partial_\sigma J^M(\mu, \sigma) > 0 \qquad \text{for all } \sigma \in (0, \sigma_0).$$

□

### A.2 Proof of Proposition 2

***Proof.*** For $M = 1$,

$$J^1(\mu, \sigma) = -\mathbb{E}[c(\mu + \sigma\varepsilon)].$$

By Assumptions 1 and 5, differentiation under the expectation is justified by dominated convergence, and

$$\partial_\sigma J^1(\mu, \sigma) = -\mathbb{E}[\nabla c(\mu + \sigma\varepsilon)^\top \varepsilon].$$

Let $A = \mu + \sigma\varepsilon$. By Gaussian integration by parts, for each coordinate $i$,

$$\mathbb{E}[\partial_i c(A)\varepsilon_i] = \sigma \mathbb{E}[\partial^2_{ii} c(A)].$$

Summing over $i = 1, \ldots, d$ yields

$$\mathbb{E}[\nabla c(A)^\top \varepsilon] = \sigma \mathbb{E}[\mathrm{tr}(\nabla^2 c(A))].$$

By Assumptions 4 and 5,

$$\nabla^2 c(A) \succeq \lambda I_d.$$

Therefore,

$$\mathrm{tr}(\nabla^2 c(A)) \geq \lambda d.$$

Hence,

$$\partial_\sigma J^1(\mu, \sigma) = -\sigma \mathbb{E}[\mathrm{tr}(\nabla^2 c(A))] \leq -\lambda d \sigma < 0.$$

□

**Lemma 3** (No ties under strong convexity)**.** Assume Assumptions 1, 3, and 4. Let $X_1, \ldots, X_M$ be independent random variables on $\mathbb{R}^d$ with densities with respect to Lebesgue measure. Then

$$\mathbb{P}\left(\exists i \neq j \text{ such that } c(X_i) = c(X_j)\right) = 0.$$

***Proof.*** We first show that each level set of $c$ has Lebesgue measure zero. By Assumptions 3 and 4, 0 is the unique minimizer of $c$, and $c(0) = 0$.

Fix $t \in \mathbb{R}$. If $t < 0$, then $\{x : c(x) = t\}$ is empty. If $t = 0$, then $\{x : c(x) = t\} = \{0\}$, which has Lebesgue measure zero. Now suppose $t > 0$ and define

$$C_t = \{x \in \mathbb{R}^d : c(x) \leq t\}.$$

The set $C_t$ is convex. It also has nonempty interior because Assumptions 1 and 3 imply $c(x) \leq (L/2)\|x\|^2$, so a sufficiently small ball around 0 is contained in $C_t$. We claim that

$$\{x : c(x) = t\} \subseteq \partial C_t.$$

Suppose, for contradiction, that $x \in \{c = t\}$ belongs to the interior of $C_t$. Then there exists a nonzero vector $u$ such that $x + u \in C_t$ and $x - u \in C_t$. By strong convexity,

$$c(x) = c\left(\frac{x + u + x - u}{2}\right) \leq \frac{c(x + u) + c(x - u)}{2} - \frac{\lambda}{2}\|u\|^2 \leq t - \frac{\lambda}{2}\|u\|^2 < t,$$

contradicting $c(x) = t$. Thus, the level set is contained in the boundary of the convex set $C_t$. The boundary of a convex set in $\mathbb{R}^d$ has Lebesgue measure zero. Hence every level set of $c$ has Lebesgue measure zero.

Since each $X_i$ has a density,

$$\mathbb{P}(c(X_i) = t) = \mathbb{P}(X_i \in \{x : c(x) = t\}) = 0$$

for every $t \in \mathbb{R}$. Thus $c(X_i)$ has no atoms. The same argument as in Lemma 1 gives

$$\mathbb{P}(\exists i \neq j : c(X_i) = c(X_j)) = 0.$$

□

### A.3 Proof of Proposition 3

***Proof.*** Assumptions 3 and 4 imply

$$c(a) \geq \frac{\lambda}{2}\|a\|^2 \qquad \text{for all } a \in \mathbb{R}^d.$$

Assumptions 1 and 3 imply

$$c(a) \leq \frac{L}{2}\|a\|^2 \qquad \text{for all } a \in \mathbb{R}^d,$$

and

$$\|\nabla c(a)\| = \|\nabla c(a) - \nabla c(0)\| \leq L\|a\|.$$

By Lemma 3, the selected index

$$m^\star \in \arg\min_{m=1,\ldots,M} c(A_m)$$

is almost surely unique. For a fixed realization of $\varepsilon_{1:M}$, define

$$F(\mu, \sigma) = \max_{m=1,\ldots,M} -c(\mu + \sigma\varepsilon_m).$$

By Danskin's theorem,

$$\nabla_\mu F(\mu, \sigma) = -\nabla c(A_{m^\star}),$$

and

$$\partial_\sigma F(\mu, \sigma) = -\nabla c(A_{m^\star})^\top \varepsilon_{m^\star}$$

almost surely. Furthermore,

$$\|\nabla c(A_{m^\star})\| \leq L\|A_{m^\star}\| \leq L\left(\|\mu\| + \sigma \max_m \|\varepsilon_m\|\right),$$

and

$$|\nabla c(A_{m^\star})^\top \varepsilon_{m^\star}| \le L\|A_{m^\star}\|\|\varepsilon_{m^\star}\| \le L\left(\|\mu\| + \sigma \max_m \|\varepsilon_m\|\right) \max_m \|\varepsilon_m\|.$$

Both right-hand sides are integrable because $M < \infty$ and Gaussian moments are finite. Thus, differentiation can be interchanged with expectation, yielding

$$\nabla_\mu J^M(\mu,\sigma) = \mathbb{E}[-\nabla c(A_{m^\star})],$$

and

$$\partial_\sigma J^M(\mu,\sigma) = \mathbb{E}[-\nabla c(A_{m^\star})^\top \varepsilon_{m^\star}].$$

Using $\|\nabla c(a)\| \le L\|a\|$,

$$\|\nabla_\mu J^M(\mu,\sigma)\| \le \mathbb{E}[\|\nabla c(A_{m^\star})\|] \le L\mathbb{E}[\|A_{m^\star}\|].$$

Similarly,

$$|\partial_\sigma J^M(\mu,\sigma)| \le \mathbb{E}[\|\nabla c(A_{m^\star})\|\|\varepsilon_{m^\star}\|] \le L\mathbb{E}[\|A_{m^\star}\|\|\varepsilon_{m^\star}\|].$$

Since $m^\star$ minimizes $c(A_m)$,

$$c(A_{m^\star}) = \min_{m=1,\dots,M} c(A_m).$$

Using the quadratic bounds on $c$,

$$\frac{\lambda}{2}\|A_{m^\star}\|^2 \le c(A_{m^\star}) = \min_m c(A_m) \le \frac{L}{2}\min_m \|A_m\|^2.$$

Therefore,

$$\|A_{m^\star}\| \le \sqrt{\frac{L}{\lambda}} \min_{m=1,\dots,M} \|A_m\|.$$

Substituting this bound gives

$$\|\nabla_\mu J^M(\mu,\sigma)\| \le L\sqrt{\frac{L}{\lambda}}\,\mathbb{E}\left[\min_m \|A_m\|\right],$$

and

$$|\partial_\sigma J^M(\mu,\sigma)| \le L\sqrt{\frac{L}{\lambda}}\,\mathbb{E}\left[\min_m \|A_m\|\|\varepsilon_{m^\star}\|\right].$$

□

# B Details of the gradient analysis

We provide additional details for the vector field analysis in Sec. 5.1 and the effect of Adam's normalization in Sec. 5.3.

## B.1 Details for the vector field analysis

In Sec. 5.1, we visualized the gradient landscape of the ReMax objective to build intuition for its exploration-encouraging properties. Here we describe the setup and methodology used to generate these vector fields.

We consider a stateless continuous-control problem in which the policy is a one-dimensional Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$. To enable differentiation through sampling, actions are parameterized using the reparameterization trick, $a_i = \mu + \sigma\varepsilon_i$, where $\varepsilon_i \sim \mathcal{N}(0,1)$ are i.i.d. standard normal noise variables. We use the quadratic deterministic reward $r(a) = -a^2$, making $a = 0$ the unique optimal action. For entropy regularization, we set $\alpha = 0.5$ and use $r(a) = -a^2 + \alpha\frac{1}{2}\log(2\pi\sigma^2)$.

To construct the vector field, we evaluate gradients of the ReMax objective with respect to $\mu$ and $\sigma$ on a grid defined by $\mu \in [-2, 2]$ and $\sigma \in [0.1, 2]$. For each coordinate $(\mu, \sigma)$, we sample a batch of $B = 16$ actions and compute gradients using the reparameterization (RP) estimator with automatic differentiation. A single stochastic gradient evaluation would produce a noisy vector field. To approximate the expected gradient landscape $\mathbb{E}[\nabla_{\mu,\sigma}\mathcal{J}^M]$, we therefore average RP gradients over 100 independent Monte Carlo trials for each grid coordinate.

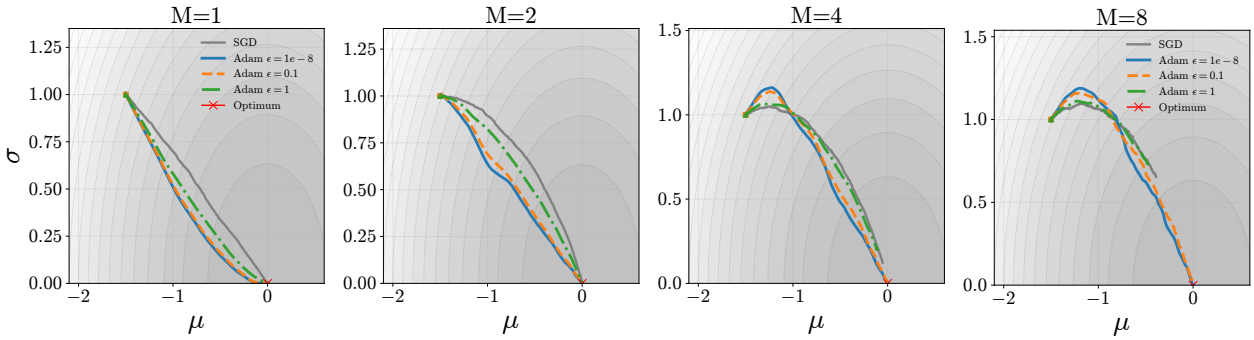



Figure 5: The two-dimensional trajectories of the parameter $(\mu, \sigma)$ optimized by Adam starting from $(\mu, \sigma) = (-1.5, 1.0)$ with different $\varepsilon$ values ($\varepsilon \in \{10^{-8}, 10^{-1}, 1.0\}$) and by SGD. As the vector field indicates, increasing $M$ makes the path deviate more from the optimum and slows convergence. However, with smaller $\varepsilon$, convergence near the optimum remains fast, whereas larger $\varepsilon$ aligns the trajectory with the SGD path that follows the vector field.

### B.2 Details for Adam's normalization

In Sec. 5.3, we showed that Adam's normalization can mitigate the damping effect of ReMax. To illustrate this, we plot the convergence path of $(\mu, \sigma)$ in the two-dimensional parameter space. As $M$ increases, the trajectory becomes more gradual around the optimum, consistent with the vector-field analysis.

Furthermore, the consistency with SGD observed for larger $\varepsilon$ in Fig. 2 also appears in these trajectories, supporting the argument that increasing $\varepsilon$ makes the optimization path follow the vector field more closely.

## C Details of the experiments

Here we provide additional details for the experiments in Sec. 7.

### C.1 Setup

**Implementation.** The implementation of the ReMAC algorithm is based on the `rejax`[1] implementation of SAC. We include the code in the supplementary material.

**Hyperparameters.** We follow the SAC and PPO implementations and hyperparameters provided in the `rejax` library (Liesen et al., 2024), which contains environment-specific tuned configurations for Brax tasks. Listing all hyperparameters would largely duplicate that repository, so we refer readers to the environment-specific configuration files in the supplementary material at `brax/configs/` for the exact configurations used in our experiments. The hyperparameters (learning rate, batch size, discount factor, Polyak factor, etc.) were tuned for each environment by the authors of `rejax`. We tuned the learning rate for ReMAC by sweeping $\{10^{-4}, 2\times10^{-4}, 3\times10^{-4}, 5\times10^{-4}, 10^{-3}\}$ with 3 random seeds and selecting a value that performed reasonably well across environments and $M$ for fixed $\varepsilon = 10^{-8}$. The selected learning rates are provided in Tab. 1. The other hyperparameters are the same as those used for SAC and PPO in `rejax`.

### C.2 Additional results

Here we provide additional results for the experiments in Sec. 7.

**Entropy and return with different $\varepsilon$** We plot the average policy entropy and return with different $\varepsilon \in \{10^{-8}, 10^{-2}, 10^{-1}, 1.0\}$ for $M = 1, 2, 4, 8$ in Figs. 6–13. For all $M$, entropy increases as $\varepsilon$ increases,

[1] https://github.com/keraJLi/rejax

Table 1: Selected hyperparameters for ReMAC.

| **Environment** | **Learning Rate** |
|---|---|
| HalfCheetah | $10^{-4}$ |
| Ant | $2 \times 10^{-4}$ |
| Hopper | $3 \times 10^{-4}$ |
| Reacher | $5 \times 10^{-4}$ |
| Swimmer | $10^{-4}$ |
| Walker2d | $10^{-4}$ |

Table 2: Wall-clock training time of SAC and ReMAC with $B = 8$ and $B = 16$ on HalfCheetah for 3M environment steps, vectorized over 10 random seeds.

| **SAC** | **ReMAC** ($B = 8$) | **ReMAC** ($B = 16$) |
|---|---|---|
| 1 h 03 min | 1 h 33 min | 2 h 06 min |

consistent with Sec. 5.3. The return tends to decrease for larger $\varepsilon$, as expected, because a larger $\varepsilon$ reduces the effective step size and can slow learning.

**Ablation by the Learning Rate.** Since Adam's $\varepsilon$ rescales the per-parameter step size, a natural question is whether its effect can be reproduced simply by changing the global learning rate $\alpha$. To test this, we fix $M = 4$, sweep $\alpha$ for both the default $\varepsilon = 10^{-8}$ and the large value $\varepsilon = 1$, and plot the resulting return and entropy in Figs. 14 and 15. We selected three environments (HalfCheetah, Swimmer, and Walker2d) because their default learning rate is $10^{-4}$, which makes it easy to compare the learning-rate effect. At $\varepsilon = 1$, where the large denominator strongly damps the effective step size, increasing $\alpha$ from $10^{-4}$ to $10^{-3}$ restores the effective step size and recovers most of the return lost relative to $\varepsilon = 10^{-8}$, while the entropy decreases as $\alpha$ grows yet stays above the $\varepsilon = 10^{-8}$ level, especially in HalfCheetah and Swimmer. At $\varepsilon = 10^{-8}$, the learning rate tuned for SAC ($\alpha = 10^{-4}$) already attains the highest return, and decreasing $\alpha$ from $10^{-4}$ to $10^{-5}$ only slows learning while the entropy still collapses to the low values characteristic of small $\varepsilon$. Hence $\alpha$ and $\varepsilon$ are not interchangeable: $\alpha$ scales the whole update uniformly, whereas $\varepsilon$ reshapes the per-coordinate normalization and thereby controls the entropy, consistent with the analysis in Sec. 5.3. Furthermore, we found that the best average return is achieved either with $\varepsilon = 1$ (HalfCheetah) or with $\varepsilon = 10^{-8}$ (Walker2d and Swimmer), depending on the environment, further suggesting the importance of tuning $\varepsilon$ and $\alpha$ jointly to achieve the best performance with ReMAC.

**Ablation by the Batch Size.** The unbiased ReMax estimator draws $B$ action samples per state to estimate the order-$M$ objective and is only defined for $B \geq M$, with larger $B$ reducing the variance of the estimate. We ablate $B \in \{8, 16\}$ at $\varepsilon = 10^{-8}$ with the tuned learning rate for $M \in \{1, 2, 4, 8\}$ in Fig. 16. Across all tasks and all $M$, the two settings lie within each other's confidence intervals: $B = 16$ is comparable to, and occasionally marginally better than, $B = 8$, and the relative ordering across $M$ is unchanged. Because the estimator becomes noisier as $M$ approaches $B$, we use $B = 16$ for the largest setting $M = 8$ and $B = 8$ otherwise; this ablation confirms that the choice of $B$ does not materially affect the reported results.

**Computational cost.** We report the wall-clock training time of SAC and ReMAC with $B = 8$ and $B = 16$ on HalfCheetah for 3M environment steps, vectorized over 10 random seeds. This setting provides a controlled comparison because SAC and ReMAC share the same training pipeline and network architectures and differ mainly in the actor loss, while ReMAC also omits the temperature update used for entropy regularization. The evaluation frequency is once every 30,000 environment steps, and all experiments are run on a GeForce RTX 2080 Ti. The results are reported in Tab. 2. ReMAC is slower than SAC because its actor update requires additional Q-network evaluations for $B$ action samples per state. As expected, the wall-clock time increases with $B$. This additional computational cost is a limitation of the current implementation.

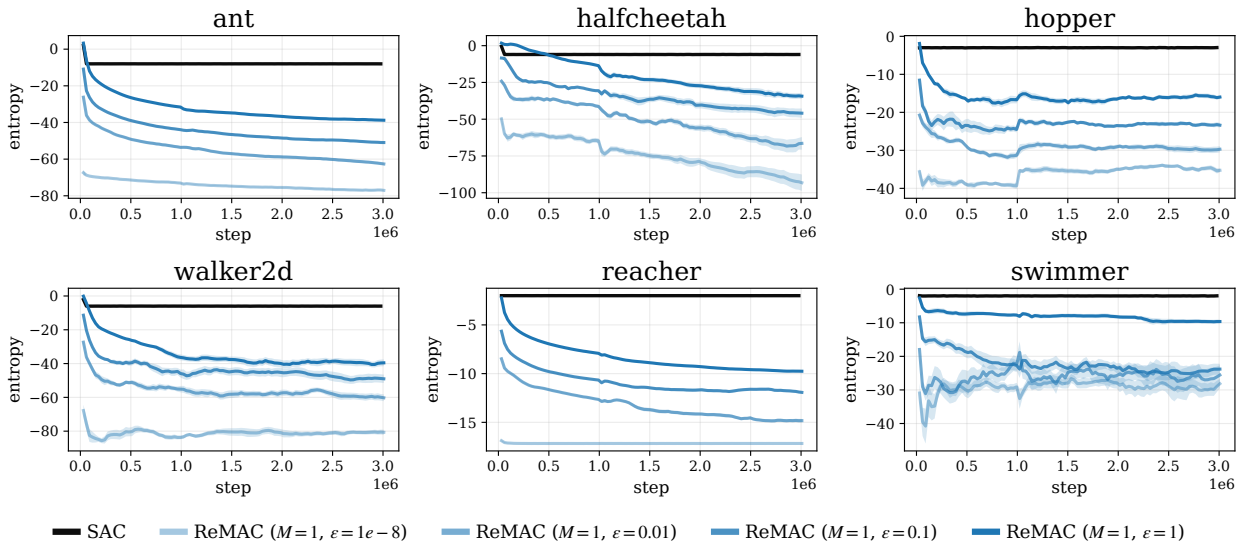



Figure 6: The average policy entropy of ReMAC with $M = 1$ and different $\varepsilon \in \{10^{-8}, 10^{-2}, 10^{-1}, 1.0\}$ using Adam for all tasks.

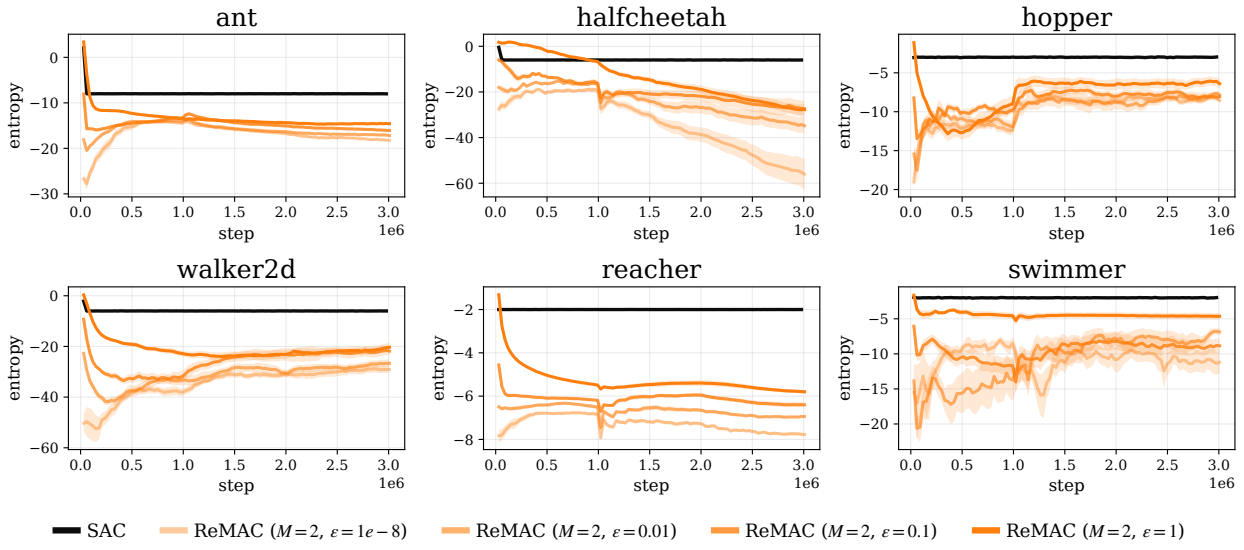



Figure 7: The average policy entropy of ReMAC with $M = 2$ and different $\varepsilon \in \{10^{-8}, 10^{-2}, 10^{-1}, 1.0\}$ using Adam for all tasks.

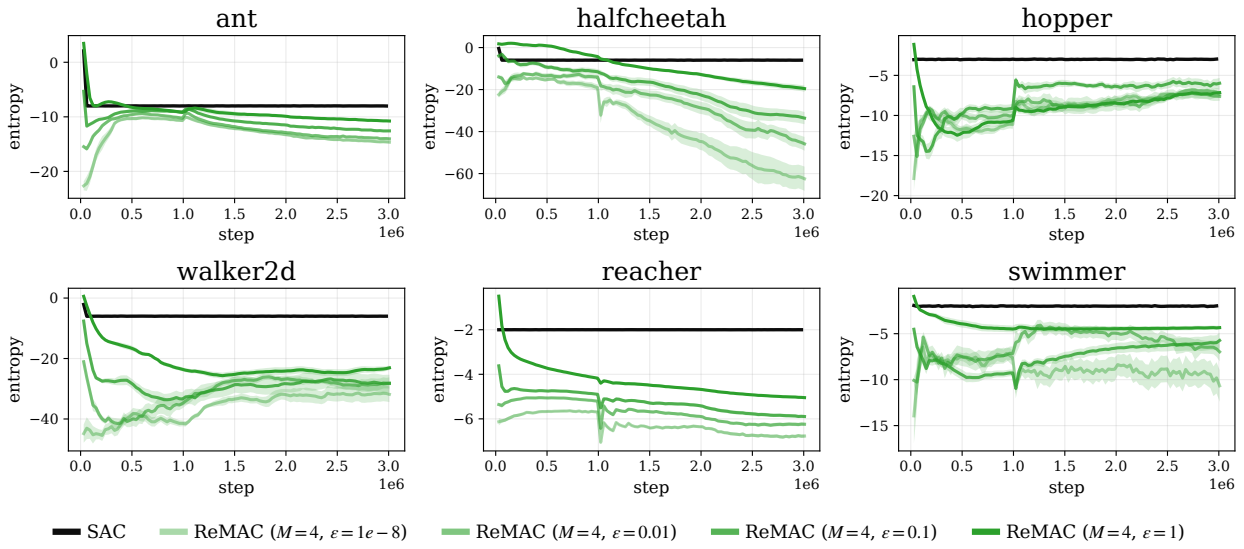



Figure 8: The average policy entropy of ReMAC with $M = 4$ and different $\varepsilon \in \{10^{-8}, 10^{-2}, 10^{-1}, 1.0\}$ using Adam for all tasks.

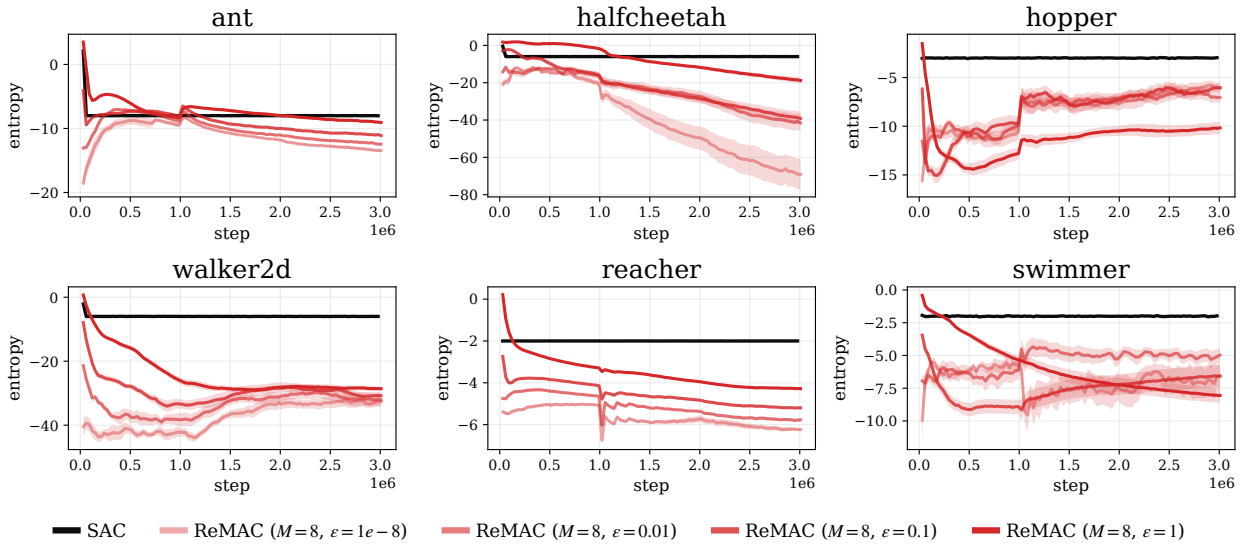



Figure 9: The average policy entropy of ReMAC with $M = 8$ and different $\varepsilon \in \{10^{-8}, 10^{-2}, 10^{-1}, 1.0\}$ using Adam for all tasks.

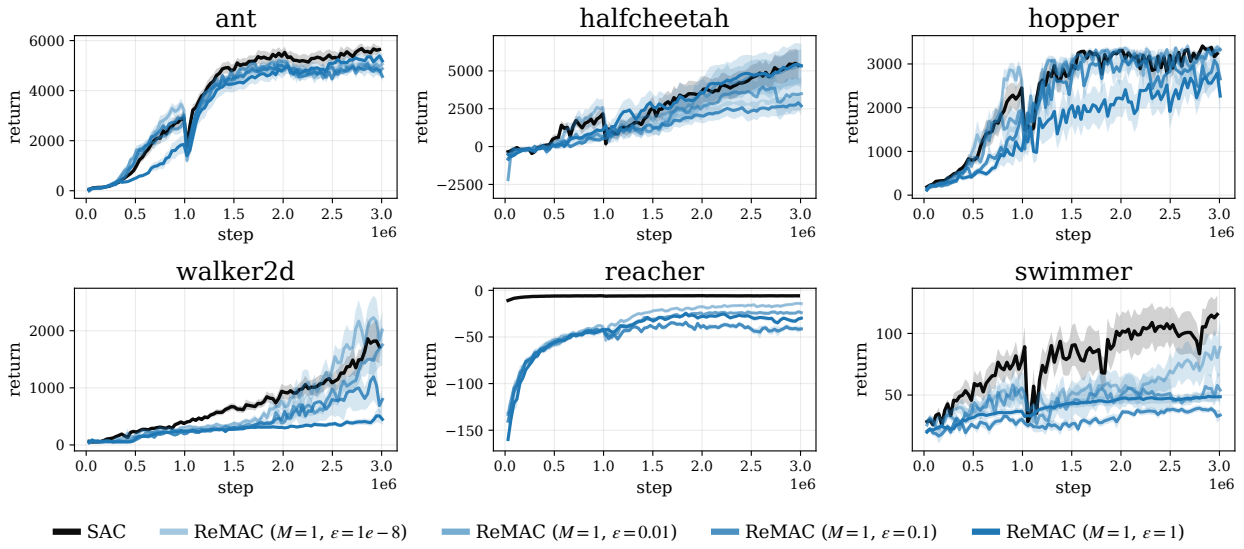



Figure 10: The average return of ReMAC with $M = 1$ and different $\varepsilon \in \{10^{-8}, 10^{-2}, 10^{-1}, 1.0\}$ using Adam for all tasks.

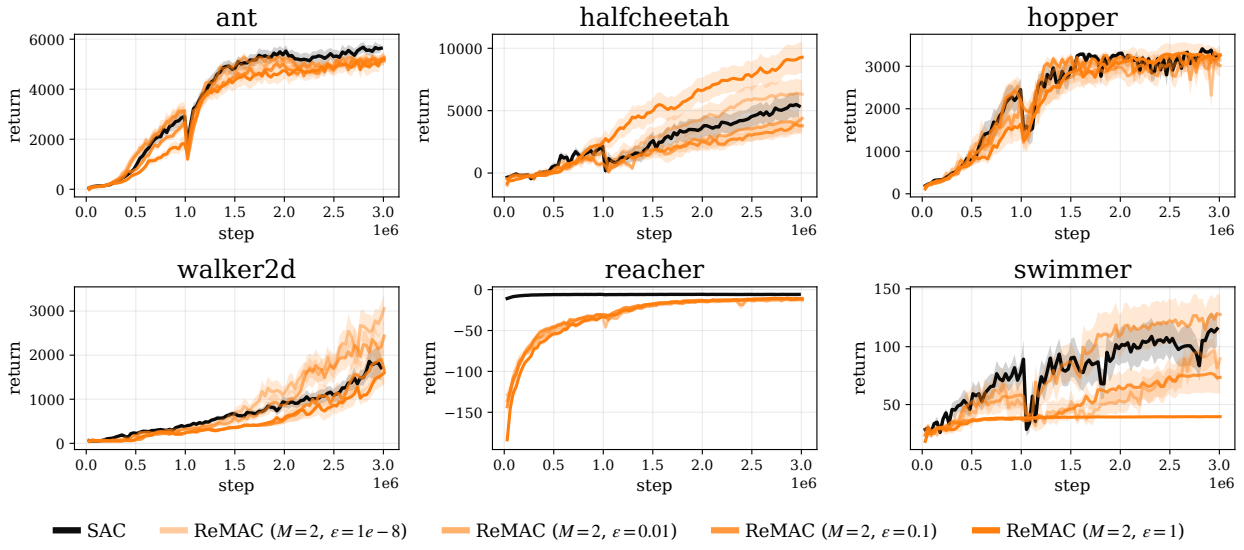



Figure 11: The average return of ReMAC with $M = 2$ and different $\varepsilon \in \{10^{-8}, 10^{-2}, 10^{-1}, 1.0\}$ using Adam for all tasks.

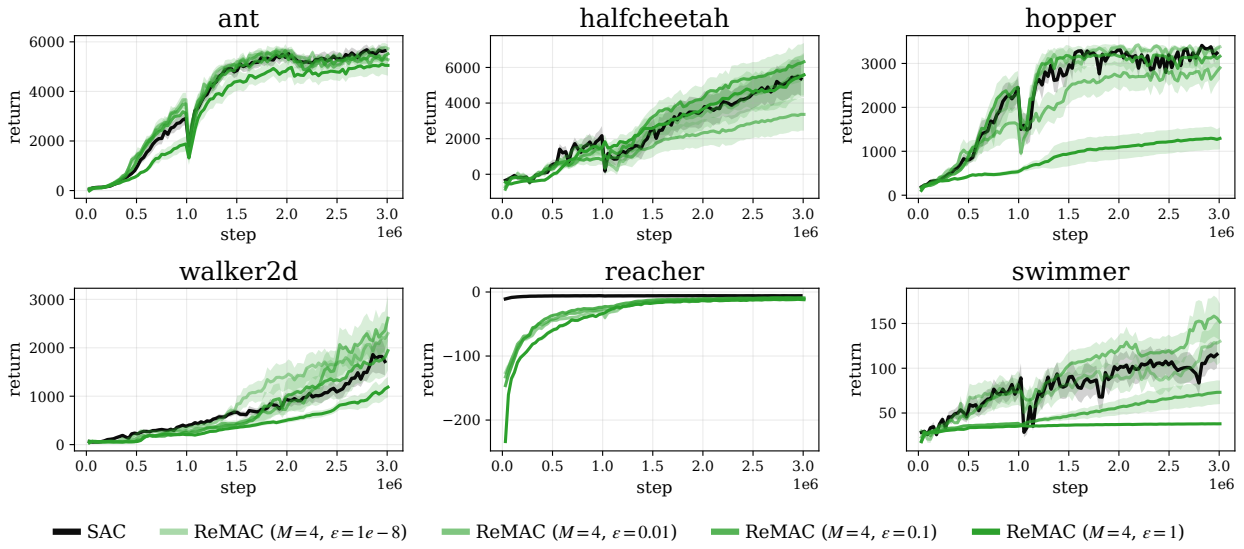



Figure 12: The average return of ReMAC with $M = 4$ and different $\varepsilon \in \{10^{-8}, 10^{-2}, 10^{-1}, 1.0\}$ using Adam for all tasks.

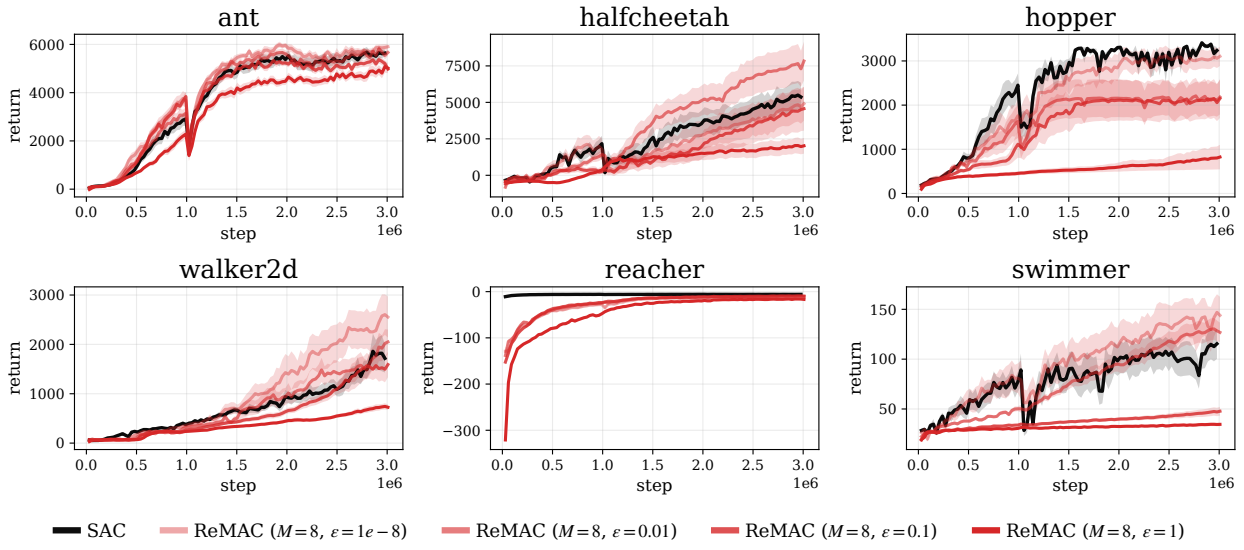



Figure 13: The average return of ReMAC with $M = 8$ and different $\varepsilon \in \{10^{-8}, 10^{-2}, 10^{-1}, 1.0\}$ using Adam for all tasks.

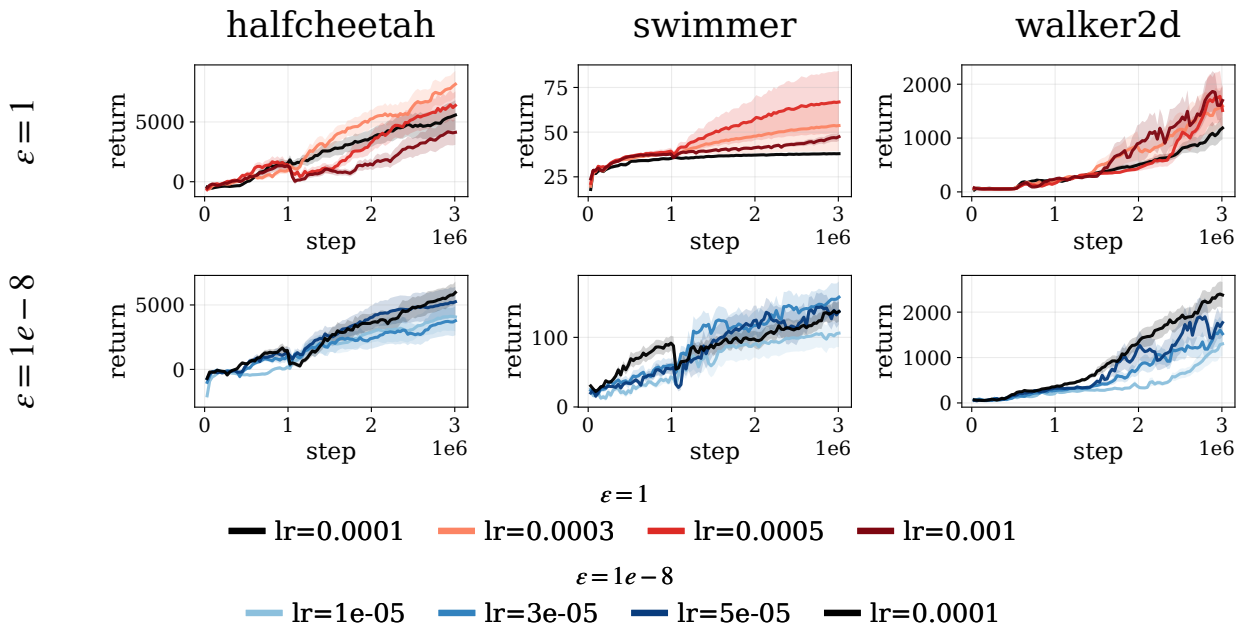



Figure 14: The average return of ReMAC with $M = 4$ under different learning rates: $\varepsilon = 1$ (top row, $\alpha \in \{10^{-4}, 3 \times 10^{-4}, 5 \times 10^{-4}, 10^{-3}\}$) and $\varepsilon = 10^{-8}$ (bottom row, $\alpha \in \{10^{-5}, 3 \times 10^{-5}, 5 \times 10^{-5}, 10^{-4}\}$), for HalfCheetah, Swimmer, and Walker2d. The shared default learning rate $\alpha = 10^{-4}$ is drawn in black in both rows.

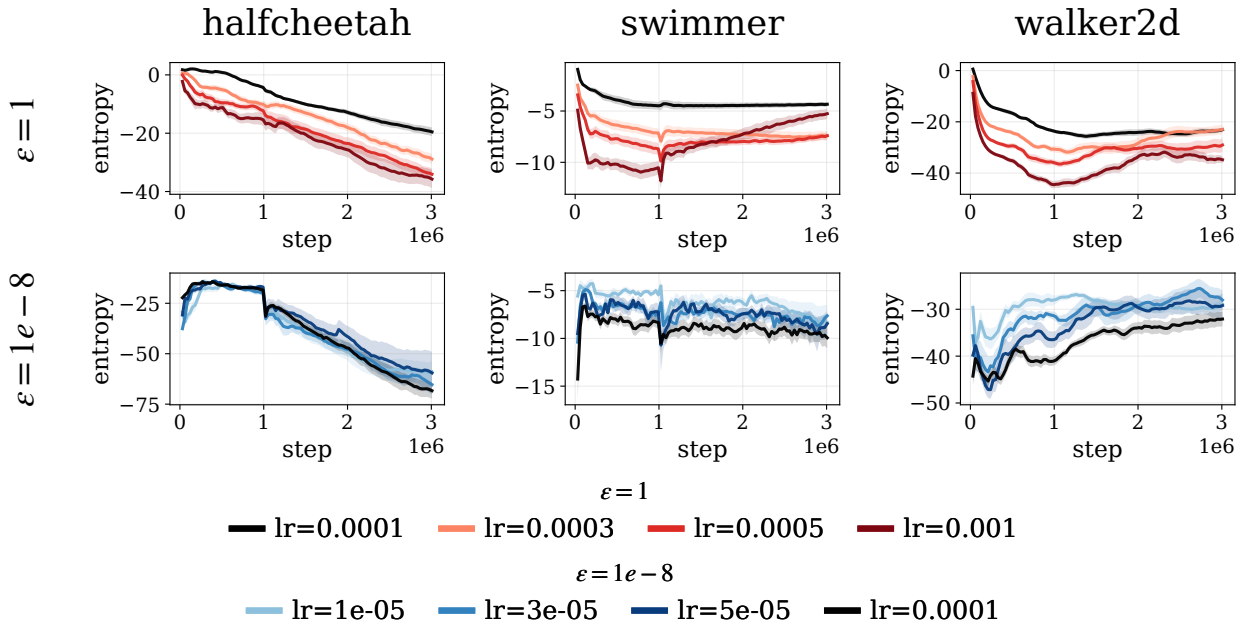
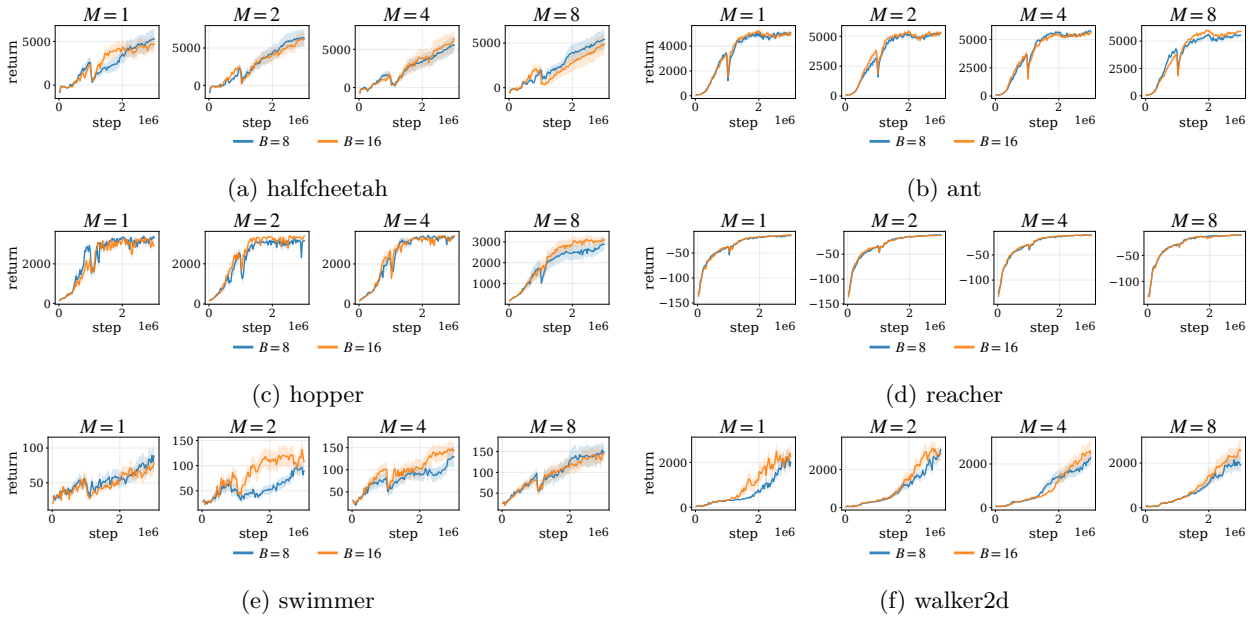



Figure 15: The average policy entropy of ReMAC with $M = 4$ under different learning rates: $\varepsilon = 1$ (top row, $\alpha \in \{10^{-4}, 3 \times 10^{-4}, 5 \times 10^{-4}, 10^{-3}\}$) and $\varepsilon = 10^{-8}$ (bottom row, $\alpha \in \{10^{-5}, 3 \times 10^{-5}, 5 \times 10^{-5}, 10^{-4}\}$), for HalfCheetah, Swimmer, and Walker2d. The shared default learning rate $\alpha = 10^{-4}$ is drawn in black in both rows.




(a) halfcheetah (b) ant

(c) hopper (d) reacher

(e) swimmer (f) walker2d

Figure 16: The average return of ReMAC with $M = 1, 2, 4, 8$ and $\varepsilon = 10^{-8}$ under different batch sizes $B \in \{8, 16\}$ for all tasks.