Seizing Serendipity: Exploiting the Value of Past Success in Off-Policy Actor-Critic

Tianying Ji [1] **Yu Luo** [1] **Fuchun Sun** [1] **Xianyuan Zhan** [2 3] **Jianwei Zhang** [4] **Huazhe Xu** [3 5 6]

Abstract

Learning high-quality Q-value functions plays a key role in the success of many modern off-policy deep reinforcement learning (RL) algorithms. Previous works primarily focus on addressing the value overestimation issue, an outcome of adopting function approximators and off-policy learning. Deviating from the common viewpoint, we observe that Q-values are often underestimated in the latter stage of the RL training process, potentially hindering policy learning and reducing sample efficiency. We find that such a long-neglected phenomenon is often related to the use of inferior actions from the current policy in Bellman updates as compared to the more optimal action samples in the replay buffer. To address this issue, our insight is to incorporate sufficient exploitation of past successes while maintaining exploration optimism. We propose the Blended Exploitation and Exploration (BEE) operator, a simple yet effective approach that updates Q-value using both historical best-performing actions and the current policy. Based on BEE, the resulting practical algorithm BAC outperforms state-of-the-art methods in **over 50** continuous control tasks and achieves strong performance in failure-prone scenarios and **real-world robot** tasks. Benchmark results and videos are available at https://jity16.github.io/BEE/.

[1]Department of Computer Science and Technology, Tsinghua University [2]Institute for AI Industry Research, Tsinghua University [3]Shanghai Artificial Intelligence Laboratory [4]Department of Informatics, University of Hamburg [5]Institute for Interdisciplinary Information Sciences, Tsinghua University [6]Shanghai Qi Zhi Institute. Correspondence to: Fuchun Sun <fcsun@tsinghua.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

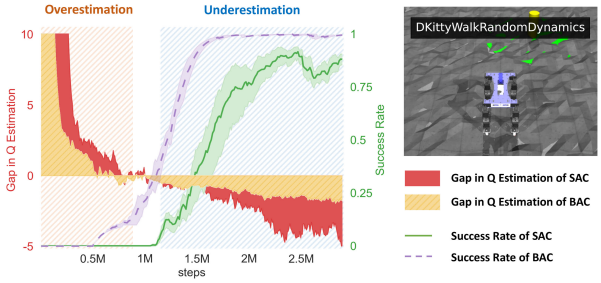


Figure 1. **Example of the underestimation issue.** In the DKittyWalkRandomDynamics task, when the current policy-generated action is inferior to the best action in the replay buffer, which usually occurs in the later stage of training, SAC is more prone to underestimation pitfalls than BAC. The gap in Q estimation is evaluated by comparing the SAC Q-values and the Monte-Carlo Q estimates.

1. Introduction

Reinforcement learning (RL) has achieved impressive progress in solving many complex decision-making problems in recent years (Mnih et al., 2015; Silver et al., 2016; Hutter et al., 2016; Ouyang et al., 2022). Many of these advances are obtained by off-policy deep RL algorithms, where the ability to leverage off-policy samples to learn high-quality value functions underpins their effectiveness. Value overestimation (Fujimoto et al., 2018; Moskovitz et al., 2021) has long been recognized as an important issue in off-policy RL algorithms, which is primarily associated with the function approximation errors (Fujimoto et al., 2018) and the side-effect of off-policy learning (Auer et al., 2008; Jin et al., 2018; Azar et al., 2017), and can potentially lead to suboptimal policies. To tackle this issue, techniques for alleviating value overestimations, such as double-Q technique, have been ubiquitously adopted in modern off-policy RL algorithms (Haarnoja et al., 2018a; Laskin et al., 2020; Han & Sung, 2021; Moskovitz et al., 2021).

Intriguingly, we find that in common online off-policy actor-critic (AC) methods, the well-known value overestimation issue could disappear and be replaced by value underestimation in the later training stage. Figure 1 shows an illustrative example of such a phenomenon in a robotic task, and we also observe similar patterns over a diverse range of RL tasks and different off-policy AC algorithms, as will be

demonstrated in the following content of our paper. Such a phenomenon does not simply result from the techniques used for alleviating value overestimation (e.g., double-Q trick), but more fundamentally, because the Q-value update could be negatively impacted by the actions a' sampled from the current suboptimal policy during the Bellman Backup process. Such suboptimality is inevitable since the policy optimization updates over the current Q-value function (i.e., $\pi \leftarrow \arg\max_{a\in\pi} Q(s,a)$) is often impossible to recover the ideal policy within only a few policy gradient updates, especially with an evolving Q network. This can potentially lead to underestimated Q target update values, leading to inferior learning performance and sample efficiency.

To address this long-neglected phenomenon, we find that allowing the agent to fully exploit the best-performing actions stored in the replay buffer can be a natural cure. More concretely, if there exist more optimal actions in the replay buffer as compared to the ones generated by the current policy, then we can leverage them to perform more optimistic Bellman updates to resolve the underestimation issue. Such more optimal actions can be abundant especially in the later off-policy RL training stage, since the replay buffer is already filled by relatively diverse state-action pairs, which may be caused by exploration behaviors in online RL and the non-optimality optimization nature of the actor-critic framework.

In this paper, we connect this intuition with Bellman operators: the Bellman Exploitation operator enables effective exploitation of high-quality historical samples while the Bellman Exploration operator targets maintaining exploration optimism. This gives rise to a remarkably simple and effective mechanism, the Blended Exploration and Exploitation (BEE) operator, which combines the merits of both sides. BEE operator provides superior Q-value estimation, effectively avoiding the value underestimation issue. Moreover, it can be flexibly integrated into existing off-policy AC frameworks, leading to two practical algorithm instantiations: BAC (BEE Actor-Critic) for model-free settings and MB-BAC (Model-based BAC) for model-based settings.

BAC outperforms other popular online RL methods on various **MuJoCo**, **DMControl**, **Meta-World**, **ManiSkill2**, **Adroit**, **Shadow Dexterous Hand**, **MyoSuite**, **ROBEL** benchmark tasks by a large margin. On many failure-prone tasks, BAC achieves over 2x the evaluation scores of the strongest baseline. Moreover, we conduct **real-world** experiments on complex quadruped robot locomotion tasks, and BAC achieves strong performance. Furthermore, in our experiments, we observed unanimously improved performance when applying the BEE operator to different backbone algorithms, highlighting its flexibility and generic nature.

2. Related Works

Off-policy actor-critic methods, alternating between Q-value estimation and policy optimization w.r.t the Q-value, have been a cornerstone in reinforcement learning (RL) research (Casas, 2017; Mnih et al., 2016; Haarnoja et al., 2018a; Fujimoto et al., 2018; Lee et al., 2020; Zheng et al., 2022). A widely recognized challenge in these methods is the accuracy of value estimation (Kimura et al., 1998; Grondman et al., 2012), which is crucial for effective policy extraction. Inaccurate value estimations can significantly hinder policy updates and misguide exploration efforts.

Overestimation could erroneously attribute high values to suboptimal states and actions (Hasselt, 2010; Fujimoto et al., 2018; Cetin & Celiktutan, 2023). It has been a long-discussed problem. Previous works suggest overestimation is an outcome of the combination of off-policy learning and high-dimensional, nonlinear function approximation (Hasselt, 2010; Fujimoto et al., 2018; Kuznetsov et al., 2020), also associated with optimistic exploration (Jin et al., 2018; Laskin et al., 2020; Moskovitz et al., 2021). Hence, attempts to correct for overestimation, *e.g.*, taking the minimum of two separate critics, have been widely adopted in AC methods (Fujimoto et al., 2018; Haarnoja et al., 2018a; Han & Sung, 2021; Sun et al., 2022).

While underestimation could hamper the reselection of potentially high-value state-action pairs and thus negatively impact policy optimization, it has received much less attention compared to overestimation. Some existing efforts may blame the underestimation issues for directly taking the minimum value from an ensemble of Q-values (a technique originally intended to combat overestimation), thus devising a less extreme form of the minimum value to avoid overly conservative estimates (Ciosek et al., 2019; Moskovitz et al., 2021; Peer et al., 2021).

However, attributing underestimation solely to the clipped double-Q technique is an oversimplification. Our findings suggest that the inherent nature of actor-critic optimization in RL also contributes to underestimation. This indicates that previous works that focused solely on adjusting the double-Q technique may not fully address the issue. A more comprehensive approach that considers the underlying interplay of actor-critic methods is necessary to effectively address underestimation in reinforcement learning.

For extensive related works, please refer to Appendix A.

3. Preliminaries

Markov decision process. We denote a discounted Markov decision process (MDP) as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ the action space, $r : \mathcal{S} \times \mathcal{A} \in [-R_{max}, R_{max}]$ the reward function, and

$\gamma \in (0,1)$ the discount factor, and $P(\cdot \mid s,a)$ stands for transition dynamics.

Off-policy actor-critic RL. In Q-learning, the policy is derived from the state-action value function Q by selecting the maximizing action. The learning of the Q-value involves repeated application of the Bellman optimality operator, *i.e.*, $\mathcal{T}^*Q(s,a) \triangleq r(s,a)+\gamma\cdot\max_{a'\in\mathcal{A}}\mathbb{E}_{s'\sim P(s'|s,a)}[Q(s',a')]$. However, it entails traversing all possible actions, being intractable in continuous action spaces (Kumar et al., 2019; Garg et al., 2023). Off-policy actor-critic methods tackle this issue by alternating between training a policy to maximize the Q value, denoted as $\pi = \arg\max_{a\sim\pi} Q(s,a)$, and repeatedly applying a Bellman evaluation operator $\mathcal{T}$ (Sutton, 1988; Watkins, 1989), defined as:

$$\mathcal{T}Q(s,a) \triangleq r(s,a)+\gamma\cdot\mathbb{E}_{s'\sim P(\cdot|s,a)}\left[\mathbb{E}_{a'\sim\pi(\cdot|s')}[Q(s',a')]\right].$$

In-sample learning. In-sample learning paradigms have been widely explored in offline RL. In-sample learning methods (Kostrikov et al., 2021; Xu et al., 2023; Garg et al., 2023) learn $V(s)$ and $Q(s,a)$ completely using dataset samples, following general learning objectives for $V(s)$ and $Q(s,a)$, depending on different choices of the f function. $\min_V \mathbb{E}_{(s,a)\sim\mathcal{D}}\mathcal{L}_V^f(Q(s,a)-V(s))$ for updating V-function and $\max_Q \mathbb{E}_{(s,a,s')\sim\mathcal{D}}[r(s,a)+\gamma V(s')-Q(s,a)]^2$ for updating Q-function. The well-known offline RL algorithm IQL also belongs to this family with $\mathcal{L}_V^f(y) = |\tau - \mathbb{1}(y<0)|y^2$, where $\tau \in (0,1)$ is the expectile hyperparameter. IQL can recover the optimal value function under the dataset support constraints. In particular, if $f=\log(x)$, it corresponds to EQL (Xu et al., 2023) and XQL (Garg et al., 2023) with $\mathcal{L}_V^f(y) = \exp(y/\alpha) - y/\alpha$. If $f = x-1$, it corresponds to SQL (Xu et al., 2023) with $\mathcal{L}_V^f(y) = \mathbb{1}(1+y/2\alpha>0)(1+y/2\alpha)^2 - y\,/alpha$. These Insights from the good practices of in-sample learning methods in offline RL could be leveraged to enhance online RL Q-value estimation.

4. Exploiting past success for off-policy optimization

In this section, we delve into the long-neglected underestimation bias in the latter stage of the RL training process. We identify that this bias can be partially attributed to the inherent non-optimality of the current policy in the Actor-Critic framework[1]. These discoveries encourage us to exploit the more optimal actions in the replay buffer to shorten the gap to the optima, hence mitigating underestimation. Finally, we arrive at a simple, versatile, yet effective solution, the BEE operator, and demonstrate its effectiveness.

[1]For comprehensive investigations of the underestimation issue, please see Appendix F.

4.1. Underestimation issue in Actor-Critic

The underestimation bias has long been attributed to the double-Q-technique, as highlighted in previous studies (Fujimoto et al., 2018; Moskovitz et al., 2021). Yet, one must consider whether this technique is the sole cause of the bias.

In Figure 2, we plot the Q-value estimation for SAC and TD3, alongside their variants without the double-Q technique. Our observations indicate that both SAC and TD3 encounter an underestimation issue across various robotic tasks. Intriguingly, even when we eliminate the double-Q-technique from well-known algorithms SAC (Haarnoja et al., 2018a) and TD3 (Fujimoto et al., 2018), the problem of underestimation still persists. This suggests the existence of other, less explored factors contributing to the underestimation issue. We identify the underlying optimization procedure in the Actor-Critic framework itself may also contribute to underestimation.

The optimization procedure of the AC framework contributes to underestimation. Ideally, the Bellman update needs to solve $Q(s,a) \leftarrow r(s,a) + \gamma\mathbb{E}_{s'}[\max_{a'} Q(s',a')]$. However, as $\max_{a'} Q(s',a')$ operations are often impractical to calculate, so in the AC framework, we typically iteratively evaluate target Q-value as $\mathbb{E}_\pi[Q(s',a')]$, while implicitly conducting the $\max$-Q operation in a separate policy improvement step to learn policy π. Note that the ideal $\pi = \arg\max_{a'\sim\pi} Q(s',a')$ is not possible to achieve practically within only a few policy gradient updates (Fujimoto et al., 2019; Chan et al., 2022). Hence, the actual target value used in AC Bellman update $\mathbb{E}_{s',a'\sim\pi}Q(s',a')$ can have a high chance to be smaller than $\mathbb{E}_{s'}[\max_{a'} Q(s',a')]$, causing underestimation.

In a nutshell, the inherent non-optimality of the current policy in the AC framework contributes to underestimation. Specifically, the discrepancy between the theoretically optimal policy and the practically achievable policy, especially in the latter stages of learning, would negatively affect the target update value. This is because the target-update actions a' sampled from the current policy may be inferior compared to those generated by an ideally optimal policy.

Hence, reducing the gap between target-update actions and those of an ideal optimal policy could lessen the underestimation issue. Now, the critical challenge lies in identifying a more optimal source for these target-update actions.

4.2. More optimal source for target-update actions

Actually, actions sampled from the non-optimality of the current policy may fall short of the optimal ones stored in the replay buffer. Naturally, exploiting the optimal actions from the replay buffer to bootstrap Q would shorten the gap to the optima, hence mitigating underestimation.

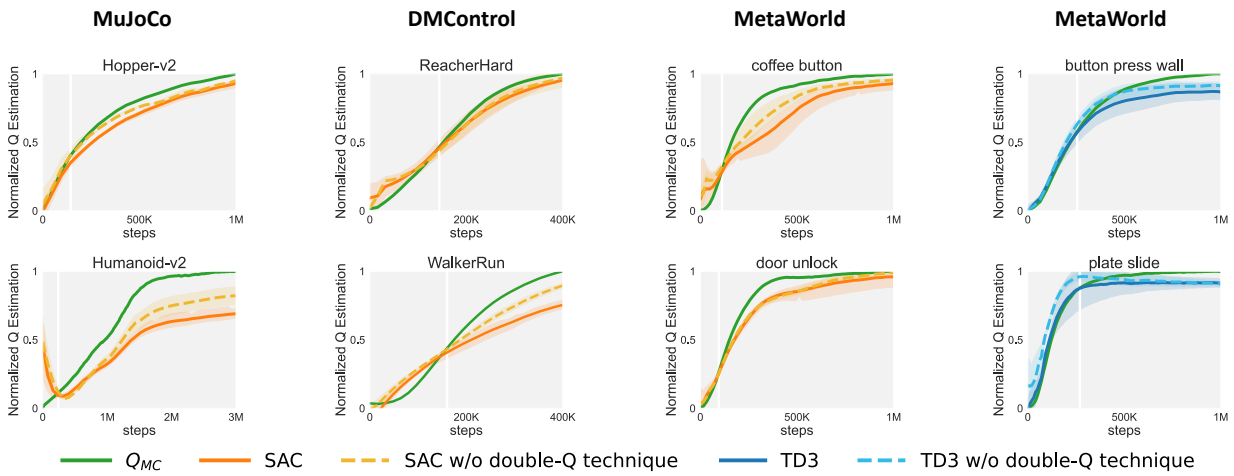


Figure 2. **Normalized Q estimation value comparison.** We plot the normalized Q-value estimates of two widely-used off-policy actor-critic algorithms, namely SAC and TD3, across various benchmark suites, spanning locomotion and manipulation tasks. The approximated true value Q_{MC} is obtained by Monte-Carlo Q estimates are derived from trajectories sampled using the current policy. The white vertical line marks the separation of the overestimation and underestimation stages during training. Moreover, we disabled the double-Q technique of SAC and TD3 and observed the underestimation issue still occurs.

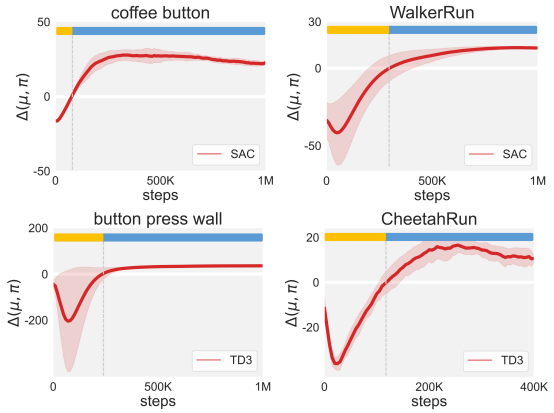


Figure 3. $\Delta(\mu, \pi)$ **with an SAC or TD3 agent.** Blue bars correspond to positive $\Delta(\mu, \pi)$, indicating an optimal policy derived from the replay buffer would outperform the current policy.

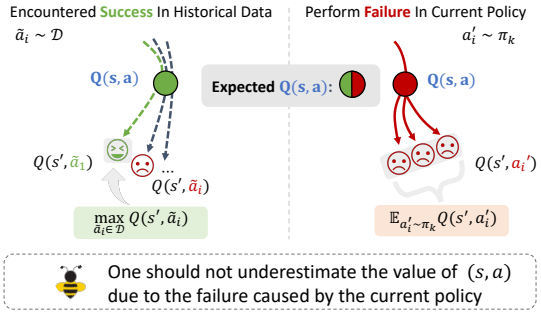


Figure 4. **Illustrative figure on target-update actions.** Leveraging the high-quality samples from the replay buffer would produce a more optimistic Q-value, thus mitigating value underestimation. Practically, the Bellman evaluation operator, whose target-update actions a' are only sampled from the current policy, tends to underestimate it.

To discern the optimal actions from the replay buffer, it's essential to characterize the implicit policy that can be inferred from it. During policy learning, the replay buffer accumulates samples from a series of historical policies, denoted as $\Pi^k = \{\pi_0, \pi_1, \ldots, \pi_k\}$, each associated with a mixture distribution weight α_k. Hence, the state-action visitation density in replay buffer is defined as $d^{\mu_k}(s, a) = \sum_{i=1}^{k} \alpha_i^k d^{\pi_i}(s, a)$, where μ_k is a mixed policy derived from the replay buffer (Zhang et al., 2021b; Wang et al., 2022b).

To quantify the discrepancy between the value of the optimal actions derived from the replay buffer and those from the current policy, we calculate the expected difference $\Delta(\mu_k, \pi_k)$ between the maximum Q-value over the policy mixture μ_k and the expected Q-value under the existing policy π_k at policy iteration k, stated as,

$$\Delta(\mu_k, \pi_k) = \mathbb{E}_s \left[\max_{a \sim \mu_k} Q^{\mu_k}(s, a) - \mathbb{E}_{a \sim \pi_k} \left[Q^{\pi_k}(s, a) \right] \right].$$

A positive $\Delta(\mu_k, \pi_k)$ indicates that the value of optimal target-update actions in the replay buffer exceeds that of the actions generated by the current policy. This suggests that an optimal policy derived from the replay buffer would outperform the current policy, implying a potential under-exploitation of valuable historical data.

Figure 3 empirically shows that during the course of training, the replay buffer could contain more optimal actions as compared to the ones generated by the current policy. This becomes prevalent, especially in the latter training stage,

(a) Q from pure $\mathcal{T}_{exploit}$ (b) Q from pure $\mathcal{T}_{explore}$ (c) Q from our $\mathcal{B}$ (d) Optimal Q and actions

Figure 5. **Comparison of three operators on a toy grid world.** The agent's goal is to navigate from the bottom of the maze to the top left. The color of each square shows the learned value, red arrows reveal incorrect actions, and question marks indicate unencountered states. (a) employing a pure exploitation operator may lead to insufficient exploration, causing the agent to miss out on blocks. Conversely, (b) a pure exploration operator $\mathcal{T}^{\pi}_{explore}$ tends to overestimate the value of less frequently visited areas while underestimating those on the optimal path, resulting in increased sample complexity, (c) Our BEE operator, however, balances between exploitation and exploration, improving convergence and accuracy. And (d) reveals the optimal Q and actions.

when the replay buffer is filled with high-performance samples. Such an observation indicates a notable shortfall of existing methods in exploiting the good samples in the replay buffer. In light of this, allowing the RL agent to swiftly seize the serendipities, i.e., luckily, successful experiences can be a natural cure to resolve the underestimation issue, as illustrated in Figure 4. Then, we shift our focus to devise a method for utilizing the optimal actions from the replay buffer as target-update actions to boost Q-value estimation.

4.3. Blended Exploitation and Exploration operator

To extract the best-performing actions from the replay buffer for updating the Q-target value, we consider the policy mixture μ induced by the replay buffer, which contains many samples and varies per policy iteration. Based on μ, we introduce the Bellman Exploitation operator $\mathcal{T}_{exploit}$:

$$\mathcal{T}^{\mu}_{exploit}Q(s,a) = r(s,a) + \gamma \cdot \max_{a' \in \mathcal{A}, \mu(a'|s')>0} \mathbb{E}_{s' \sim P(s'|s,a)}[Q(s',a')]. \quad (1)$$

It yields a Q-value estimation that is less affected by the optimality level of the current policy. Several offline RL methods (Kostrikov et al., 2021; Xu et al., 2023; Garg et al., 2023) have also focused on computing $\max Q$ constrained to the support of a pre-collected dataset for Bellman update, yet rely on a stationary behavior policy, which could be viewed as a reduced form of the $\mathcal{T}_{exploit}$ operator.

Meanwhile, to maintain the exploration optimism, we utilize the general Bellman Exploration operator. Here, $\omega(s',a'|\pi)$ refers to a chosen exploration term.

$$\mathcal{T}^{\pi}_{explore}Q(s,a) = r(s,a) + \gamma \cdot \mathbb{E}_{s' \sim P(s'|s,a)}\mathbb{E}_{a' \sim \pi(a'|s')} [Q(s',a') - \omega(s',a'|\pi)] \quad (2)$$

With the Bellman Exploitation and Bellman Exploration operators, which respectively capitalize on past successes and promote the exploration of uncertain regions, we aim to take the merits of both.

While there are numerous ways to blend the two operators, a simple yet efficient algorithm has long been advocated (Fujimoto & Gu, 2021). We propose the BEE operator, employing a linear combination to regulate the balance, aligns well with the requirement. The $\mathcal{T}_{exploit}$ component is particularly vital, as it provides a reference Q-value, thereby improving Q-value estimation. Later in the paper, extensive experiments showcase the efficiency of the BEE operator.

Definition 4.1. The Blended Exploitation and Exploration (**BEE**) Bellman operator $\mathcal{B}$ is defined as:

$$\mathcal{B}^{\{\mu,\pi\}}Q(s,a) = \lambda \cdot \mathcal{T}^{\mu}_{exploit}Q(s,a) + (1-\lambda) \cdot \mathcal{T}^{\pi}_{explore}Q(s,a) \quad (3)$$

Here, μ is the policy mixture, π is the current policy, and $\lambda \in (0,1)$ is a trade-off hyperparameter.

The choice of λ in Eq.(3) impacts the exploitation-exploration trade-off, as shown in Figure 5. Besides choosing a fixed number, λ can also be autonomously and adaptively tuned with multiple methods as detailed in Appendix C.3.3. The single-operator design incurs comparable computational costs to general-purpose algorithms such as SAC (Haarnoja et al., 2018a), and is relatively lightweight compared to other methods that require training a large number of Q-networks to tackle the exploration-exploitation dilemma (Ciosek et al., 2019; Chen et al., 2021).

4.4. Superior Q-value estimation using BEE operator

For a better understanding of the BEE operator, we conduct a theoretical analysis of its dynamic programming properties in the tabular MDP setting, covering policy evaluation, policy improvement, and policy iteration. All proofs are included in Appendix B.

Proposition 4.2 (Policy evaluation). *Consider an initial $Q_0 : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ with $|\mathcal{A}| < \infty$, and define $Q_{k+1} = \mathcal{B}^{\{\mu,\pi\}}Q_k$. Then the sequence $\{Q_k\}$ converges to a fixed point $Q^{\{\mu,\pi\}}$ as $k \to \infty$.*

Proposition 4.3 (Policy improvement). *Let $\{\mu_k, \pi_k\}$ be*

Algorithm 1 BEE Actor-Critic (BAC)

initialize: Q networks Q_ϕ, policy π_θ, replay buffer $\mathcal{D}$;
for policy training steps $t = 1, 2, \cdots, T$ **do**
  Sample N transitions (s, a, r, s') from $\mathcal{D}$
  Compute $\mathcal{T}_{exploit} Q_\phi$ by Eq.(1) {multiple design choices available}
  Compute $\mathcal{T}_{explore} Q_\phi$ by Eq.(2) {with a chosen exploration term $\omega(\cdot|\pi_\theta)$}
  Calculate the target Q value: $\mathcal{B}Q_\phi \leftarrow \lambda \mathcal{T}_{exploit} Q_\phi + (1-\lambda)\mathcal{T}_{explore} Q_\phi$
  for each environment step **do**
    Collect (s, a, s', r) with π_θ from real environment; add to $\mathcal{D}$
  end for
  for each gradient step **do**
    Update Q_ϕ by $\min_\phi (\mathcal{B}Q_\phi - Q_\phi)^2$
    Update π_θ by $\max_\theta Q_\phi(s, \pi_\theta)$
  end for
end for

the policies at iteration k, and $\{\mu_{k+1}, \pi_{k+1}\}$ be the updated policies, where π_{k+1} is the greedy policy of the Q-value. Then for all $(s, a) \in \mathcal{S} \times \mathcal{A}$, $|\mathcal{A}| < \infty$, we have $Q^{\{\mu_{k+1}, \pi_{k+1}\}}(s, a) \geq Q^{\{\mu_k, \pi_k\}}(s, a)$.

Proposition 4.4 (**Policy iteration**). *Assume $|\mathcal{A}| < \infty$, by repeating iterations of the policy evaluation and policy improvement, any initial policies converge to the optimal policies $\{\mu^*, \pi^*\}$, s.t. $Q^{\{\mu^*, \pi^*\}}(s_t, a_t) \geq Q^{\{\mu', \pi'\}}(s_t, a_t), \forall \mu' \in \Pi, \pi' \in \Pi, \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$.*

With the approximate dynamic programming properties established, our BEE operator is well-defined and flexible and could be integrated into various off-policy actor-critic algorithms. In Appendix G, we show that the BEE operator would alleviate underestimation without inciting additional overestimation, thus facilitating the estimation of Q and improving learning efficiency.

4.5. Algorithmic instantiation

We now describe two practical algorithmic instantiations based on the BEE operator $\mathcal{B}$ for both model-free and model-based RL paradigms, namely BEE Actor-Critic (BAC) and Model-Based BAC (MB-BAC), respectively. The implementation of our methods requires the specification of two main design choices: 1) a practical way to optimize the objective value on the Bellman Exploitation operator, and 2) a specific choice on the exploration term $\omega(\cdot|\pi)$ in the Bellman Exploration operator.

To effectively compute the $\max Q$-target value in Eq.(1) subject to the samples in the replay buffer, we utilize the in-sample learning objectives (Kostrikov et al., 2021; Garg et al., 2023; Xu et al., 2023) to learn the maximum Q-value over actions in the replay buffer. This treatment not only avoids the explicit computation of the policy mixture μ of replay buffer but also promotes the stability of Q estimation by only extracting actions that have been previously encountered for the Bellman update.

For the exploration term $\omega(\cdot|\pi_\theta)$, numerous options have been extensively explored in prior off-policy actor-critic methods (Haarnoja et al., 2018a; Han & Sung, 2021; Eberhard et al., 2023). Here, we employ the entropy regularization term from SAC to compute $\mathcal{T}_{explore} Q_\phi(s, a)$, where actions a' for target updating are extracted from π_θ. For extensive design choices for BAC see Appendix C.3.

Integration into Dyna-style model-based RL. Our method could be invoked into the Dyna-style model-based RL (MBRL) framework (Sutton, 1990; 1991; Kurutach et al., 2018; Buckman et al., 2018; Luo et al., 2018). As observed in (Luo et al., 2018; Lambert et al., 2020; Ghugare et al., 2023), a better policy optimizer could potentially further enhance the algorithm's performance, this motivates us to incorporate the BEE operator in existing model-based approaches. We propose a modification to the general Dyna-style algorithm, where the standard Q-value update rule is replaced with our BEE operator, resulting in the Model-based BAC (MB-BAC) algorithm.

In contrast to previous methods that utilize SAC as policy optimization backbone (Janner et al., 2019; Lai et al., 2021; Pan et al., 2020; Ji et al., 2022), MB-BAC treats real and model-generated data differently. It applies the $\mathcal{T}_{exploit}$ to real data $\mathcal{D}_e$, capitalizing on optimal real experiences while employing the $\mathcal{T}_{explore}$ on model rollout data $\mathcal{D}_m$ to explore new possibilities. This approach enhances the effective use of valuable real data and fosters exploration in new regions of the state space. The practical implementation builds upon MBPO (Janner et al., 2019) by integrating the BAC as policy optimizer, with the pseudocode in Appendix C.2.

5. Experiments

Our experimental evaluation aims to investigate the following questions: 1) How effective is the proposed BEE operator in model-based and model-free paradigms? 2) How effectively does BAC perform in failure-prone scenarios, that highlight the ability to seize serendipity and fleeting successes, particularly in various real-world tasks?

5.1. Evaluation on various standard benchmarks

To illustrate the effectiveness of the BEE operator across both model-based and model-free paradigms, we evaluate BAC and MB-BAC on various continuous control tasks.

Notably, BAC demonstrates superior performance compared to popular model-free methods across **MuJoCo** (Todorov

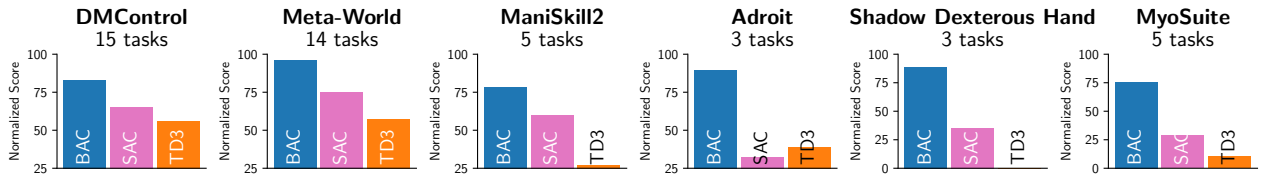


Figure 6. **Performance overview over six benchmarks.** BAC compares favorably to popular model-free methods across DMControl, Meta-World, ManiSkill2, Adroit, Shadow Dexterous Hand, and MyoSuite benchmark tasks with a single set of hyperparameters.

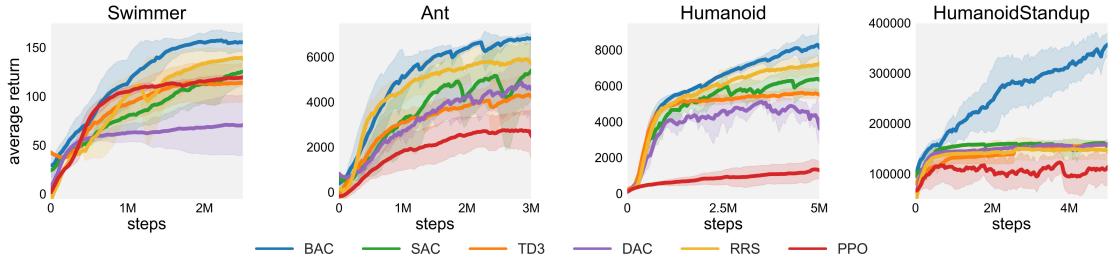


Figure 7. Training curves of BAC and five baselines on four continuous control benchmarks. Solid curves depict the mean of ten trials, and shaded regions correspond to the one standard deviation. To ensure a fair comparison, a uniform discount factor of $\gamma = 0.99$ is used for all baselines and tasks. Notably, for the Swimmer task, a significantly higher discount factor of $\gamma = 0.9999$ greatly enhances performance; these results are provided in the Appendix J.7.

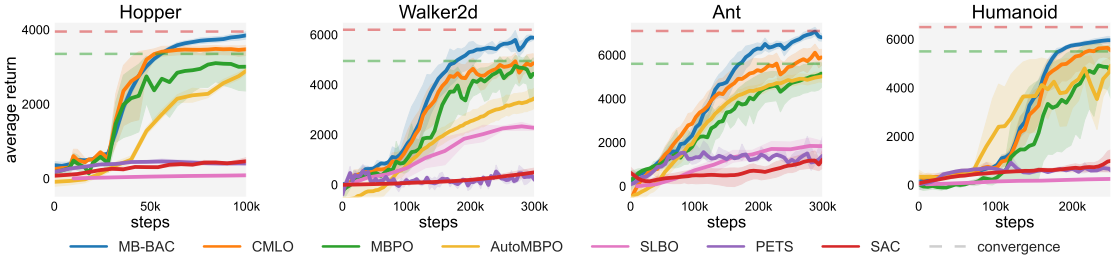


Figure 8. Training curves of MB-BAC and six baselines on four continuous control benchmarks, averaged over ten trials. The dashed lines are the asymptotic performance of SAC (up to 3M) and MBPO.

et al., 2012), **DMControl** (Tunyasuvunakool et al., 2020), **Meta-World** (Yu et al., 2019), **ManiSkill2** (Gu et al., 2023), **Adroit** (Rajeswaran et al., 2017), **MyoSuite** (Vittorio et al., 2022), **ROBEL** (Ahn et al., 2020) benchmark tasks, and even show effectiveness in sparse reward tasks and noisy environments. We summarize the performance over six benchmarks in Figure 6. Detailed performance curves on these benchmark suites are in Appendix J.

Comparison of model-free methods. We compare BAC to several popular model-free baselines, including: 1) SAC, regarded as the most popular off-policy actor-critic method; 2) TD3, which introduces the Double Q-learning trick to reduce function approximation error; 3) Diversity Actor-Critic (DAC) (Han & Sung, 2021), a variant of SAC, using a sample-aware entropy regularization instead, which is a potential choice for our $\omega(\cdot|s,a)$; 4) Random Reward Shift (RRS) (Sun et al., 2022), which learns multiple value functions (seven double-Q networks) with different shifting constants for the exploration and exploitation trade-off; 5) PPO (Schulman et al., 2017), a stable on-policy algorithm that discards historical policies.

We evaluate BAC and the baselines on a set of MuJoCo continuous control tasks. BAC surpasses all baselines in terms of eventual performance, coupled with better sample efficiency, as shown in Figure 7. Notably, the Humanoid-Standup task, known for its high action dimension and susceptibility to failure (Han & Sung, 2021), requires the algorithms to be able to seize and value serendipity. In this task, BAC gains a significantly better performance, with average returns up to 280,000 at 2.5M steps and 360,000 at 5M steps, which is 1.5x and 2.1x higher than the strongest baseline, respectively. This echoes the hypothesis that BAC exploits past serendipities in failure-prone environments. Trajectory visualizations in Figure 45 show that BAC agent could swiftly reach a stable standing, while the SAC agent ends up with a wobbling kneeling posture, the DAC agent sitting on the ground, and the RRS agent rolling around.

Experimental results on the more failure-prone tasks are

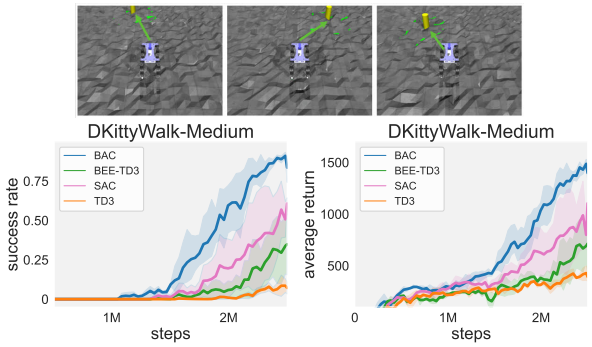


Figure 9. Success rate and average return in DKittyWalk-Medium.

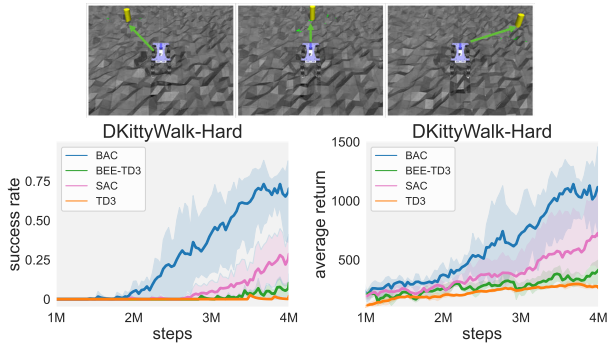
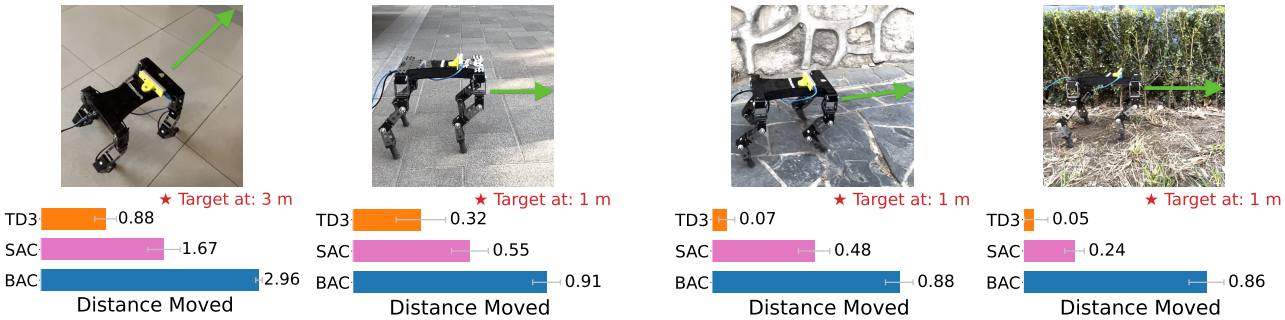


Figure 10. Success rate and average return in DKittyWalk-Hard.



(a) Smooth road (b) Rough stone road (c) Uphill stone road (d) Grassland

Figure 11. **Comparisons on four challenging real-world tasks.** The bar plots show how far the agent walks toward the goal for each algorithm averaged over ten runs. For (a) and (b), we employ the policy trained in the -Medium task, and for (c) and (d) use the policy trained in the -Hard task.

in Appendix H. We note that BAC is the ***first documented model-free method*** of solving the complex Dog tasks of DMControl. Additionally, we integrate our BEE into the TD3 algorithm and find that the ad-hoc BEE-TD3 also outperforms the original TD3 in fifteen DMControl tasks, refer to Appendix J.1.

Comparison of model-based methods. We evaluate the performance of MB-BAC, which integrates the BEE operator into the MBPO algorithm, against several model-based and model-free baselines. Among the Dyna-style counterparts, MBPO (Janner et al., 2019), CMLO (Ji et al., 2022), and AutoMBPO (Lai et al., 2021) use SAC as the policy optimizer, while SLBO (Luo et al., 2018) employs TRPO (Schulman et al., 2015). PETS (Chua et al., 2018) is a planning-based method that utilizes CEM (Botev et al., 2013) as the planner. Figure 8 showcases that MB-BAC learns faster than other modern model-based RL methods and yields promising asymptotic performance compared with model-free counterparts. Moreover, the result highlights the universality of the BEE operator.

5.2. Evaluation in real-world quadruped robot tasks

We evaluate BAC on a real quadruped robot D'Kitty (Ahn et al., 2020). We follow the sim2real paradigm as in previous legged locomotion works (Agarwal et al., 2023; Hwangbo et al., 2019; Tan et al., 2018) where the agent is trained in simulated environments with randomized terrains and then deployed in the real world without further training. The task is challenging, as agents are prone to falling due to fluctuating terrain. As for real-world scenarios, the D'Kitty robot is required to traverse various complex terrains, contending with unpredictable environmental factors.

Firstly, we construct two simulation task variants, DKittyWalk-Medium and DKittyWalk-Hard. The -Medium variant features a random height region of 0.07m, while the -Hard variant has a height of 0.09m, which is 1.4 times and 1.8 times higher than the base task DKittyWalkRandomDynamics, respectively. Given D'Kitty's leg length of around 0.3m when standing, navigating uneven terrain with height variations of over 0.2x to 0.3x the leg length poses a significant challenge, as a deviation of 0.02m would lead to a considerable shift in the center of gravity. Figure 9 and 10 demonstrate that BAC outperforms other algorithms in both tasks with clearer advantages. BAC achieves a success rate surpassing SAC by approximately 50%. The ad-hoc BEE-TD3 also outperforms the TD3.

More crucially, BAC achieves superior performance when deployed in the real world across various terrains, as shown in Figure 11. The policy learned in the -Medium variant is

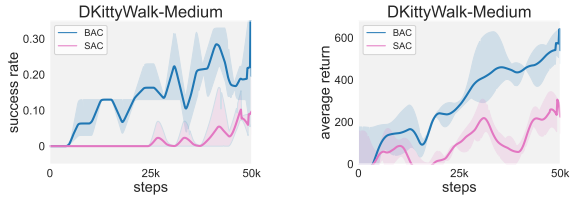


Figure 12. Comparison of the ability to seize serendipity in DKittyWalk-Medium.*Left*: success rate; *Right*: average return.

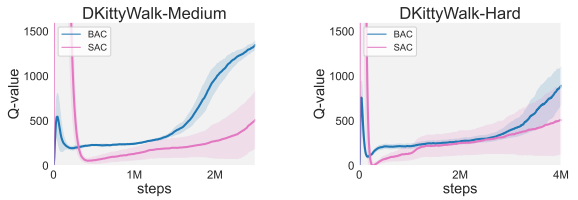


Figure 13. Q-value learning stability comparison. 10 seeds. The lower variance observed with BAC indicates better learning stability across runs.

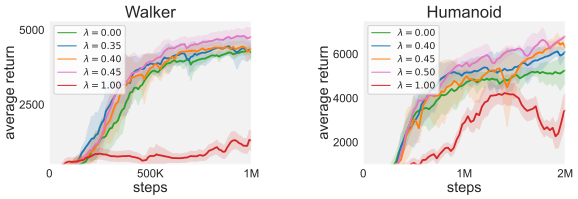


Figure 14. Hyperparameter study on λ.

deployed on two terrains — smooth road and rough stone road, with target points positioned at distances of 3m and 1m, respectively. For more challenging terrains — uphill stone roads and grasslands, we employ the policy trained in the -Hard variant, with a target point located 1m ahead. Specifically, the BAC algorithm outperformed the TD3 and SAC agents in achieving stable movement across a variety of terrains and displaying natural gaits. In contrast, the TD3 agent prefers lower postures, such as knee walking, which makes it prone to falling on uneven terrain, while the SAC agent suffers from more oscillatory gait patterns (videos are provided in the project website0. The empirical results also shed light on the necessity of algorithmic improvement for real-world robotics in addition to building better environments and designing informative rewards.

5.3. Ablation studies

Ability to seize serendipity. To better understand how well the BEE operator captures past well-performing actions, we conduct experiments on the DKittyWalk-Medium task. We initialize SAC and BAC with the identical Q network, random policy, and replay buffer. Next, we collected 15 trajectories (2400 transitions in total) using an expert policy whose success rate is 100% and adding them to the replay buffer. Keeping all components and parameters the same as in the main experiment, we train BAC and SAC on the blended buffer harboring several successful actions. Figure 12 suggests that BAC recovers success faster than SAC, indicating its supposed ability to seize serendipity.

More stable Q-value in practice. In failure-prone scenarios, policy performance typically severely oscillates across iterations due to easily encountered failure samples from the current policy in Q-value update if using the Bellman evaluation operator. The Q-value learned by the BEE operator is less affected by the optimality level of the current policy, thus it might be expected to have better learning stability. The smaller error bar in Figure 13 supports it. To be specific, this stability is especially noteworthy in environments prone to failure, where policy oscillations are significantly influenced by environmental randomness (including seed variations). The Q-values derived from BAC are less susceptible to the immediate quality of the current policy, thus exhibiting lower variance across multiple runs.

Hyperparameter study. Setting an appropriate weighted coefficient λ, BAC could balance the exploitation and exploration well. We may note that the algorithm is reduced to the online version of IQL (Kostrikov et al., 2021) for an extreme value $\lambda = 0$. According to Figure 14, and the detailed settings and hyperparameter studies in Appendix C.4, we find that a moderate choice of λ around 0.5 is sufficient to achieve the desired performance across all 35 locomotion and manipulation tasks we have benchmarked. This underscores that BAC does not need heavy tuning for strong performance.

In-depth ablation studies in terms of the ability to counteract failure, effectiveness in noisy environments, and performance in sparse-reward tasks, are presented in Appendix H.

6. Conclusion

In this paper, we investigate the overlooked issue of value underestimation in off-policy actor-critic methods and find that incorporating sufficient exploitation might mitigate this issue. These observations motivate us to propose the Blended Exploitation and Exploration (BEE) operator, which leverages the value of past successes to enhance Q-value estimation and policy learning. The proposed algorithms BAC and MB-BAC outperform both model-based and model-free methods across various continuous control tasks. Remarkably, without further training, BAC shines in real-robot tasks, emphasizing the need for improved general-purpose algorithms in real-world robotics. Finally, our work sheds light on future work on fully fusing exploitation and exploration techniques, *e.g.*, incorporating up-to-date design choices for computing $\max Q$ or exploration term, in building strong RL methods.

Impact Statement

BAC is almost as lightweight as SAC, with only a slight increase in computation time. This increase is acceptable, especially since BAC often requires fewer interactions to achieve similar performance, potentially reducing overall computation time. Besides, as with any off-policy RL algorithm, safety measures are necessary to prevent unsafe behavior during real-world exploration.

This research advances both the cognition and the application of Reinforcement Learning, particularly in the domain of off-policy actor-critic framework.

It sheds light on a critical yet previously underappreciated factor that underpins the underestimation bias in the latter stages of online RL training. Through extensive research, this work provides new insights into this previously overlooked aspect of RL. Moreover, this work demystifies the misperception that offline RL is over-conservative and incompatible with the online RL setting. It suggests a new paradigm that incorporates exploitation ingredients from offline RL to enhance pure online RL.

The elegance of the proposed algorithm lies in its simplicity and adaptability. It demonstrates exceptional performance across a broad spectrum of tasks with just one simple set of hyperparameters. This positions it as a strong candidate for a backbone algorithm in various RL applications, showing promising possibilities for deployment in real-world contexts, notably in the field of robotics.

Acknowledgements

This work was funded by the National Science and Technology Major Project of the Ministry of Science and Technology of China (2018AAA0102903), partly by Tsinghua University-Toyota Joint Research Center for Al Technology of Automated Vehicles (TTAD 2024-03). The authors would also like to thank the anonymous reviewers for their careful reading and their many insightful comments.

References

Agarwal, A., Kumar, A., Malik, J., and Pathak, D. Legged locomotion in challenging terrains using egocentric vision. In *Conference on Robot Learning*, 2023.

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A., and Bellemare, M. G. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems*, 2021.

Ahn, M., Zhu, H., Hartikainen, K., Ponte, H., Gupta, A., Levine, S., and Kumar, V. Robel: Robotics benchmarks for learning with low-cost robots. In *Conference on robot learning*, 2020.

Anschel, O., Baram, N., and Shimkin, N. Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In *International conference on machine learning*, pp. 176–185. PMLR, 2017.

Auer, P. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

Auer, P., Jaksch, T., and Ortner, R. Near-optimal regret bounds for reinforcement learning. In *Advances in Neural Information Processing Systems*, 2008.

Aytar, Y., Pfaff, T., Budden, D., Paine, T., Wang, Z., and De Freitas, N. Playing hard exploration games by watching youtube. In *Advances in Neural Information Processing Systems*, 2018.

Azar, M. G., Osband, I., and Munos, R. Minimax regret bounds for reinforcement learning. In *International Conference on Machine Learning*, 2017.

Badia, A. P., Sprechmann, P., Vitvitskyi, A., Guo, D., Piot, B., Kapturowski, S., Tieleman, O., Arjovsky, M., Pritzel, A., Bolt, A., et al. Never give up: Learning directed exploration strategies. In *International Conference on Learning Representations*, 2020.

Bai, C., Wang, L., Han, L., Hao, J., Garg, A., Liu, P., and Wang, Z. Principled exploration via optimistic bootstrapping and backward induction. In *International Conference on Machine Learning*, 2021.

Baker, B., Akkaya, I., Zhokov, P., Huizinga, J., Tang, J., Ecoffet, A., Houghton, B., Sampedro, R., and Clune, J. Video pretraining (vpt): Learning to act by watching unlabeled online videos. In *Advances in Neural Information Processing Systems*, 2022.

Botev, Z. I., Kroese, D. P., Rubinstein, R. Y., and L'Ecuyer, P. The cross-entropy method for optimization. In *Handbook of statistics*, volume 31, pp. 35–59. Elsevier, 2013.

Brafman, R. I. and Tennenholtz, M. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct): 213–231, 2002.

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

Buckman, J., Hafner, D., Tucker, G., Brevdo, E., and Lee, H. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *Advances in Neural Information Processing Systems*, 2018.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019.

Casas, N. Deep deterministic policy gradient for urban traffic light control. *arXiv preprint arXiv:1703.09035*, 2017.

Cetin, E. and Celiktutan, O. Learning pessimism for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

Chan, A., Silva, H., Lim, S., Kozuno, T., Mahmood, A. R., and White, M. Greedification operators for policy optimization: Investigating forward and reverse kl divergences. *The Journal of Machine Learning Research*, 23 (1):11474–11552, 2022.

Chen, X., Wang, C., Zhou, Z., and Ross, K. W. Randomized ensembled double q-learning: Learning fast without a model. In *International Conference on Learning Representations*, 2021.

Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, 2018.

Ciosek, K., Vuong, Q., Loftin, R., and Hofmann, K. Better exploration with optimistic actor critic. In *Advances in Neural Information Processing Systems*, 2019.

D'Oro, P., Schwarzer, M., Nikishin, E., Bacon, P.-L., Bellemare, M. G., and Courville, A. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *The Eleventh International Conference on Learning Representations*, 2022.

Eberhard, O., Hollenstein, J., Pinneri, C., and Martius, G. Pink noise is all you need: Colored noise exploration in deep reinforcement learning. In *International Conference on Learning Representations*, 2023.

Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K. O., and Clune, J. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K. O., and Clune, J. First return, then explore. *Nature*, 590(7847): 580–586, 2021.

Franceschetti, M., Lacoux, C., Ohouens, R., Raffin, A., and Sigaud, O. Making reinforcement learning work on swimmer. *arXiv preprint arXiv:2208.07587*, 2022.

Fruit, R., Pirotta, M., Lazaric, A., and Ortner, R. Efficient bias-span-constrained exploration-exploitation in reinforcement learning. In *International Conference on Machine Learning*, 2018.

Fujimoto, S. and Gu, S. S. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Fujimoto, S., van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. *Proceedings of Machine Learning Research*, 80:1587–1596, 2018.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, 2019.

Gallouédec, Q., Cazin, N., Dellandréa, E., and Chen, L. panda-gym: Open-source goal-conditioned environments for robotic learning. *arXiv preprint arXiv:2106.13687*, 2021.

Garg, D., Hejna, J., Geist, M., and Ermon, S. Extreme q-learning: Maxent rl without entropy. In *International Conference on Learning Representations*, 2023.

Ghugare, R., Bharadhwaj, H., Eysenbach, B., Levine, S., and Salakhutdinov, R. Simplifying model-based rl: Learning representations, latent-space models, and policies with one objective. In *International Conference on Learning Representations*, 2023.

Grondman, I., Busoniu, L., Lopes, G. A., and Babuska, R. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307, 2012.

Gu, J., Xiang, F., Li, X., Ling, Z., Liu, X., Mu, T., Tang, Y., Tao, S., Wei, X., Yao, Y., et al. Maniskill2: A unified benchmark for generalizable manipulation skills. *arXiv preprint arXiv:2302.04659*, 2023.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018a.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

Han, S. and Sung, Y. Diversity actor-critic: Sample-aware entropy regularization for sample-efficient exploration. In *International Conference on Machine Learning*, 2021.

Hansen, N., Wang, X., and Su, H. Temporal difference learning for model predictive control. In *ICML*, 2022.

Hansen-Estruch, P., Kostrikov, I., Janner, M., Kuba, J. G., and Levine, S. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

Hasselt, H. Double q-learning. *Advances in neural information processing systems*, 23, 2010.

Hazan, E., Kakade, S., Singh, K., and Van Soest, A. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, 2019.

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Osband, I., et al. Deep q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

Hutter, M., Gehring, C., Jud, D., Lauber, A., Bellicoso, C. D., Tsounis, V., Hwangbo, J., Bodie, K., Fankhauser, P., Bloesch, M., et al. Anymal-a highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ international conference on intelligent robots and systems*, 2016.

Hwangbo, J., Lee, J., Dosovitskiy, A., Bellicoso, D., Tsounis, V., Koltun, V., and Hutter, M. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

Ishfaq, H., Cui, Q., Nguyen, V., Ayoub, A., Yang, Z., Wang, Z., Precup, D., and Yang, L. Randomized exploration in reinforcement learning with general value function approximation. In *International Conference on Machine Learning*, 2021.

Islam, R., Ahmed, Z., and Precup, D. Marginalized state distribution entropy regularization in policy optimization. *arXiv preprint arXiv:1912.05128*, 2019.

Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, 2019.

Ji, T., Luo, Y., Sun, F., Jing, M., He, F., and Huang, W. When to update your model: Constrained model-based reinforcement learning. *Advances in Neural Information Processing Systems*, 35:23150–23163, 2022.

Jin, C., Allen-Zhu, Z., Bubeck, S., and Jordan, M. I. Is q-learning provably efficient? In *Advances in Neural Information Processing Systems*, 2018.

Kim, H., Kim, J., Jeong, Y., Levine, S., and Song, H. O. Emi: Exploration with mutual information. In *International Conference on Machine Learning*, 2019.

Kimura, H., Kobayashi, S., et al. An analysis of actor/critic algorithms using eligibility traces: Reinforcement learning with imperfect value function. In *International Conference on Machine Learning*, 1998.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2021.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, 2019.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

Kurutach, T., Clavera, I., Duan, Y., Tamar, A., and Abbeel, P. Model-ensemble trust-region policy optimization. In *International Conference on Learning Representations*, 2018.

Kuznetsov, A., Shvechikov, P., Grishin, A., and Vetrov, D. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In *International Conference on Machine Learning*, pp. 5556–5566, 2020.

Lai, H., Shen, J., Zhang, W., Huang, Y., Zhang, X., Tang, R., Yu, Y., and Li, Z. On effective scheduling of model-based reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021.

Lambert, N., Amos, B., Yadan, O., and Calandra, R. Objective mismatch in model-based reinforcement learning. *arXiv preprint arXiv:2002.04523*, 2020.

Lan, Q., Pan, Y., Fyshe, A., and White, M. Maxmin q-learning: Controlling the estimation bias of q-learning. *arXiv preprint arXiv:2002.06487*, 2020.

Laskin, M., Lee, K., Stooke, A., Pinto, L., Abbeel, P., and Srinivas, A. Reinforcement learning with augmented data. In *Advances in Neural Information Processing Systems*, 2020.

Lee, A. X., Nagabandi, A., Abbeel, P., and Levine, S. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. In *Advances in Neural Information Processing Systems*, 2020.

Lee, D., Defourny, B., and Powell, W. B. Bias-corrected q-learning to control max-operator bias in q-learning. In *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pp. 93–99. IEEE, 2013.

Lee, L., Eysenbach, B., Parisotto, E., Xing, E., Levine, S., and Salakhutdinov, R. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

Lee, S., Seo, Y., Lee, K., Abbeel, P., and Shin, J. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, 2022.

Li, J., Zhan, X., Xu, H., Zhu, X., Liu, J., and Zhang, Y.-Q. When data geometry meets deep function: Generalizing offline reinforcement learning. In *International Conference on Learning Representations*, 2022.

Liu, X.-H., Xue, Z., Pang, J., Jiang, S., Xu, F., and Yu, Y. Regret minimization experience replay in off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021.

Luo, Y., Xu, H., Li, Y., Tian, Y., Darrell, T., and Ma, T. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. In *International Conference on Learning Representations*, 2018.

Lyu, J., Ma, X., Li, X., and Lu, Z. Mildly conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2022.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 2016.

Moskovitz, T., Parker-Holder, J., Pacchiano, A., Arbel, M., and Jordan, M. Tactical optimism and pessimism for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021.

Nair, A., Gupta, A., Dalal, M., and Levine, S. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Nikolov, N., Kirschner, J., Berkenkamp, F., and Krause, A. Information-directed exploration for deep reinforcement learning. In *International Conference on Learning Representations*, 2019.

Niu, H., Qiu, Y., Li, M., Zhou, G., HU, J., Zhan, X., et al. When to trust your simulator: Dynamics-aware hybrid offline-and-online reinforcement learning. In *Advances in Neural Information Processing Systems*, 2022.

Ostrovski, G., Bellemare, M. G., Oord, A., and Munos, R. Count-based exploration with neural density models. In *International Conference on Machine Learning*, 2017.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, 2022.

Pan, F., He, J., Tu, D., and He, Q. Trust the model when it is confident: Masked model-based actor-critic. In *Advances in Neural Information Processing Systems*, 2020.

Pathak, D., Gandhi, D., and Gupta, A. Self-supervised exploration via disagreement. In *International Conference on Machine Learning*, 2019.

Peer, O., Tessler, C., Merlis, N., and Meir, R. Ensemble bootstrapping for q-learning. In *International Conference on Machine Learning*, 2021.

Peters, J. and Schaal, S. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.

Pineda, L., Amos, B., Zhang, A., Lambert, N. O., and Calandra, R. Mbrl-lib: A modular library for model-based reinforcement learning. *Arxiv*, 2021. URL https://arxiv.org/abs/2104.10159.

Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., Kumar, V., and Zaremba, W. Multi-goal reinforcement learning: Challenging robotics environments and request for research, 2018.

Pomerleau, D. A. Alvinn: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems*, 1988.

pranz24. pytorch-soft-actor-critic. https://github.com/pranz24/pytorch-soft-actor-critic, 2018.

Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., and Levine, S. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *Artificial Intelligence and Statistics*, 2011.

Schaal, S. Learning from demonstration. In *Advances in Neural Information Processing Systems*, 1996.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International Conference on Machine Learning*, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shah, R. M. and Kumar, V. Rrl: Resnet as representation for reinforcement learning. In *International Conference on Machine Learning*, 2021.

Shao, L., You, Y., Yan, M., Yuan, S., Sun, Q., and Bohg, J. Grac: Self-guided and self-regularized actor-critic. In *Conference on Robot Learning*, pp. 267–276. PMLR, 2022.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Sinha, S., Song, J., Garg, A., and Ermon, S. Experience replay with likelihood-free importance weights. In *Learning for Dynamics and Control Conference*, 2022.

Sun, H., Han, L., Yang, R., Ma, X., Guo, J., and Zhou, B. Optimistic curiosity exploration and conservative exploitation with linear reward shaping. In *Advances in Neural Information Processing Systems*, 2022.

Sutton, R. S. Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44, 1988.

Sutton, R. S. Integrated architecture for learning, planning, and reacting based on approximating dynamic programming. In *International Conference on Machine Learning*, 1990.

Sutton, R. S. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

Szita, I. and Lőrincz, A. The many faces of optimism: a unifying approach. In *International conference on Machine learning*, 2008.

Tan, J., Zhang, T., Coumans, E., Iscen, A., Bai, Y., Hafner, D., Bohez, S., and Vanhoucke, V. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*, 2018.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems*, 2012.

Tunyasuvunakool, S., Muldal, A., Doron, Y., Liu, S., Bohez, S., Merel, J., Erez, T., Lillicrap, T., Heess, N., and Tassa, Y. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

Vittorio, C., Huawei, W., Guillaume, D., Massimo, S., and Vikash, K. Myosuite – a contact-rich simulation suite for musculoskeletal motor control. https://github.com/myohub/myosuite, 2022. URL https://arxiv.org/abs/2205.13600.

Wang, C., Luo, X., Ross, K. W., and Li, D. Vrl3: A data-driven framework for visual deep reinforcement learning. In *Advances in Neural Information Processing Systems*, 2022a.

Wang, T., Bao, X., Clavera, I., Hoang, J., Wen, Y., Langlois, E., Zhang, S., Zhang, G., Abbeel, P., and Ba, J. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*, 2019.

Wang, X., Wongkamjan, W., and Huang, F. Live in the moment: Learning dynamics model adapted to evolving policy. In *Decision Awareness in Reinforcement Learning Workshop at ICML 2022*, 2022b.

Watkins, C. J. C. H. Learning from delayed rewards. *PhD thesis, Cambridge University, Cambridge, England*, 1989.

Xu, H., Jiang, L., Li, J., Yang, Z., Wang, Z., Chan, V. W. K., and Zhan, X. Offline rl with no ood actions: In-sample learning via implicit value regularization. In *International Conference on Learning Representations*, 2023.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, 2019.

Yu, W., Wang, R., Li, R., Gao, J., and Hu, X. Historical best q-networks for deep reinforcement learning. In *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 6–11. IEEE, 2018.

Zhan, W., Huang, B., Huang, A., Jiang, N., and Lee, J. Offline reinforcement learning with realizability and single-policy concentrability. In *Conference on Learning Theory*, 2022.

Zhang, C., Cai, Y., Huang, L., and Li, J. Exploration by maximizing rényi entropy for reward-free rl framework. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021a.

Zhang, H., Yu, H., Zhao, J., Zhang, D., Zhou, H., Huang, C., Ye, C., et al. How to fine-tune the model: Unified model shift and model bias policy optimization. *Advances in Neural Information Processing Systems*, 36, 2024.

Zhang, T., Rashidinejad, P., Jiao, J., Tian, Y., Gonzalez, J. E., and Russell, S. Made: Exploration via maximizing deviation from explored regions. In *Advances in Neural Information Processing Systems*, 2021b.

Zhang, Z., Pan, Z., and Kochenderfer, M. J. Weighted double q-learning. In *IJCAI*, pp. 3455–3461, 2017.

Zheng, L., Fiez, T., Alumbaugh, Z., Chasnov, B., and Ratliff, L. J. Stackelberg actor-critic: Game-theoretic reinforcement learning algorithms. In *AAAI conference on artificial intelligence*, 2022.

Contents

A Extensive Related Works **17**

B Omitted Proofs **18**

C Implementation Details and Extensive Design Choices **20**
C.1 Primary implementation details on BAC . . . 20
C.2 Primary implementation details on MB-BAC algorithm . . . 21
C.3 Possible design choices and extensions . . . 22
C.3.1 More design choices on computing $\mathcal{T}_{exploit}Q$ . . . 22
C.3.2 More design choices on exploration term $\omega(\cdot|\pi)$ . . . 22
C.3.3 Extensions: automatic adaptive λ mechanisms . . . 22
C.4 Hyperparameter settings . . . 24
C.5 Computing infrastructure and computational time . . . 24

D Environment Setup **26**
D.1 Environment setup for evaluating BAC . . . 26
D.2 Environment setup for evaluating MB-BAC . . . 29

E Baselines Implementation **30**

F Investigations on the Underestimation Issue **31**
F.1 Why underestimation and under-exploitation matters? . . . 31
F.2 What may cause the underestimation issue? . . . 31
F.3 How to mitigate underestimation issue? . . . 32
F.4 The existence of "under-exploitation" stage . . . 33
F.5 Explanations on the existence of under-exploitation circumstance . . . 33

G Superior Q-value Estimation using BEE Operator **35**

H Effectiveness in Failure-prone Scenarios **36**
H.1 The ability to counteract failure . . . 36
H.2 Effectiveness in noisy environments . . . 36
H.3 Illustrative example on the failure-prone scenario. . . . 37
H.4 Effectiveness in sparse-reward tasks . . . 38
H.5 Task visualizations in failure-prone scenarios . . . 39

I Additional Baselines Comparison **41**

J More Benchmark Results **42**
J.1 Evaluation on DMControl benchmark tasks . . . 42
J.2 Evaluation on Meta-World benchmark tasks . . . 44
J.3 Evaluation on Adroit benchmark tasks . . . 46
J.4 Evaluation on MyoSuite benchmark tasks . . . 46
J.5 Evaluation on ManiSkill2 benchmark tasks . . . 47
J.6 Evaluation on Shadow Dexterous Hand benchmark tasks . . . 47
J.7 Evaluation on Swimmer benchmark tasks with an extremely high γ. . . . 47

A. Extensive Related Works

Off-policy actor-critic methods leverage a replay buffer to update the Q-function and policy (Casas, 2017; Mnih et al., 2016), providing higher sample efficiency than on-policy RL methods. The prior works commonly rely on the standard policy gradient formulation (Peters & Schaal, 2008) for policy improvement. Various attempts have been devoted to modifying the policy evaluation procedure, primarily pursuing a high-quality value function to tackle the exploration or exploitation issue — central concerns in online RL (Burda et al., 2019; Ecoffet et al., 2019).

Despite the ongoing interest in exploration and exploitation, most previous works devoted to exploration design following the optimism principle in the face of uncertainty (Auer et al., 2008; Fruit et al., 2018; Szita & Lőrincz, 2008), but view exploitation as merely maximizing Q-function. The Bellman evaluation operator, $\mathcal{T}Q(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P} \mathbb{E}_{a' \sim \pi} Q(s',a')$, underpins the critic learning. Existing efforts can be summarized into modifying this operator $\mathcal{T}$ in three main ways: 1) perturbing action a' with techniques such as ϵ-greedy, target policy smoothing (Fujimoto et al., 2018), and pink noise (Eberhard et al., 2023); 2) augmenting reward r to foster exploration (Ostrovski et al., 2017; Burda et al., 2019; Badia et al., 2020; Zhang et al., 2021b); 3) directly adjusting Q values such as max-entropy RL methods (Zhang et al., 2021a; Hazan et al., 2019; Lee et al., 2019; Islam et al., 2019; Haarnoja et al., 2018a; Han & Sung, 2021) that infuse the operator with an entropy term, and optimistic exploration methods that learn Upper Confidence Bound (UCB) (Ishfaq et al., 2021; Auer, 2002; Nikolov et al., 2019) of ensemble Q-value networks (Bai et al., 2021; Ciosek et al., 2019; Moskovitz et al., 2021).

Reducing value estimation bias, both underestimation and overestimation, has been widely explored in value-based reinforcement learning (RL) methods (Lan et al., 2020; Anschel et al., 2017; Yu et al., 2018; Lee et al., 2013; Zhang et al., 2017). However, value estimation bias remains a significant issue in off-policy actor-critic methods.

In essence, value overestimation might be associated with optimistic exploration (Jin et al., 2018; Laskin et al., 2020; Moskovitz et al., 2021), alongside factors such as off-policy learning and high-dimensional, nonlinear function approximation. Hence, attempts to correct for overestimation, *e.g.*, taking the minimum of two separate critics, have been widely adopted in the above exploration-driven methods (Fujimoto et al., 2018; Haarnoja et al., 2018a; Han & Sung, 2021; Sun et al., 2022). Yet directly applying such a minimum may cause underestimation (Hasselt, 2010). To mitigate it, prior methods (Ciosek et al., 2019; Moskovitz et al., 2021) seek for a milder form, assuming the epistemic uncertainty as the standard deviation of ensemble Q values. We identify the value underestimation that particularly occurs in the latter training stages and uncover its long-neglected culprit. Our findings suggest that incorporating sufficient exploitation into current exploration-driven algorithms would be a natural solution and lead to an improved algorithm.

Experience Replay (ER) (Mnih et al., 2015) boosts exploitation in off-policy RL by enabling data reuse. Recent works in prioritized replay (Schaul et al., 2015; Liu et al., 2021; Sinha et al., 2022) propose various metrics to replay or reweight important transitions more frequently, benefiting sample efficiency. We primarily implement BAC with the vanilla ER method for simplicity, yet more advanced ER techniques could be integrated for further enhancement. Outside the online RL paradigm, imitation learning (Pomerleau, 1988; Schaal, 1996; Ross et al., 2011) and offline RL algorithms (Fujimoto et al., 2019; Kumar et al., 2019; 2020; Kostrikov et al., 2021; Zhan et al., 2022) are known for their effective exploitation of provided datasets. Although the prospect of integrating these techniques to enhance online RL is attractive, offline learning is often considered overly conservative and requires a reasonable-quality dataset for high performance (Li et al., 2022), leading to limited success in improving online learning (Niu et al., 2022). In standard online RL, we only have access to a dynamic and imperfect replay buffer, rather than a well-behaved dataset. As a result, recent efforts are mainly under a two-stage paradigm, integrating these techniques as policy pre-training for subsequent online training, such as initializing the policy with behavior cloning (Hester et al., 2018; Shah & Kumar, 2021; Wang et al., 2022a; Baker et al., 2022) or performing offline RL followed by online fine-tuning (Nair et al., 2020; Lee et al., 2022; Hansen-Estruch et al., 2023). By contrast, our work suggests a new paradigm that incorporates exploitation ingredients from offline RL to enhance pure online RL, as demonstrated in our proposed framework.

B. Omitted Proofs

Proposition B.1 (**Policy evaluation**). *Consider an initial $Q_0 : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ with $|\mathcal{A}| < \infty$, and define $Q_{k+1} = \mathcal{B}^{\{\mu,\pi\}} Q_k$. Then the sequence $\{Q_k\}$ converges to a fixed point $Q^{\{\mu,\pi\}}$ as $k \to \infty$.*

Proof. First, let us show that the BEE operator $\mathcal{B}$ is a γ-contraction operator in the $\mathcal{L}_\infty$ norm.

Let Q_1 and Q_2 be two arbitrary Q functions, for the Bellman Exploitation operator $\mathcal{T}_{exploit}$, since target-update actions a' are extracted from μ, we have that,

$$\begin{aligned}
\|\mathcal{T}^{\mu}_{exploit} Q_1 - \mathcal{T}^{\mu}_{exploit} Q_2\|_\infty =& \max_{s,a} |(r(s,a) + \gamma \mathbb{E}_{s'} \max_{a' \sim \mu}[Q_1(s',a')]) - (r(s,a) + \gamma \mathbb{E}_{s'} \max_{a' \sim \mu}[Q_2(s',a')])| \\
=& \gamma \max_{s,a} |\mathbb{E}_{s'}[\max_{a' \sim \mu} Q_1(s',a') - \max_{a' \sim \mu} Q_2(s',a')]| \\
\leq& \gamma \max_{s,a} \mathbb{E}_{s'} |\max_{a' \sim \mu} Q_1(s',a') - \max_{a' \sim \mu} Q_2(s',a')| \\
\leq& \gamma \max_{s,a} \|Q_1 - Q_2\|_\infty = \gamma \|Q_1 - Q_2\|_\infty
\end{aligned}$$

Also, for the Bellman Exploration Operator $\mathcal{T}_{explore}$, as $a' \sim \pi$, we have,

$$\begin{aligned}
\|\mathcal{T}^{\pi}_{explore} Q_1 - \mathcal{T}^{\pi}_{explore} Q_2\|_\infty =& \max_{s,a} |\gamma \mathbb{E}_{s'} \left[\mathbb{E}_{a' \sim \pi} Q_1(s',a') - \gamma \mathbb{E}_{a' \sim \pi} Q_2(s',a')\right]| \\
\leq& \gamma \max_{s,a} \mathbb{E}_{s'} |\mathbb{E}_{a' \sim \pi} Q_1(s',a') - \mathbb{E}_{a' \sim \pi} Q_2(s',a')| \\
\leq& \gamma \max_{s,a} \mathbb{E}_{s'} \mathbb{E}_{a' \sim \pi} |Q_1(s',a') - Q_2(s',a')| \\
\leq& \gamma \max_{s,a} \|Q_1 - Q_2\|_\infty = \gamma \|Q_1 - Q_2\|_\infty
\end{aligned}$$

Combining the results, we have that the BEE operator satisfies γ-contraction property:

$$\begin{aligned}
\|\mathcal{B}^{\{\mu,\pi\}} Q_1 - \mathcal{B}^{\{\mu,\pi\}} Q_2\|_\infty =& \|\lambda(\mathcal{T}^{\mu}_{exploit} Q_1 - \mathcal{T}^{\mu}_{exploit} Q_2) + (1-\lambda)(\mathcal{T}^{\pi}_{explore} Q_1 - \mathcal{T}^{\pi}_{explore} Q_2)\|_\infty \\
\leq& \lambda \|\mathcal{T}^{\mu}_{exploit} Q_1 - \mathcal{T}^{\mu}_{exploit} Q_2\|_\infty + (1-\lambda) \|\mathcal{T}^{\pi}_{explore} Q_1 - \mathcal{T}^{\pi}_{explore} Q_2\|_\infty \\
\leq& \lambda \gamma \|Q_1 - Q_2\|_\infty + (1-\lambda)\gamma \|Q_1 - Q_2\|_\infty = \gamma \|Q_1 - Q_2\|_\infty
\end{aligned}$$

we conclude that the BEE operator is a γ -contraction, which naturally leads to the conclusion that any initial Q function will converge to a unique fixed point by repeatedly applying $\mathcal{B}^{\{\mu,\pi\}}$.

□

Proposition B.2 (**Policy improvement**). *Let $\{\mu_k, \pi_k\}$ be the policies at iteration k, and $\{\mu_{k+1}, \pi_{k+1}\}$ be the updated policies, where π_{k+1} is the greedy policy of the Q-value. Then for all $(s,a) \in \mathcal{S} \times \mathcal{A}$, $|\mathcal{A}| < \infty$, we have $Q^{\{\mu_{k+1},\pi_{k+1}\}}(s,a) \geq Q^{\{\mu_k,\pi_k\}}(s,a)$.*

Proof. At iteration k, μ_k denotes the policy mixture and π_k the current policy, and the corresponding value function is $Q^{\{\mu,\pi\}}$. We firstly update the policies from $\{\mu_k, \pi_k\}$ to $\{\mu_k, \pi_{k+1}\}$, where π_{k+1} is the greedy policy w.r.t $J_{\pi_k,\mu_k}(\mu_k, \pi)$, *i.e.*, $\pi_{k+1} = \arg\max_\pi \mathbb{E}_{a \sim \pi}[Q^{\{\mu_k,\pi_k\}}(s,a) - \omega(s,a|\pi)]$.

We commence with the proof that $Q^{\{\mu_k,\pi_{k+1}\}}(s,a) \geq Q^{\{\mu_k,\pi_k\}}(s,a)$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$. Since $\pi_{k+1} = \arg\max_\pi J_{\pi_k,\mu_k}(\mu_k, \pi)$, we have that $J_{\pi_k,\mu_k}(\mu_k, \pi_{k+1}) \geq J_{\pi_k,\mu_k}(\mu_k, \pi_k)$. Expressing $J_{\pi_k,\mu_k}(\mu_k, \pi_{k+1})$ and $J_{\pi_k,\mu_k}(\mu_k, \pi_k)$ by their definition, we have $\mathbb{E}_{a \sim \pi_{k+1}}[Q^{\{\mu_k,\pi_k\}}(s,a) - \omega(s,a|\pi_{k+1})] \geq \mathbb{E}_{a \sim \pi_k}[Q^{\{\mu_k,\pi_k\}}(s,a) - \omega(s,a|\pi_k)]$.

In a similar way to the proof of the soft policy improvement (Haarnoja et al., 2018a), we come to the following inequality:

$$
\begin{aligned}
Q^{\{\mu_k,\pi_k\}}(s_t,a_t) =& r(s_t,a_t) + \gamma \mathbb{E}_{s_{t+1}}\big\{\lambda \cdot \max_{\tilde{a}_{t+1}\sim\mu_k} Q^{\{\mu_k,\pi_k\}}(s_{t+1},\tilde{a}_{t+1}) \\
&+ (1-\lambda)\cdot \mathbb{E}_{a_{t+1}\sim\pi_k}[Q^{\mu_k,\pi_k}(s_{t+1},a_{t+1}) - \omega(s_{t+1},a_{t+1}|\pi_k)]\big\} \\
\leq& r(s_t,a_t) + \gamma \mathbb{E}_{s_{t+1}}\{\lambda \cdot \max_{\tilde{a}_{t+1}\sim\mu_k} Q^{\{\mu_k,\pi_k\}}(s_{t+1},\tilde{a}_{t+1}) + \\
&(1-\lambda)\cdot \mathbb{E}_{a_{t+1}\sim\pi_{k+1}}[Q^{\{\mu_k,\pi_k\}}(s_{t+1},a_{t+1}) - \omega(s_{t+1},a_{t+1}|\pi_{k+1})]\} \\
&\vdots \\
\leq& Q^{\{\mu_k,\pi_{k+1}\}}(s_t,a_t)
\end{aligned}
$$

Here, the inequality is obtained by repeatedly expanding $Q^{\{\mu_k,\pi_k\}}$ on the RHS through $Q^{\{\mu_k,\pi_k\}}(s,a) = r(s,a) + \gamma\mathbb{E}_{s'}\big\{\lambda \cdot \max_{\tilde{a}'\sim\mu_k} Q^{\{\mu_k,\pi_k\}}(s',\tilde{a}') + (1-\lambda)\cdot \mathbb{E}_{a'\sim\pi_k}[Q^{\{\mu_k,\pi_k\}}(s',a') - \omega(s',a'|\pi_k)]\big\}$ and applying the inequality $\mathbb{E}_{a\sim\pi_{k+1}}[Q^{\{\mu_k,\pi_k\}}(s,a) - \omega(s,a|\pi_{k+1})] \geq \mathbb{E}_{a\sim\pi_k}[Q^{\{\mu_k,\pi_k\}}(s,a) - \omega(s,a|\pi_k)]$. Finally, we arrive at convergence to $Q^{\{\mu_k,\pi_{k+1}\}}(s_t,a_t)$.

Then, we expand the historical policy sequence $\Pi_k = \{\pi_0,\pi_1,\cdots,\pi_{k-1}\}$ by adding the policy π_k, and obtain $\Pi_{k+1} = \{\pi_0,\pi_1,\cdots,\pi_k\}$. Next, we consider to prove $Q^{\{\mu_{k+1},\pi_{k+1}\}}(s,a) \geq Q^{\{\mu_k,\pi_{k+1}\}}(s,a), \forall(s,a) \in \mathcal{S}\times\mathcal{A}$. Recall that μ_{k+1} is the stationary policy mixture of Π_{k+1}, if the state-action visitation density $d^{\pi_i}(s,a) > 0, i = 0,\dots k$, then the corresponding mixture distribution $d^{\mu}(s,a) > 0$, hence the support region of μ_k is a subset of the support region of μ_{k+1}, *i.e.*, $\text{supp}(\mu_k) \in \text{supp}(\mu_{k+1})$. Since $\max_{a\sim\mu_i} Q(s,a) = \max_{a\in\text{supp}(\mu_i)} Q(s,a)$, then for any $Q : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$, the following inequality can be established:

$$
\max_{a\sim\mu_{k+1}} Q(s,a) \geq \max_{a\sim\mu_k} Q(s,a), \forall s \in \mathcal{S}
$$

Hence, we expand the $Q^{\{\pi,\mu\}}$ and utilize the above inequality repeatedly, then we obtain

$$
\begin{aligned}
Q^{\{\mu_k,\pi_{k+1}\}}(s,a) =& r(s,a) + \lambda\gamma\cdot\mathbb{E}_{s'}[\max_{a'\sim\mu_k(\cdot|s')} Q^{\{\mu_k,\pi_{k+1}\}}(s',a')] \\
&+ (1-\lambda)\gamma\mathbb{E}_{s'}\mathbb{E}_{a'\sim\pi_{k+1}}[Q^{\{\mu_k,\pi_{k+1}\}}(s',a')] \\
\leq& r(s,a) + \lambda\gamma\cdot\mathbb{E}_{s'}[\max_{a'\sim\mu_{k+1}(\cdot|s')} Q^{\{\mu_k,\pi_{k+1}\}}(s',a')] \\
&+ (1-\lambda)\gamma\mathbb{E}_{s'}\mathbb{E}_{a'\sim\pi_{k+1}}[Q^{\{\mu_k,\pi_{k+1}\}}(s',a')] \\
&\vdots \\
\leq& Q^{\{\mu_{k+1},\pi_{k+1}\}}(s,a)
\end{aligned}
$$

With the inequalities of these two stages, the policy improvement property is satisfied, $Q^{\{\mu_{k+1},\pi_{k+1}\}}(s,a) \geq Q^{\{\mu_k,\pi_k\}}(s,a), \forall(s,a) \in \mathcal{S}\times\mathcal{A}, |\mathcal{A}| < \infty$.

□

Proposition B.3 (**Policy iteration**). *Assume $|\mathcal{A}| < \infty$, by repeating iterations of the policy evaluation and policy improvement, any initial policies converge to the optimal policies $\{\mu^*,\pi^*\}$, s.t. $Q^{\{\mu^*,\pi^*\}}(s_t,a_t) \geq Q^{\{\mu',\pi'\}}(s_t,a_t), \forall\mu' \in \Pi, \pi' \in \Pi, \forall(s_t,a_t) \in \mathcal{S}\times\mathcal{A}$.*

Proof. Let Π be the space of policy distributions and let $\{\mu_i,\pi_i\}$ be the policies at iteration i. By the policy improvement property in Proposition 4.3, the sequence $Q^{\{\mu_i,\pi_i\}}$ is monotonically increasing. Also, for any state-action pair $(s_t,a_t) \in \mathcal{S}\times\mathcal{A}$, each Q^{μ_i,π_i} is bounded due to the discount factor γ. Thus, the sequence of $\{\mu_i,\pi_i\}$ converges to some $\{\mu^*,\pi^*\}$ that are local optimum. We will still need to show that $\{\mu^*,\pi^*\}$ are indeed optimal, we assume finite MDP, as typically assumed for convergence proof in usual policy iteration (Sutton, 1988). At convergence, we get $J_{\mu^*,\pi^*}(\mu^*,\pi^*)[s] \geq J_{\mu^*,\pi^*}(\mu',\pi')[s], \forall\pi' \in \Pi, \mu' \in \Pi$. Using the same iterative augument as in the proof of Proposition 4.3, we get $Q^{\{\mu^*,\pi^*\}}(s,a) \geq Q^{\{\mu',\pi'\}}(s,a)$ for all $(s,a) \in \mathcal{S}\times\mathcal{A}$. Hence, $\{\mu^*,\pi^*\}$ are optimal in Π.

□

C. Implementation Details and Extensive Design Choices

C.1. Primary implementation details on BAC

Instantiating BAC amounts to specifying two main components: the use of in-sample learning for calculating the Bellman Exploitation operator $\mathcal{T}_{exploit}$, and the application of entropy regularization in the Bellman Exploration operator $\mathcal{T}_{explore}$. Here we provide the details for our primary implementation. For a broader discussion of potential design choices and extensions refer to Section C.3.

In-sample learning for $\mathcal{T}_{exploit}$. We leverage a simple and efficient approach for policy extraction using expectile regression (Kostrikov et al., 2021) to learn the value function, where only a hyperparameter τ is introduced. Considering that some large Q-values potentially are a result of "lucky" samples, we introduce a state value function V which approximates a high expectile of $Q(s,a)$ on the replay buffer $\mathcal{D}$. In this way, we can better account for the potential variance in Q-values, reducing overestimation error risk and ensuring that our algorithm is not relying solely on "lucky" samples.

To be specific, we initialize a state value V network to capture the maximum of Q value. Given the replay buffer $\mathcal{D}$, we can update the V network by a high expectile τ of $Q(s,a)$,

$$V(s) \leftarrow \arg\min_V \mathbb{E}_{(s,a)\sim\mathcal{D}} \left[|\tau - \mathbb{1}(Q(s,a) - V(s) < 0)|(Q(s,a) - V(s))\right]^2$$

Given $\tau > 0.5$, this asymmetric loss function would downweight the contributions of $Q(s,a)$ when $Q(s,a) < V(s)$ while giving more weights to larger values. If $\tau \to 1$, we have $V(s) \to \max_{a\sim\mu_k} Q(s,a)$. Hence, the target value of $\mathcal{T}_{exploit}$ can be calculated by

$$\mathcal{T}_{exploit}Q(s,a) = r(s,a) + \gamma\mathbb{E}_{s'\sim\mathcal{D}}\left[V(s')\right].$$

Entropy regularization in $\mathcal{T}_{explore}$. Based on the follow-up actions a' derived from fresh policy π_θ, we compute $\mathcal{T}_{explore}Q(s,a)$, employing the entropy regularization $\alpha \log \pi(a_t|s_t)$ from SAC (Haarnoja et al., 2018a) as the $\omega(\cdot|\pi)$. To ease the computational burden of learning a separate V-function for $\mathcal{T}_{explore}$, we opt to directly compute the expectation of the Q-value. Thus, the target value of the Bellman Exploration operator $\mathcal{T}_{exploit}$ can be calculated as follows:

$$\mathcal{T}_{explore}Q(s,a) = r(s,a) + \gamma\mathbb{E}_{s'\sim\mathcal{D}}\big[\mathbb{E}_{a'\sim\pi}Q(s',a') - \alpha\log\pi(a'|s')\big]$$

Algorithm overview on BAC. The pseudocode of our proposed BAC is provided in Algorithm 2.

Algorithm 2 Primary Implementation of BEE Actor-Critic (BAC)

initialize: Q networks Q_ϕ, V network V_ψ, policy π_θ, replay buffer $\mathcal{D}$
Sample n tuples from random policy and add to $\mathcal{D}$
repeat
  for each gradient step **do**
    Sample a mini-batch of N transitions (s,a,r,s') from $\mathcal{D}$
    Update V_ψ by $\min_\psi \mathbb{E}_{s,a}\left[|\tau - \mathbb{1}(Q_\phi(s,a) < V_\psi(s))|(Q_\phi(s,a) - V_\psi(s))^2\right]$
  end for
  for each environment step **do**
    Collect data with π_θ from real environment; add to $\mathcal{D}$
  end for
  for each gradient step **do**
    Compute $\mathcal{T}_{exploit}Q_\phi(s,a) \leftarrow r + \gamma\mathbb{E}_{s'}[V_\psi(s')]$
    Compute $\mathcal{T}_{explore}Q_\phi(s,a) \leftarrow r + \gamma\mathbb{E}_{s'}\mathbb{E}_{a'\sim\pi_\theta}[Q_\phi(s',a') - \alpha\log\pi_\theta(a'|s')]$
    Calculate the target Q value: $\mathcal{B}Q_\phi \leftarrow \lambda\mathcal{T}_{exploit}Q_\phi + (1-\lambda)\mathcal{T}_{explore}Q_\phi$
    Update Q_ϕ by $\min_\phi \left(\mathcal{B}Q_\phi - Q_\phi\right)^2$
    Update π_θ by $\max_\theta Q_\phi(s,a)$
  end for
until the policy performs well in the environment

C.2. Primary implementation details on MB-BAC algorithm

Modeling and learning the dynamics models. We adopt the widely used model learning technique in our baseline methods (Janner et al., 2019; Lai et al., 2021; Ji et al., 2022; Zhang et al., 2024). To be specific, MB-BAC uses a bootstrap ensemble of dynamics models $\{\hat{f}_{\phi_1}, \hat{f}_{\phi_2}, \ldots, \hat{f}_{\phi_K}\}$. They are fitted on a shared replay buffer $\mathcal{D}_e$, with the data shuffled differently for each model in the ensemble. The objective is to optimize the Negative Log Likelihood (NLL),

$$\mathcal{L}^H(\phi) = \sum_t^H [\mu_\phi(s_t, a_t) - s_{t+1}]^T \Sigma_\phi^{-1}(s_t, a_t)[\mu_\phi(s_t, a_t) - s_{t+1}] + \log\det \Sigma_\phi(s_t, a_t).$$

The prediction for these ensemble models is, $\hat{s}_{t+1} = \frac{1}{K}\sum_{i=1}^K \hat{f}_{\phi_i}(s_t, a_t)$. More details on network settings are presented in Table 2.

Policy optimization and model rollouts. We employ BAC as the policy optimization oracle in MB-BAC. Using the truncated short model rollouts strategy (Janner et al., 2019; Lai et al., 2021; Pan et al., 2020; Ji et al., 2022), we generate model rollouts from the current fresh policy. In the policy evaluation step, we repeatedly apply the BEE operator to the Q-value. We compute the V-function for the Bellman Exploitation operator on the environment buffer $\mathcal{D}_e$, which contains real environment interactions collected by historical policies, and we compute the $\mathcal{T}_{explore}Q$ operation to the model buffer $\mathcal{D}_m$ generated by the current policy π.

Algorithm overview on MB-BAC. We give an overview of MB-BAC in Algorithm 3.

Algorithm 3 Primary Implementation of Model-based BAC (MB-BAC)

initialize: Q networks Q_ϕ, V network V_ψ, policy π_θ, ensemble models $\{\hat{f}_{\phi_1}, \hat{f}_{\phi_2}, \ldots, \hat{f}_{\phi_K}\}$, environment buffer $\mathcal{D}_e$ and model buffer $\mathcal{D}_m$;
repeat
  for each environment step **do**
    Collect data with π_θ from real environment; add to $\mathcal{D}_e$. // Interactions with real env
  end for
  for each gradient step **do**
    Train all models $\{\hat{f}_{\phi_1}, \hat{f}_{\phi_2}, \ldots, \hat{f}_{\phi_K}\}$ on $\mathcal{D}_e$. // Model learning
  end for
  for each model rollout step **do**
    Perform h-step model rollouts using policy π_θ; add to $\mathcal{D}_m$. // Model rollouts
  end for
  // Policy optimization
  Update V_ψ by $\min_\psi \mathbb{E}_{s,a\sim\mathcal{D}_e}|\tau - \mathbb{1}\left(Q_\phi(s,a) < V_\psi(s)\right)|\left(Q_\phi(s,a) - V_\psi(s)\right)^2$
  Compute $\mathcal{T}_{exploit}Q_\phi(s,a) = r(s,a) + \gamma\mathbb{E}_{s'\sim\mathcal{D}_m}[V_\phi(s')]$.
  Compute $\mathcal{T}_{explore}Q_\phi(s,a) = r(s,a) + \gamma\mathbb{E}_{s'\sim\mathcal{D}_m}\mathbb{E}_{a'\sim\pi}[Q(s',a') - \alpha\log\pi_\theta(a'|s')]$.
  Calculate the target Q value: $\mathcal{B}Q_\phi \leftarrow \lambda\mathcal{T}_{exploit}Q_\phi + (1-\lambda)\mathcal{T}_{explore}Q_\phi$
  for each gradient step **do**
    Update Q_ϕ by $\min_\phi \left(\mathcal{B}Q_\phi - Q_\phi\right)^2$
    Update π_θ by $\max_\theta Q_\phi(s,a)$. // Policy optimization
  end for
until the policy performs well in the environment

C.3. Possible design choices and extensions

C.3.1. More design choices on computing $\mathcal{T}_{exploit}Q$

Towards computing $\mathcal{T}_{exploit}Q$ based on the policy mixture μ, a direct solution might be using an autoencoder to model μ (Fujimoto et al., 2019; Lyu et al., 2022). Unfortunately, in the online setting, learning μ would be computationally expensive as it varies dynamically with policy iterations. In our main implementation, we use the expectile regression, an in-sample approach, for the computation of $\max Q$. Beyond this, here we introduce two other in-sample techniques that can be used to calculate $\max Q$.

Sparse Q-learning. Sparse Q-learning (Xu et al., 2023) considers an implicit value regularization framework by imposing a general behavior regularization term. When applied Neyman $\mathcal{X}^2$-divergence as the regularization term, the state value function can be trained by

$$V(s) \leftarrow \arg\min_V \mathbb{E}_{(s,a)\sim\mathcal{D}} \left[\mathbb{1}\left(1 + \frac{Q(s,a) - V(s)}{2\alpha} > 0\right) \left(1 + \frac{Q(s,a) - V(s)}{2\alpha}\right)^2 + \frac{V(s)}{2\alpha} \right].$$

Exponential Q-learning. Similar to sparse Q-learning, exponential Q-learning (Xu et al., 2023) utilizes Reverse KL divergence as the regularization term and the state value function $V(s)$ can be updated by

$$V(s) \leftarrow \arg\min_V \mathbb{E}_{(s,a)\sim\mathcal{D}} \left[\exp\left(\frac{Q(s,a) - V(s)}{\alpha}\right) + \frac{V(s)}{\alpha} \right].$$

Based on the state value function $V(s)$ learned by sparse Q-learning or exponential Q-learning, we can compute the $\mathcal{T}_{exploit}Q$ by,

$$\mathcal{T}_{exploit}Q(s,a) = r(s,a) + \gamma \mathbb{E}_{s'\sim\mathcal{D}} \left[V(s') \right].$$

C.3.2. More design choices on exploration term $\omega(\cdot|\pi)$

In our primary implementation, we adopt the widely-used entropy regularization proposed in SAC (Haarnoja et al., 2018a). Various exploration terms $\omega(s, a|\pi)$, which have been extensively explored in previous off-policy actor-critic methods (Haarnoja et al., 2018a; Fujimoto et al., 2018; Han & Sung, 2021), could be adopted in our algorithm.

Variant on target policy smoothing regularization. Here we conduct an ablation study upon adopting the target policy smoothing regularization introduced by TD3 (Fujimoto et al., 2018), we term the variant of our algorithm TD3-BEE. Compared to TD3, our method exhibits improvements, as demonstrated in two experiments on D'Kitty in the main paper, as well as the Walker2d-v2 experiment in Figure 15.

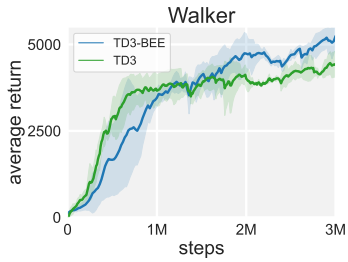


Figure 15. Ablation on the target policy smoothing regularization variant.

Other possible extensions. Various up-to-date advances in exploration term designs can be incorporated into our algorithm. For instance, pink noise (Eberhard et al., 2023) could be utilized to replace target policy smoothing regularization. Additionally, specially designed entropy terms, such as state or state-action occupancy entropy based on Shannon entropy (Hazan et al., 2019; Islam et al., 2019; Lee et al., 2019) or R'enyi entropy (Zhang et al., 2021a), could be considered. In certain "hard-exploration" scenarios (Aytar et al., 2018; Ecoffet et al., 2019; 2021), it may be beneficial to use specially tailored exploration terms, such as sample-aware entropy regularization (Han & Sung, 2021), particularly in sparse-reward or delayed-reward scenarios.

C.3.3. Extensions: automatic adaptive λ mechanisms

In our main experiments, we use a fixed λ value for simplicity. Although the value of λ does not fluctuate significantly in most of the environments we tested, specific scenarios may necessitate some tuning effort to find an appropriate λ.

To circumvent this λ search, we present three possible automatic adaptation methods for λ. The first two mechanisms involve using a binary value for λ, allowing the agent to freely switch between exploration and exploitation.

- $\min(\lambda)$. The insight here is to choose the smaller of the target update values induced by the Bellman Exploration operator and Bellman Exploitation operator, which might aid in alleviating the overestimation issue and enhance learning stability. The possible drawback is that it might prefer to exclusively choose the conservative Q-value. We formulate this mechanism as,

$$\lambda = \mathbb{1}\left(\mathcal{T}_{exploit}Q(s,a) - \mathcal{T}_{explore}Q(s,a) \leq 0\right).$$

  where $\mathbb{1}(x \leq 0)$ is an indicator function

$$\mathbb{1}(x \leq 0) = \begin{cases} 0 & x > 0, \\ 1 & x \leq 0. \end{cases}$$

- $\max(\lambda)$. This mechanism, conversely, selects the larger of the two values. This method might yield unstable results due to the influence of function approximation error. We formulate this mechanism as

$$\lambda = \mathbb{1}\left(\mathcal{T}_{exploit}Q(s,a) - \mathcal{T}_{explore}Q(s,a) \geq 0\right).$$

We also design a third mechanism for suggesting a continuous value of λ.

- $\mathrm{ada}(\lambda)$. Upon integrating new data into the replay buffer, the Bellman error variation would be small if the data is well exploited, and larger if not. Hence, when the Bellman error on the new-coming data is small, we may curtail reliance on executing $\mathcal{T}_{exploit}$ in the replay buffer and allocate more weight towards exploration. Motivated by this insight, we could adjust the value of λ according to the Bellman error. In practice, we divide the current Bellman error δ_k by the prior Bellman error δ_{k-1} to focus more on the Bellman error caused by the introduction of new-coming data. This way, λ can be automatically adapted during training as follows:

$$\lambda = \mathrm{clip}\left(\frac{\delta_k}{\delta_{k-1}}, 0, 1\right).$$

  Here, $\mathrm{clip}(\cdot, 0, 1)$ clips the λ by removing the value outside of the interval $[0, 1]$.

Remark 1: In Figure 16, we depict the learning curves of these three automatic λ adjustment mechanisms on Walker2d and Humanoid tasks, along with the eventual performance of SAC and the primary BAC. In these two settings, the ada(λ) mechanism generally yields competitive eventual performance, while min(λ) and max(λ) are more influenced by the environment settings. For instance, in the Humanoid task, we observed that the $\min(\lambda)$ mechanism almost entirely selects 0 after 1M iterations, thus could be considered as reducing to SAC in the later stages, and its final performance matches that of SAC; however, in the Walker2d environment, $\min(\lambda)$ results in a λ that switches more frequently.

Remark 2: Additionally, the third mechanism $\mathrm{ada}(\lambda)$ often yields promising results. Although it might introduce some oscillation, its advantage lies in providing guidelines for choosing λ, such as setting it to a fixed constant. As shown in Figure 17, the final fixed values of λ chosen for these three environments fall within the range of 0.4 to 0.5.

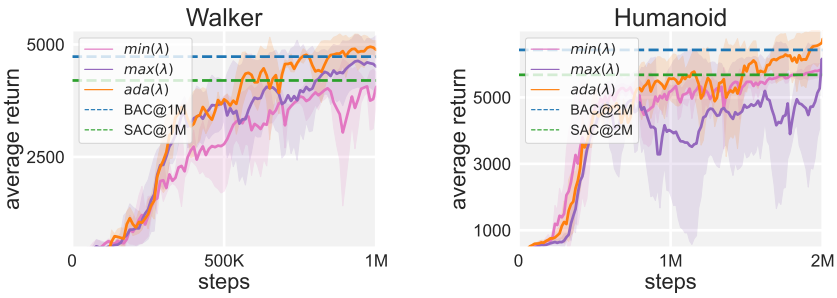


Figure 16. Learning curves with different lambda mechanisms in Walker2d and Humanoid tasks, where the dotted line indicates the eventual performance of BAC and SAC.

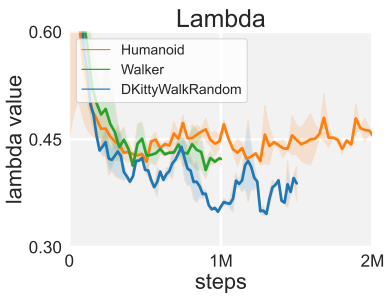


Figure 17. Curve of λ with $\mathrm{ada}(\lambda)$ mechanism in different environments.

C.4. Hyperparameter settings

A simple setting of hyperparameters λ and τ. The primary implementation of BAC introduces two extra hyperparameters, the balance factor λ and the quantile τ. A single set of hyperparameters ($\lambda = 0.5, \tau = 0.7$) is sufficient for our algorithm to achieve strong performance throughout all experiments across MuJoCo, DMControl, Meta-World, ROBEL, Panda-gym, Adroit, maniskill2, and Shadow Dexterous Hand benchmark suites.

Notably, the BEE operator is not bound to hyperparameter τ. As detailed in Appendix C.3, implementing BEE with other in-sample learning offline techniques, such as SQL and EQL instead of IQL would not have τ at all.

Heuristics for selecting λ and τ.

- λ: We initiated from $\lambda = 0.5$ as a balanced weight for $\mathcal{T}_{exploit}$ and $\mathcal{T}_{explore}$. Figure 16 depicts that moderate values around 0.5 obtain good performance. Besides, the automatic adaptive mechanisms we provided in Appendix C.3.3 may suffice and circumvent tuning.
- τ: Our choice to primarily use 0.7 comes from the IQL paper (Kostrikov et al., 2021) which uses 0.7 for MuJoCo tasks. And $\tau = 0.7$ already suffices for expected performance, thus we mostly use 0.7 in DMControl and Meta-World tasks.

Hyperparameters Tables. Hyperparameters for BAC are outlined in Table 1 and Table 2, respectively.

In MB-BAC, we follow the hyperparameters specified in MBPO (Janner et al., 2019). The symbol "$x \to y$ over epochs $a \to b$" denotes a linear function for establishing the rollout length. That is, at epoch t, $f(t) = \min(\max(x + \frac{t-a}{b-a} \cdot (y - x), x), y)$. And we set $\lambda = 0.5$ and $\tau = 0.7$ for each task.

Hyperparameter study. For practical use, $\lambda = 0.5$, $\tau = 0.7$ may suffice. If slightly tuned, we find that BAC can achieve even better performance. We provide the performance comparison of a wider set of hyperparameters in Figure 18. The results reveal that utilizing a higher $\tau = 0.9$ is not problematic and, in fact, enhances performance in ReacherHard in comparison to $\tau = 0.7$. Each BAC instance with varied hyperparameters surpasses the SAC in ReacherHard.

Remark 3: A high (e.g., 0.9) τ may not be problematic in the online setting. This differs from offline RL. In the offline setting, the peak distributions will occur in various quantiles for different datasets, thus an unsuitable τ may cause erroneous estimation. However, in online settings, ongoing interactions could enrich peak data. As policy improves, the replay buffer accumulates high-value data, thus reducing sensitivity to τ.

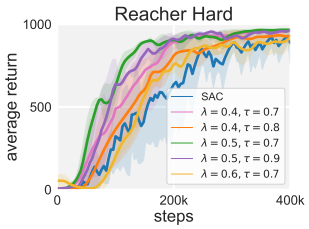


Figure 18. Learnings curves of BAC with a wider set of hyperparameters on ReacherHard.

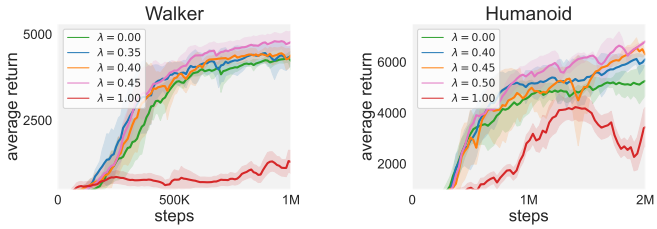


Figure 19. Parameter study on λ. The experiments are run over 4 random seeds.

Setting an appropriate weighted coefficient λ, BAC could balance the exploitation and exploration well. We may note that the algorithm is reduced to the online version of IQL (Kostrikov et al., 2021) for an extreme value $\lambda = 0$. According to Figure 19, and the detailed settings and hyperparameter studies in Appendix C.4, we find that a moderate choice of λ around 0.5 is sufficient to achieve the desired performance across all locomotion and manipulation tasks we have benchmarked. This underscores that BAC does not need heavy tuning for strong performance.

C.5. Computing infrastructure and computational time

Table 3 presents the computing infrastructure used for training our algorithm BAC and MB-BAC on benchmark tasks, along with the corresponding computational time. Compared to SAC, the total training time of BAC only increased by around 8% for Humanoid (1.19 H for 5M steps). Thus. we believe the additional costs are acceptable. Further, for practical use, BAC requires fewer interactions for similar performance, which may lower the needed computation time.

Table 1. Hyperparameter settings for BAC in benchmark tasks.

Hyper-parameter	Value
Q-value network	MLP with hidden size 512
V-value network	MLP with hidden size 512
policy network	Gaussian MLP with hidden size 512
discounted factor	0.99
soft update factor	0.005
learning rate	0.0003
batch size	512
policy updates per step	1
value updates per step	1
λ	0.5
τ	0.7

Table 2. Hyperparameter settings MB-BAC in benchmark tasks.

Hyper-parameter	Hopper	Walker2d	Ant	Humanoid
dynamical models network	Gaussian MLP with 4 hidden layers of size 200			
ensemble size	5			
model rollouts per policy update	400			
rollout schedule	$1 \rightarrow 15$ over epochs $20 \rightarrow 100$	1	$1 \rightarrow 25$ over epochs $20 \rightarrow 100$	$1 \rightarrow 25$ over epochs $20 \rightarrow 300$
policy network	Gaussian with hidden size 512		Gaussian with hidden size 1024	
policy updates per step	40	20	20	20

Table 3. Computing infrastructure and the computational time for MuJoCo benchmark tasks.

	Hopper	Walker	Swimmer	Ant	HumanoidStandup	Humanoid
CPU	AMD EPYC 7763 64-Core Processor (256 threads)					
GPU	NVIDIA GeForce RTX 3090 $\times$ 4					
BAC computation time in hours	-	-	6.56	13.47	11.65	16.57
SAC computation time in hours	-	-	6.08	12.88	11.18	15.38
MB-BAC computation time in hours	18.35	19.51	-	27.57	-	30.86

D. Environment Setup

We evaluate the BEE operator across over 50 diverse continuous control tasks, spanning **MuJoCo, ROBEL, DMControl, Meta-World, Adroit, ManiSkill2, Panda-gym, Shadow Dexterous Hand, MyoSuite** benchmark suites. We find our algorithm BAC excels in both locomotion and manipulation tasks.

Besides, we conduct experiments on 4 noisy environments and 6 sparse reward tasks to further showcase the effectiveness of the BEE operator.

As a versatile plugin, it seamlessly enhances performance with various policy optimization methods, shining in model-based and model-free paradigms.

We also validate BAC using a **cost-effective D'Kitty robot** to navigate various complex terrains and finally reach goal points and desired postures. The 4 real-world quadruped locomotion tasks highlight BAC's effectiveness in real-world scenarios.

★ Visualizations of these tasks are provided in Figure 20, 21, 22, 23, 24, 25, 26, 27 and 28.

D.1. Environment setup for evaluating BAC

MuJoCo benchmark tasks. We benchmark BAC on four continuous control tasks in OpenAI Gym (Brockman et al., 2016) with the MuJoCo (Todorov et al., 2012) physics simulator, including Swimmer, Ant, Humanoid, HumanoidStandup, using their standard versions.

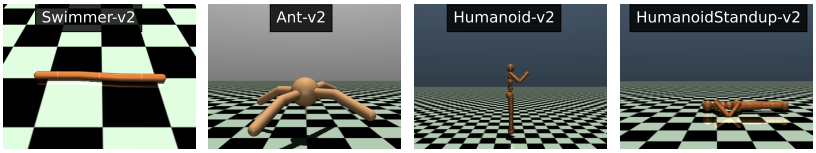


Figure 20. Visualization of **simulated tasks** from MuJoCo.

D'Kitty simulated tasks. ROBEL (Ahn et al., 2020) is an open-source platform of cost-effective robots designed for real-world reinforcement learning. The D'Kitty robot, with four legs, specializes in agile-legged locomotion tasks. ROBEL platform provides a set of continuous control benchmark tasks for the D'Kitty robot. Details for the base task (DKittyWalkRandomDynamics):

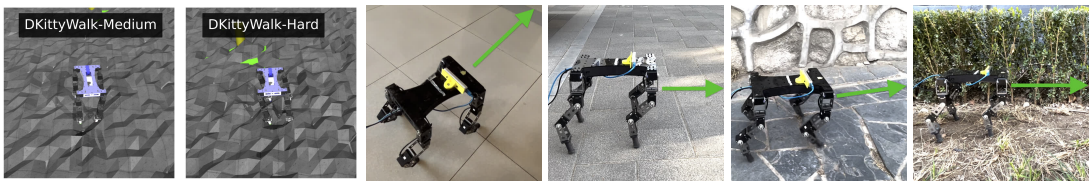
- Task: D'Kitty robot moves from an initial position $p_{t,kitty}$ to a desired one p_{goal} while maintaining balance and facing direction.
- Setting: Randomized joint parameters, damping, friction, and terrain with heights up to 0.05m.
- Reward function: The reward function contains five parts, the upright standing reward $r_{t,upright}$, the distance penalty $d_{t,goal} = \|p_{goal} - p_{t,kitty}\|_2$, the heading alignment $h_{t,goal} = R_{y,t,kitty}(p_{goal} - p_{t,kitty})/d_{t,goal}$, a small task success bonus r_{bonus_small} and a big task success bonus r_{bonus_big}. Thus, the reward function is defined as
$$r_t = r_{t,upright} - 4d_{t,goal} + 2h_{t,goal} + r_{bonus_small} + r_{bonus_big}.$$
- Success indicator: The success is defined by meeting distance and orientation criteria. The formulation of the success indicator is:
$$\phi_{se}(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\mathbb{1}\left(d^{(\tau)}_{T,goal} < 0.5\right) * \mathbb{1}\left(u^{(\tau)}_{T,kitty} > \cos(25^\circ)\right)\right].$$
The "Success Rate" in our experiments refers to the success percentage over 10 runs.

To construct more challenging locomotion tasks, we modify the base task DKittyWalkRandomDynamics by increasing terrain unevenness:

- *DKittyWalk-Hard*: the randomized height field is generated with heights up to 0.07m.
- *DKittyWalk-Medium*: the randomized height field is generated with heights up to 0.09m.

D'Kitty real-world tasks. Our real-world validation experiments are performed using a cost-effective D'Kitty robot. D'Kitty (Ahn et al., 2020) is a twelve-DOF quadruped robot capable of agile locomotion. It consists of four identical legs mounted on a square base. Its feet are 3D-printed parts with rubber ends.

The D'Kitty robot is required to traverse various complex terrains, contending with unpredictable environmental factors, and finally reach a target point. We evaluate BAC and baseline methods on four different terrains: *smooth road* (with a target point at 3m), *rough stone road* (target point at 1m), *uphill stone road* (target point at 1m), and *grassland* (target point at 1m).



(a) D'Kitty simulated tasks (b) D'Kitty real-world tasks

Figure 21. Visualization of **simulated tasks** and **real-world robot tasks** from D'Kitty.

DMControl tasks. The DeepMind Control Suite (DMControl) (Tunyasuvunakool et al., 2020), provides a set of continuous control tasks with standardized structures and interpretable rewards. We evaluate BAC BEE-TD3, SAC, and TD3 on 15 diverse benchmark tasks from DMControl, including challenging high-dimensional tasks like Humanoid Walk, Humanoid Run, DogWalk, and DogRun.

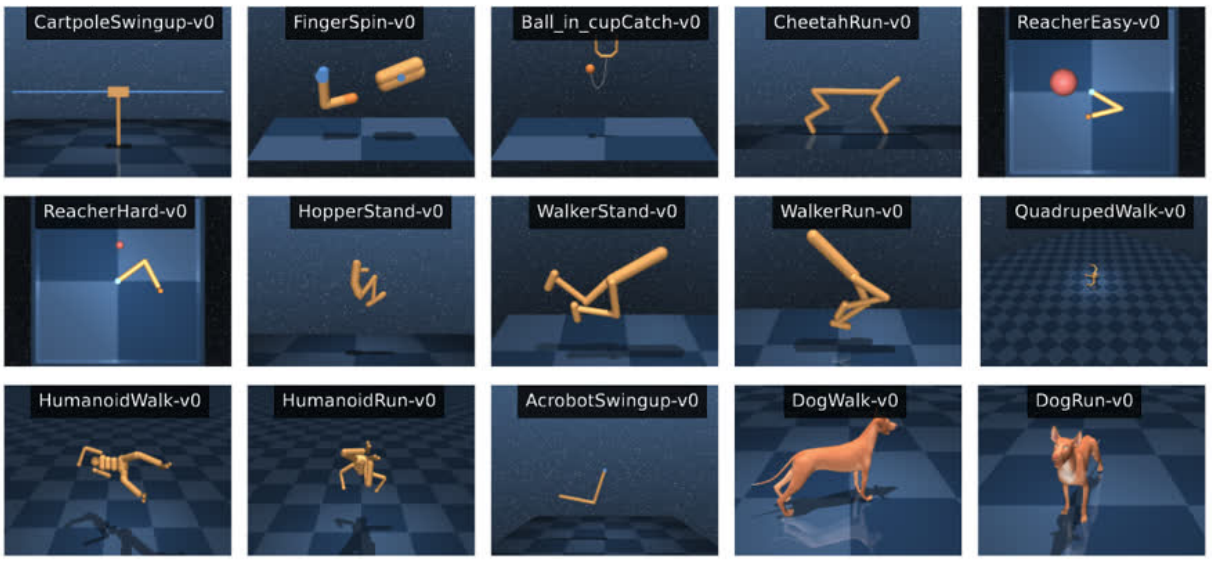


Figure 22. Visualization of **simulated tasks** from DMControl.

Meta-World tasks. Meta-World (Yu et al., 2019) provides a suite of simulated manipulation tasks with everyday objects, all of which are contained in a shared, tabletop environment with a simulated Sawyer arm. We evaluate BAC in 14 individual Meta-World tasks. Note that we conduct experiments on the goal-conditioned versions of the tasks from Meta-World-v2, which are considered harder than the single-goal variant often used in other works.

Figure 23. Visualization of **simulated tasks** from Meta-World.

Adroit tasks. Adroit (Rajeswaran et al., 2017) provides various dexterous manipulation tasks, consisting of a Shadow Dexterous Hand attached to a free arm. The system can have up to 30 actuated degrees of freedom.

Figure 24. Visualization of **simulated tasks** from Adroit.

ManiSkill2 tasks. ManiSkill2 (Gu et al., 2023) provides a suite of simulation tasks for learning generalizable manipulation skills and tackling long-horizon and complex daily chores.

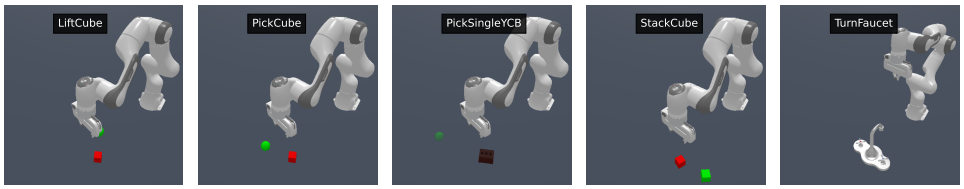


Figure 25. Visualization of **simulated tasks** from ManiSkill2.

Shadow Dexterous Hand tasks. The environments of Shadow Dexterous Hand (Plappert et al., 2018) are based on the Shadow Dexterous Hand, 5 which is an anthropomorphic robotic hand with 24 degrees of freedom. Of those 24 joints, 20 can be controlled independently, whereas the remaining ones are coupled joints. This sophisticated design mirrors human hand movements, allowing for complex and nuanced robotic tasks.

Panda-gym. Panda-gym provides (Gallouédec et al., 2021) a simulated environment of Franka Emika Panda robot for common tasks used to evaluate the RL algorithms.

Figure 26. Visualization of **simulated tasks** from Shadow Dexterous Hand.

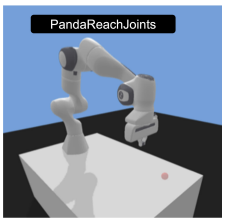


Figure 27. Visualization of **simulated task** from Panda-gym.

MyoSuite MyoSuite (Vittorio et al., 2022) is a set of challenging environments and tasks that help test how well reinforcement learning algorithms work in controlling muscles and bones in a realistic way. It includes detailed models of the elbow, wrist, and hand that can interact physically, making it possible to learn tasks that require complex and skillful movements. These tasks vary from simple body postures to more advanced actions like turning a key, spinning a pen, or rolling two balls in one hand.

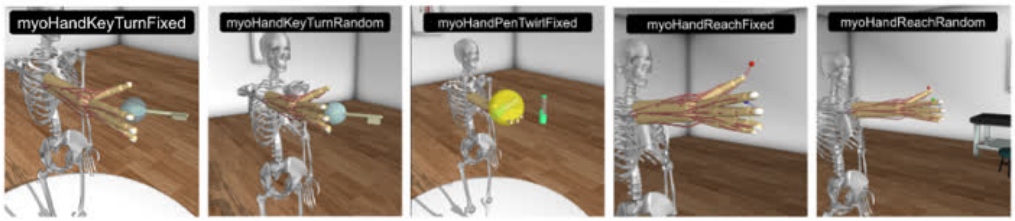


Figure 28. Visualization of **simulated tasks** from MyoSuite.

D.2. Environment setup for evaluating MB-BAC

We evaluate MB-BAC and its counterparts on four continuous control tasks in MuJoCo (Todorov et al., 2012). To ensure a fair comparison, we follow the same settings as our model-based baselines (MBPO (Janner et al., 2019), AutoMBPO (Lai et al., 2021), CMLO (Ji et al., 2022)), in which observations are truncated. The details of the experimental environments are provided in Table 4.

Table 4. Overview on environment settings for MB-BAC and model-based baselines. Here, θ_t denotes the joint angle and z_t denotes the height at time t.

	State Space Dimension	Action Space Dimension	Horizon	Terminal Function
Hopper-v2	11	3	1000	$z_t \leq 0.7$ or $\theta_t \geq 0.2$
Walker2d-v2	17	6	1000	$z_t \geq 2.0$ or $z_t \leq 0.8$ or $\theta_t \leq -1.0$ or $\theta_t \geq 1.0$
Ant-v2	27	8	1000	$z_t < 0.2$ or $z_t > 1.0$
Humanoid-v2	45	17	1000	$z_t < 1.0$ or $z_t > 2.0$

E. Baselines Implementation

Model-free RL algorithms. We compare with five popular model-free baselines, Soft Actor-Critic (SAC) (Haarnoja et al., 2018a), Diversity Actor-Critic (DAC) (Han & Sung, 2021), Random Reward Shift (RRS) (Sun et al., 2022), Twin Delayed DDPG (TD3) (Fujimoto et al., 2018), Proximal Policy Optimization (PPO) (Schulman et al., 2017). For RRS, we use the RRS-7 0.5 version as it provides better performance across diverse environments compared to other alternatives (RRS-3 0.5, RRS-3 1.0, RRS-7 1.0). For MuJoCo tasks, the hyperparameters of DAC, RRS, TD3, and PPO are kept the same as the authors' implementations. We list the hyperparameters of TD3 in Table 6. Note that we mostly follow the implementation of the original paper but improve upon certain hyperparameter choices for DMControl and Meta-World tasks.

The implementation of SAC is based on the open-source repo (pranz24 (2018), MIT License). And we use automating entropy adjustment (Haarnoja et al., 2018b) for automatic α tuning. On MuJoCo benchmarks, we retain other parameters as used by the authors (Haarnoja et al., 2018a). On DMControl benchmarks, we followed the SAC hyperparameters suggested by TD-MPC paper (Hansen et al., 2022). On Meta-World benchmarks, we followed the SAC hyperparameters suggested by Meta-World paper (Yu et al., 2019). We list the hyperparameters of SAC in Table 5.

Table 5. Hyperparameter settings for SAC in MuJoCo and DMControl, Meta-World benchmark tasks.

	MuJoCo	DMControl	Meta-World
optimizer for Q	Adam(β_1=0.9, β_2=0.999)		
optimizer for α	Adam(β_1=0.5, β_2=0.999)		
learning rate	3×10^{-4}	1×10^{-4} (otherwise) 3×10^{-4} (Dog)	3×10^{-4}
discount (γ)	0.99		
number of hidden units per layer	256	1024	256
number of samples per minibatch	256	512 (otherwise) 2048 (Dog)	500
target smoothing coefficient (τ)	0.005		
target update interval	1	2	1
gradient steps	1		

Table 6. Hyperparameter settings for TD3 in MuJoCo and DMControl, Meta-World benchmark tasks.

	MuJoCo	DMControl	Meta-World
optimizer for Q	Adam(β_1=0.9, β_2=0.999)		
exploration noise	$\mathcal{N}(0, 0.1)$		
learning rate	3×10^{-4}	1×10^{-4} (otherwise) 3×10^{-4} (Dog)	3×10^{-4}
discount (γ)	0.99		
hidden layers	(400, 300)	(512, 512)	(512, 512)
number of samples per minibatch	100	256 (otherwise) 512 (Dog)	256
target smoothing coefficient (τ)	0.005		
target update interval	1		

Model-based RL algorithms. As for model-based methods, we compare with four state-of-the-art model-based algorithms, MBPO (Janner et al., 2019), SLBO (Luo et al., 2018), CMLO (Ji et al., 2022), AutoMBPO (Lai et al., 2021). The implementation of SLBO is taken from an open-source MBRL benchmark (Wang et al., 2019), while MBPO is implemented based on the `MBRL-LIB` toolbox (Pineda et al., 2021). To facilitate a fair comparison, MB-BAC and MBPO are run with identical network architectures and training configurations as specified by `MBRL-LIB`.

F. Investigations on the Underestimation Issue

The underestimation issue matters. While prior works focus more on reducing overestimation, our work shows that mitigating underestimation itself may improve both performance and sample efficiency. Let's delve deeper.

F.1. Why underestimation and under-exploitation matters?

Underestimation in the under-exploitation stage would negatively impact Q-value estimation. Underestimating Q-values of (s, a) due to suboptimal current policy successors, ignoring high-value replay buffer successors, hampers reselection of (s, a) . Two issues might arise,

- **Reduce sample efficiency**: The agent would require more samples to re-encounter such (s, a).
- **Hinder policy learning**: Misleading Q may trap the policy in ineffective exploration. The issue is exacerbated in failure-prone scenarios where high-value tuples are serendipities and policy performance oscillates.

F.2. What may cause the underestimation issue?

Underestimation bias may result in inefficient exploration, diminished sample efficiency, and, ultimately, reduced policy learning. The underestimation bias has been associated with the use of double-Q-technique, as highlighted in previous works (Fujimoto et al., 2018; Moskovitz et al., 2021). However, is the underestimation issue a sore outcome of the double-Q technique?

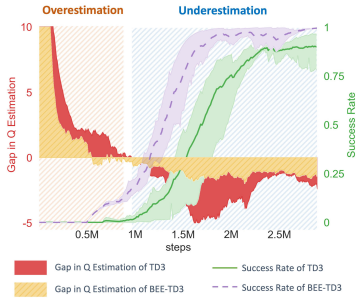


Figure 29. TD3 is also prone to underestimation pitfalls in the latter stage of training.

Underestimation issue in various off-policy actor-critic algorithms. AC algorithms are susceptible to the underestimation issue, as shown in Figure 4. To further illustrate this issue, we quantify the Q-value estimation gap of TD3 and BEE-TD3 in the DKittyWalkRandomDynamics task. The gap in Q estimation is evaluated by comparing the TD3/BEE-TD3's Q-values and the Monte-Carlo Q estimates using the trajectories in the replay buffer.

As shown in Figure 29, TD3 also experiences underestimation pitfalls during the later stages of training. Notably, we observe that the BEE operator helps to mitigate this underestimation issue and finally benefits performance.

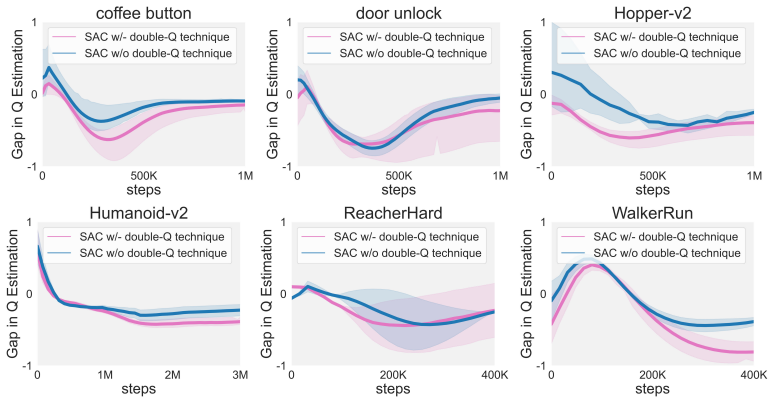
Double-Q technique is not the only culprit. We identify that the underestimation problem also occurs in many off-policy Actor-Critic (AC) algorithms independent of this technique. In this section, we investigate the causes of underestimation in the AC framework, irrespective of the double-Q technique's application. We also empirically show that various off-policy AC algorithms, with or without the double-Q trick, are prone to underestimation issues in many tasks.

In Figure 30, we plot Q-value estimation of SAC and TD3, along with their variants by eliminating the double-Q technique. We observe that both SAC and TD3 would face the underestimation issue in various robotic tasks.

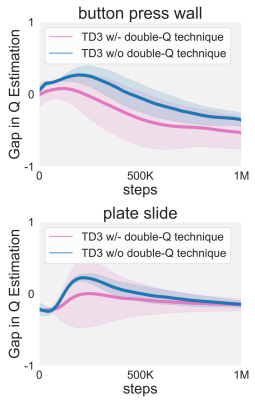
The optimization procedure of the AC framework can also contribute to underestimation. Ideally, the Bellman update needs to solve $Q(s, a) \leftarrow r(s, a) + \gamma \mathbb{E}_s[\max_a Q(s, a)]$. However, as $\max_a Q(s, a)$ operations are often impractical to calculate, so in the AC framework, we typically iteratively evaluate target Q-value as $\mathbb{E}_\pi[Q(s, a)]$, while implicitly conducting the max-Q operation in a separate policy improvement step to learn policy π.

Note that the ideal $\pi = \arg\max_{a\sim\pi} Q(s, a)$ is not possible to achieve practically within only a few policy gradient updates. Hence, the actual target value used in AC Bellman update $\mathbb{E}_{s,a\sim\pi} Q(s, a)$ can have a high chance to be smaller than $\mathbb{E}_s[\max_a Q(s, a)]$, causing underestimation. In other words, the non-optimal current policy in the AC framework can also contribute to underestimation.

In summary, Online RL training has a relatively high variance during training, thus the agent could not monotonically increasing its performance, then the best actions from mixture of the policies may always inferior to those from the current policy. Even if having a monotonically improvement agent, the phenomena may also exist. To give more insight, the expectation $\mathbb{E}_s[\max_{a\sim\mu_k} Q^{\mu_k}(s, a)]$ could be interpreted as the Q-value of the optimal policy derived from the replay buffer, not merely any arbitrary mixture policy. This concept draws from offline RL, which shows that it is possible to extract a



(a) SAC with or without double-Q-trick faces underestimation in the latter stage of training.



(b) TD3 with or without double-Q-trick faces underestimation in the latter stage of training.

Figure 30. The gap is assessed by subtracting the Q-values generated by the current policy from Monte-Carlo Q estimates derived from trajectories sampled using the current policy. A value less than 0 indicates underestimation. Off-policy Actor-Critic algorithms (SAC, TD3) with or without double-Q-trick are prone to underestimation pitfalls.

policy from the replay buffer $\mathcal{D}_k$ that is more optimal than all preceding data-collecting policies $\pi_0, \ldots, \pi_{k-1}$, and thus could potentially be superior to π_k

F.3. How to mitigate underestimation issue?

If the policy is the true optimizer under the current Q-value, it would mitigate the value underestimation issue. Yet it is practically infeasible within limited policy gradient updates. Therefore, the immediate solution is either to improve the policy slightly under the current Q-value or to find a better way to estimate the Q-value. We experimented with increasing the number of policy update steps but found that this approach did not yield satisfactory results.

Updating the policy with more update steps could not fully solve the issue. We conducted an experiment that updated the policy with more steps, as shown in Figure 31. While a moderate increase could enhance the performance of both SAC and BAC, excessively amplifying the number of updates might impair performance, primarily due to a diminished neural network's capacity to learn and generalize, as discussed in D'Oro et al. (2022). Moreover, updating the policy with a bit more steps might mitigate the value underestimation, yet it would lead to a prolonged training style, potentially leading to a significantly higher computational load.

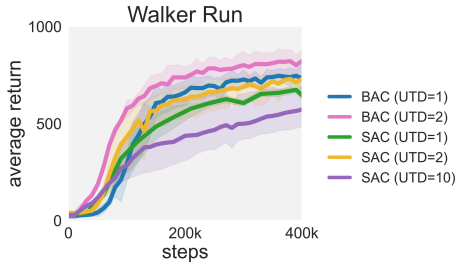


Figure 31. **Update the policy with more steps.** While a moderate increase could enhance the performance of both SAC and BAC, excessively amplifying the number of updates might impair performance, primarily due to a diminished neural network capacity to learn and generalize

Leveraging the more optimal actions in the replay buffer may help much. Many Actor-Critic algorithms commonly encounter the circumstance: the actions sampled from the current policy π fall short of the optimal ones stored in the replay buffer $\mathcal{D}$. The existence of more optimal actions in the replay buffer than generated by the current policy further supports the actual gap between the current policy and the ideal optimal one.

We identify that underestimation particularly occurs in the latter training stage, where we see a notable shortfall in the exploitation of the more optimal actions in the replay buffer, that is why we term it as ***under-exploitation***. Thus, exploiting the more optimal actions in the replay buffer to bootstrap Q would shorten the gap to the optima, hence mitigating underestimation.

F.4. The existence of "under-exploitation" stage

In our main paper, we propose $\Delta(\mu, \pi)$ to quantify the existence of the more optimal actions in the replay buffer than those generated by the current policy. Here, we will provide more empirical evidence to show its existence.

Existence in many scenarios. Here we provide more results on the existence of under-exploitation stage, as shown in Figure 32, that in various scenarios, positive $\Delta(\mu_k, \pi_k)$ occupies a significantly larger portion than negative $\Delta(\mu_k, \pi_k)$, indicating that the common Bellman Exploration operator $\mathcal{T}_{explore}$ suffers from under-exploitation stages for a prolonged period of time. Our findings indicate that underutilization of the replay buffer is a common occurrence. This sheds light on the potential for significant improvements if the buffer is fully leveraged.

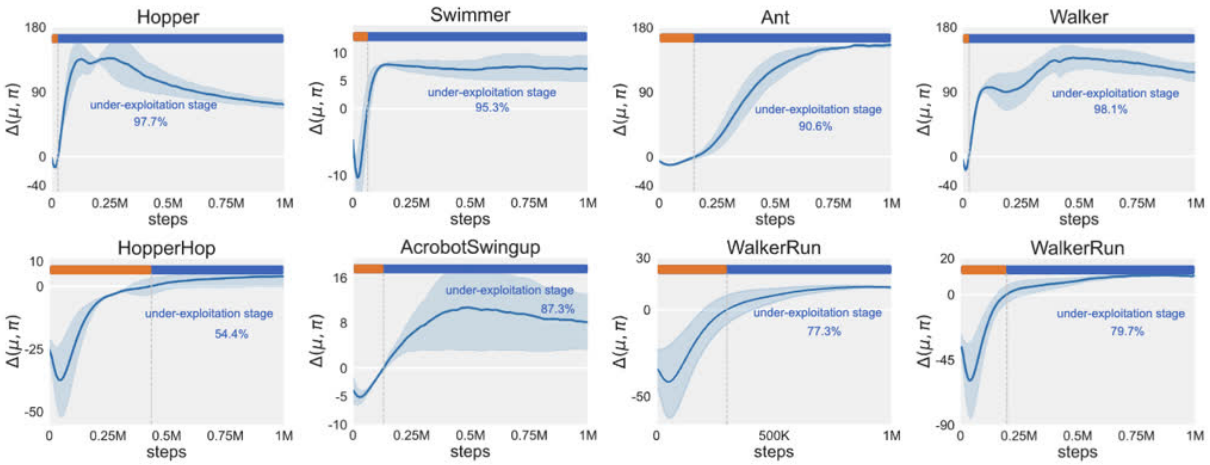


Figure 32. Visualization on under-exploitation stage on eight environments across MuJoCo and DMControl benchmark tasks, as analyzed through the lens of the SAC agent.

Existence without exploration bias. The existence of better actions in the replay buffer stems not solely from the entropy term. It also attributes to the particulars of the optimization in AC, as obtaining an optimal policy w.r.t the current Q value is practically unattainable with a few policy gradient updates. Many off-policy AC methods, relying solely on current policy for Q-value updates, may face under-exploitation issues.

Figure 33 illustrates the under-exploitation that occurs in SAC with varying α. Notably, under-exploitation is observed even in SAC instances with $\alpha = 0$, indicating the presence of under-exploitation even when there is no exploration bias. BAC mitigates under-exploitation pitfalls more, even equipped with an exploration term, when compared to the SAC instance with $\alpha = 0$.

Further, we conduct experiments by applying the BEE operator to TD3 with the exploration noise setting to zero, as shown in Figure 35. Actually, setting exploration noise to zero would degrade performance as the algorithms would fall short in exploration. However, integrating BEE with TD3 will boost the backbone algorithm performance regardless of the exploration noise.

Existence in various off-policy algorithms. Under-exploitation exists in many off-policy algorithms, not limited to SAC. Figure 34 shows that TD3 also encounters under-exploitation stages during training.

F.5. Explanations on the existence of under-exploitation circumstance

Positive $\Delta(\mu, \pi)$ during later training stages. From the visualization figures above, we often observe a positive $\Delta(\mu, \pi)$ during later training stages, indicating that the initial under-exploration stage is often followed by a subsequent under-exploitation stage. To give more insights,

- In the early training stages, the policy π performs poorly and possibly more randomly, resulting in 1) low-reward

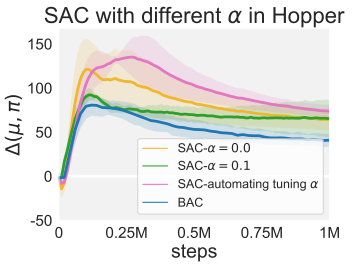


Figure 33. Under-exploitation occurs in SAC with different α.

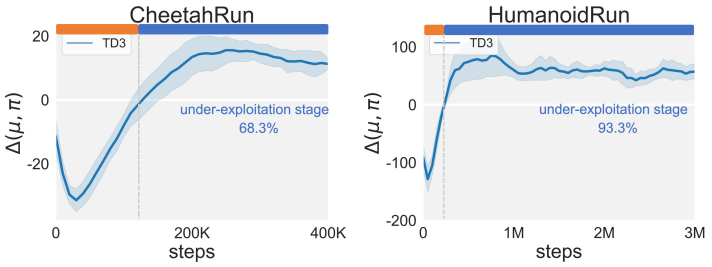


Figure 34. Visualization of $\Delta(\mu, \pi)$ on TD3 agent. Positive $\Delta(\mu, \pi)$ indicates the under-exploitation stage.

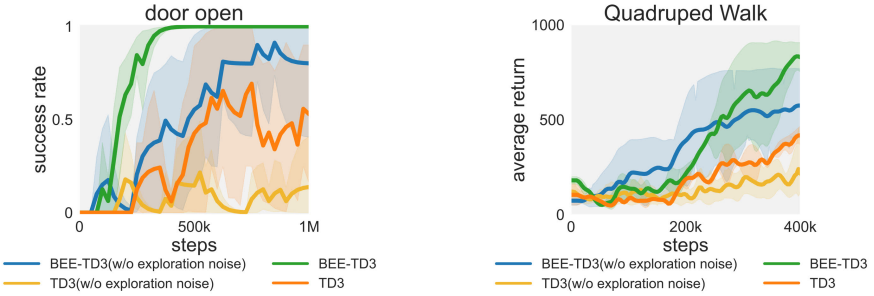


Figure 35. **Deterministic policy with/without exploration noise.** Setting exploration noise to zero would degrade performance as the algorithms would fall short in exploration. However, integrating the BEE operator with TD3 consistently improves performance by mitigating underestimation and enhancing replay buffer exploitation, regardless of exploration noise levels.

samples in the replay buffer with corresponding low Q values; 2) the exploration bonus improves the expected Q-value of the current policy.

- As training progresses, and the agent begins to solve the task, better actions than those generated by π may appear in the replay buffer. It is partially attributed to the iterative update nature of the Actor-Critic (AC) framework as discussed in Appendix F.5, which may lead to the existence of inferior actions after policy updates compared to the optimal ones in the replay buffer.

Possible causes for under-exploitation circumstance. Several factors contribute to this circumstance:

- **Exploration bias**: Exploration bias often leads to the overestimation of Q-values, promoting policy exploration of suboptimal actions.
- **AC framework nature**: Consideration of the iterative update nature of the Actor-Critic (AC) framework also brings two additional dimensions into play:

  Q-value estimation bias: During the training process, either underestimation or overestimation is inevitable. In other words, the true Q-value of the sampled actions from the current policy might be lower than some actions in the replay buffer.

  Suboptimal policy update: Ideally, each new policy should be the maximizer of the current Q to ensure policy improvement. However, obtaining such an optimal policy w.r.t the current Q function is practically unattainable with a few policy gradient updates.

G. Superior Q-value Estimation using BEE Operator

While being intuitively reasonable, BEE's potential benefits require further verification. In the following, we show that the BEE operator would facilitate the estimation of Q and thus improve sample efficiency compared to the commonly used Bellman evaluation operator.

BEE mitigates the under-exploitation pitfalls. The prevalent positive $\Delta(\mu, \pi)$ exposes the limitations of the Bellman Exploration operator $\mathcal{T}_{explore}$. The BEE operator alleviates the over-reliance on the current policy and mitigates the "under-exploitation" pitfalls by allowing the value of optimal actions in the replay buffer to be fully utilized in the Q-value update. To be more specific, when the $\mathcal{T}_{explore}$ operator is stuck in underestimation, the BEE operator would output a higher Q-value, as shown by the inequality $Q_{\mathcal{B}}^{\{\mu_k,\pi_k\}}(s,a) \geq Q_{\mathcal{T}_{explore}}^{\pi_k}(s,a) + \lambda\gamma\Delta(\mu_k,\pi_k)$. This agrees with the findings in Figure 1, the BEE operator exhibits lower underestimation bias and faster convergence of success rate, indicating its better sample efficiency.

To further illustrate this, we visualize the $\Delta(\mu, \pi)$ metric for both SAC and BAC agents in Hopper and Swimmer environments. As shown in Figure 36, BAC improves the metric $\Delta(\mu, \pi)$ more towards 0, indicating its ability to learn more accurate Q-values. BEE operator prevents a suboptimal current policy from diminishing the value of these actions. Thus, BAC has a higher likelihood of re-encountering these high-valued actions used for computing target Q-value, effectively mitigating under-exploitation pitfalls.

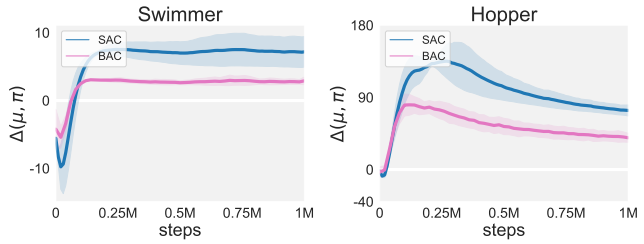


Figure 36. Visualization of $\Delta(\mu, \pi)$ with SAC and BAC agent in Hopper and Swimmer tasks. Here, BAC improves the metric more towards 0 compared to SAC.

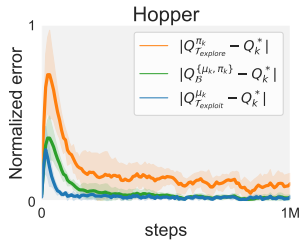


Figure 37. Q-value estimation error of different operators.

BEE exhibits no extra overestimation. While the BEE operator seeks to alleviate underestimation, it does not incite additional overestimation. This is in contrast to prior techniques that excessively increase exploration bonuses or use optimistic estimation (Brafman & Tennenholtz, 2002; Kim et al., 2019; Pathak et al., 2019), which may distort the Q-value estimates and potentially cause severe overestimation (Ciosek et al., 2019). The Bellman Exploitation operator, $\mathcal{T}_{exploit}$ does not introduce artificial bonus items and instead relies solely on the policy mixture induced by the replay buffer to calculate the maximum Q-value. Consequently, $\mathcal{T}_{exploit}$ is grounded in real experiences.

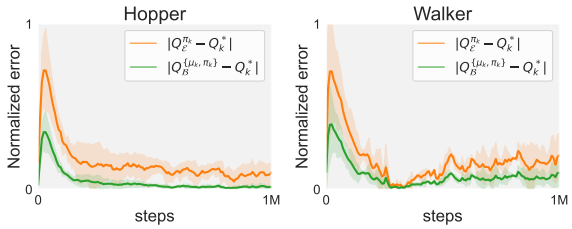


Figure 38. Q-value estimation error comparison. $\mathcal{T}_{explore}$ is referred to as $\mathcal{E}$ for brevity. And Q_k^* is obtained practically with Monte-Carlo estimation.

As illustrated in Figure 38, the Q-value function induced by the BEE operator enjoys a lower level of overestimation and underestimation. Further, as empirically shown in Figure 1 and 5, with enhanced exploitation, the BEE operator enables faster and more accurate Q-value learning, thereby reducing the chains of ineffective exploration on some inferior samples, and leading to improved sample efficiency. Moreover, we consider an extreme situation, $\lambda = 1$. We plot Q-estimation-error under $\lambda = 1$ in Figure 37, and find that it does not cause overestimation.

Actually, $\mathcal{T}_{exploit}^{\mu}$, the reduced form BEE operator when $\lambda = 1$, relies on real experience and may lead to conservative estimation. To give more insights, online learning's dynamic replay memory could be treated as a static dataset at a specific time step. Then, in practice, the Bellman exploitation operator $\mathcal{T}_{exploit}^{\mu}$ could be obtained by several effective techniques from offline RL. The pessimistic treatments in offline RL penalize overestimation heavily. Thus a pure exploitation operator practically even might help to reduce overestimation.

H. Effectiveness in Failure-prone Scenarios

In our main paper, we have shown the effectiveness of the BEE operator in terms of **ability to seize serendipity** and **more stable Q-value in practice**. Here, we investigate the superiority of the BEE operator in terms of **ability to counteract failure**, **effectiveness in noisy environments**, and **effectiveness in sparse reward environments**.

H.1. The ability to counteract failure

The BEE operator can not only grasp success but also **counteract failure**. Here, we conduct some extreme experiments to show it. We simultaneously train SAC and BAC, and at 100k steps, both have reached a certain level of performance. This suggests that there already exists several high-value (successful) samples in the replay buffer. At this point, we abruptly apply a massive perturbation to the policy and value networks (*i.e.*, at 100k steps, we substitute the current policy with a random one and reinitialize the value networks). Keep other components the same, we continue the training. This setup is a magnification of a situation often seen in failure-prone scenarios: the agent is prone to performance drop, which consequently disrupts the Q value estimate and necessitates additional sampling for recovery, thus forming a stark gap in the learning curve.

As shown in Figure 39, we can observe that the degree of performance drop in BAC after 100k steps is significantly less than that in SAC, coupled with a faster recovery speed, demonstrating its better resilience against failure. This capability possibly stems from the fact that the learned Q value by the BEE operator is less influenced by the optimal level of the current policy.

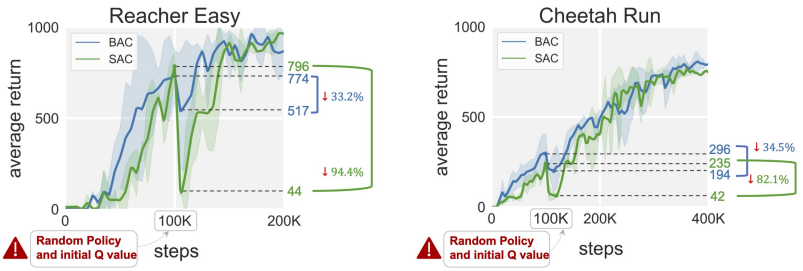
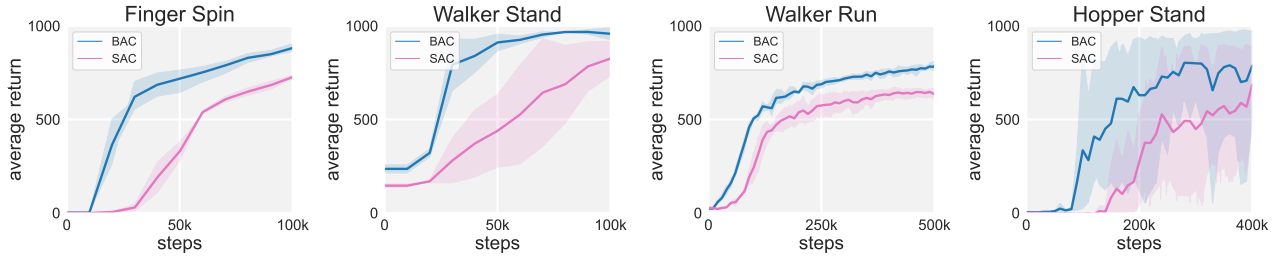


Figure 39. Comparison of the ability to counteract failure. BAC exhibits less performance drop (33.2% in ReachEasy and 34.5% in CheetahRun) and faster recovery.

H.2. Effectiveness in noisy environments

We conduct experiments in noisy environments to investigate the robustness of the BEE operator. Noisy environments are created by adding Gaussian noise to the agent's action at each step. Specifically, the environment executes $a' = a + \mathrm{WN}(\sigma^2)$ as the action, where a is the agent's original action, and $\mathrm{WN}(\sigma^2)$ is an unobserved Gaussian white noise with a standard deviation of σ.

Despite the noisy settings that can destabilize Q values and impede sample efficiency, as shown in Figure 40, BAC demonstrates desirable performance, even outperforming SAC more significantly than in noise-free environments.



(a) $\sigma = 0.1$ (b) $\sigma = 0.2$

Figure 40. Results in noisy environments: (a) in noisy FingerSpin and WalkerStand tasks with $\sigma = 0.1$; (b) in noisy WalkerRun and HopperStand tasks with a server noise $\sigma = 0.2$.

H.3. Illustrative example on the failure-prone scenario.

We provide a typical failure-prone scenario to illustrate the effectiveness of our operator.

Task description. As shown in Figure 41, a small particle spawns in a 2D continuous space of $[0, 10] \times [0, 10]$. The particle could take any random moves inside the space with a length of 0.1. The objective is to let the particle hit the small hole of radius 0.1 at $(10, 5)$. In other words, the particle receives a non-zero reward if and only if it is in the hole. Starting from a random policy, the particle has to explore the space and find the hole.

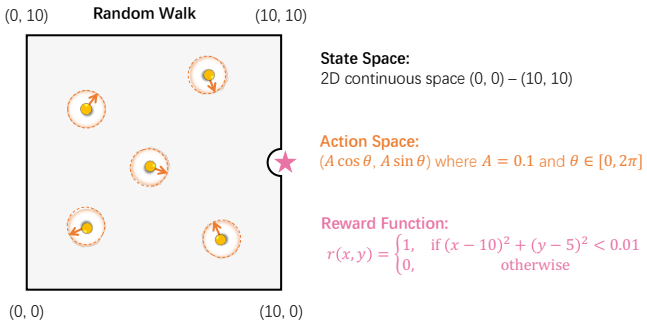


Figure 41. We construct a failure-prone scenario: Random Walk. The yellow particle has to explore the 2D space, and the target is to reach the small hole around $(10, 5)$ (pink star).

Q-value comparison. Only 10 of 100000 samples have reached the hole in the replay buffer. Figure 42 shows the Q-value heatmaps with the standard Bellman operator and our proposed BEE operator after 100, 200, and 500 Q-learning iterations. Q-values learned by the BEE operator are much closer to the expected ones in limited iterations.

Let's dive deeper. Given $\bar{s}$ is one of the successor of a tuple (s, a), the target update $r + \gamma \mathbb{E}_{a' \sim \pi} Q(s', a')$ for Standard Bellman Operator, only focuses on actions a' from the current policy π, ignoring a more optimal one $\bar{a}'$. Thus $Q(s, a)$ which should be valued higher is underestimated. Then next policy derived from the current misleading Q may prefer not to sample (s, a) again as it does not have a high value. Thus, algorithms based on the standard Bellman operator might take a substantially longer time to re-encounter these serendipities for training with decreased sample efficiency. In contrast, the BEE operator extracts the best actions in the replay buffer to construct referenced value to the Q-value function thus mitigates such underestimation issue.

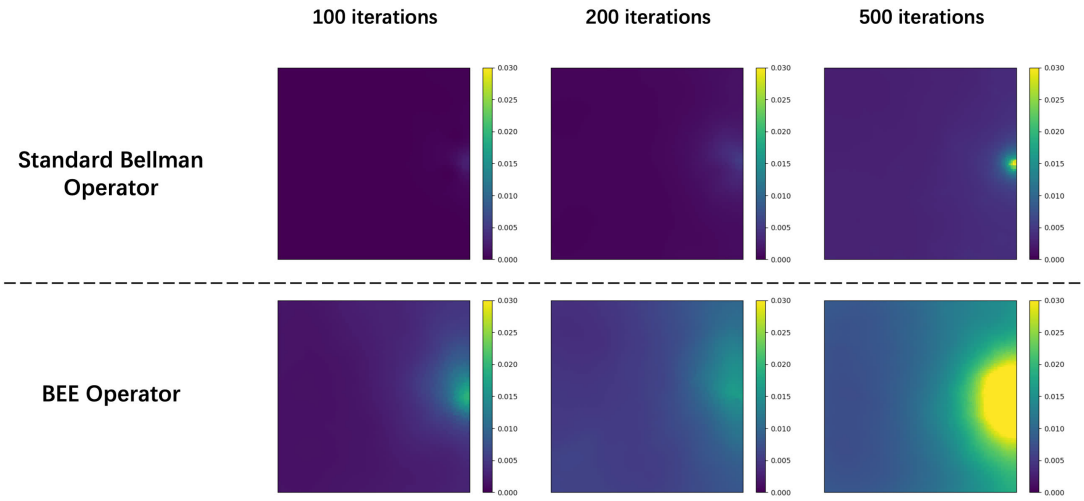


Figure 42. Q-value heatmaps with standard Bellman operator and the BEE operator after 100, 200, and 500 iterations.

H.4. Effectiveness in sparse-reward tasks

We conduct experiments in sparse reward tasks to further demonstrate the generalizability of the BEE operator. We evaluate in both robot locomotion and manipulation tasks, based on the sparse reward version of benchmark tasks from **Meta-World** (Yu et al., 2019), **panda-gym** (Gallouédec et al., 2021), **ROBEL** (Ahn et al., 2020). Here is the task description:

Meta-World manipulation tasks are based on a Sawyer robot with end-effector displacement control.

- coffee button: Push a button on the coffee machine whose position is randomized.
- hand insert: Insert the gripper into a hole.
- door open: Open a door with a revolving joint. Randomize door positions.

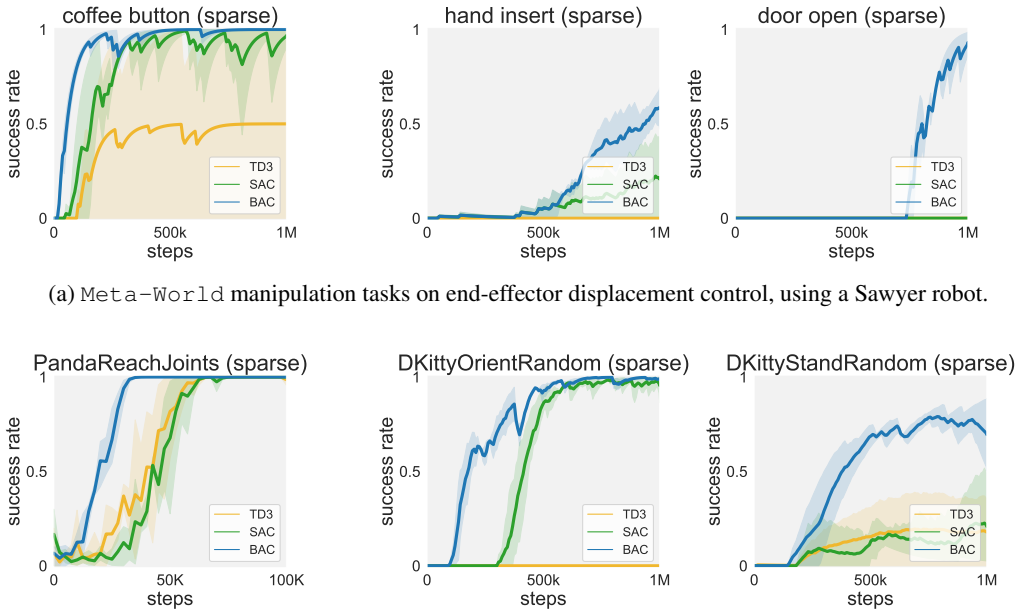
Panda-gym manipulation tasks are based on a Franka Emika Panda robot with joint angle control.

- PandaReachJoints: A target position must be reached with the gripper. This target position is randomly generated in a volume of $30cm \times 30cm \times 30cm$.

ROBEL quadruped locomotion tasks are based on a D'Kitty robot with 12 joint positions control.

- DKittyStandRandom: The D'Kitty robot needs to reach a pose while being upright from a random initial configuration. A successful strategy requires maintaining the stability of the torso via the ground reaction forces.
- DKittyOrientRandom: The D'Kitty robot needs to change its orientation from an initial facing direction to a random target orientation. A successful strategy requires maneuvering the torso via the ground reaction forces while maintaining balance.

As shown in Figure 43, our BAC surpasses the baselines by a large margin.



(a) `Meta-World` manipulation tasks on end-effector displacement control, using a Sawyer robot.

(b) `Panda-gym` manipulation tasks on joint angles control, using a Franka Emika Panda robot.

(c) `ROBEL` quadruped locomotion tasks, on 12 joint positions control, using a ROBEL D'Kitty robot.

Figure 43. **Sparse reward tasks.** BAC outperforms the baselines on six sparse reward tasks across various control types and robotic platforms, including manipulation and locomotion tasks.

H.5. Task visualizations in failure-prone scenarios

HumanoidStandup. HumanoidStandup, provided by MuJoCo (Todorov et al., 2012), is a challenging locomotion task. The environment begins with the humanoid lying on the ground, and the goal is to enable the humanoid to stand up and then keep it standing by applying torques on the various hinges. The agent takes a 17-element vector for actions.

In the HumanoidStandup task, BAC demonstrates a significantly superior performance than all other algorithms. With average returns reaching approximately 280,000 at 2.5 million steps and 36,000 at 5 million steps, BAC surpasses other algorithms whose asymptotic performance peaks at around 170,000, as illustrated in Figure 44.

Visualization in Figure 45 depicts that the BAC agent can quickly achieve a stable standing pose. In contrast, the SAC agent ends up in an unstable, swaying kneeling position, DAC ends up sitting on the ground, and the RRS agent, regrettably, is seen rolling around.

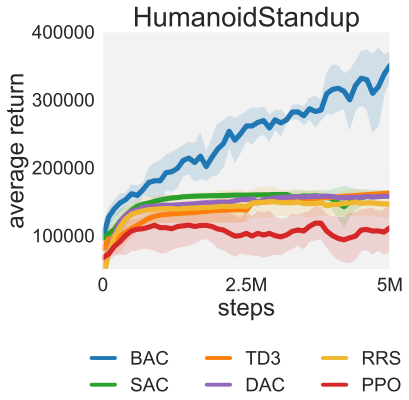


Figure 44. Learning curves of BAC and other baselines on HumanoidStandup task.

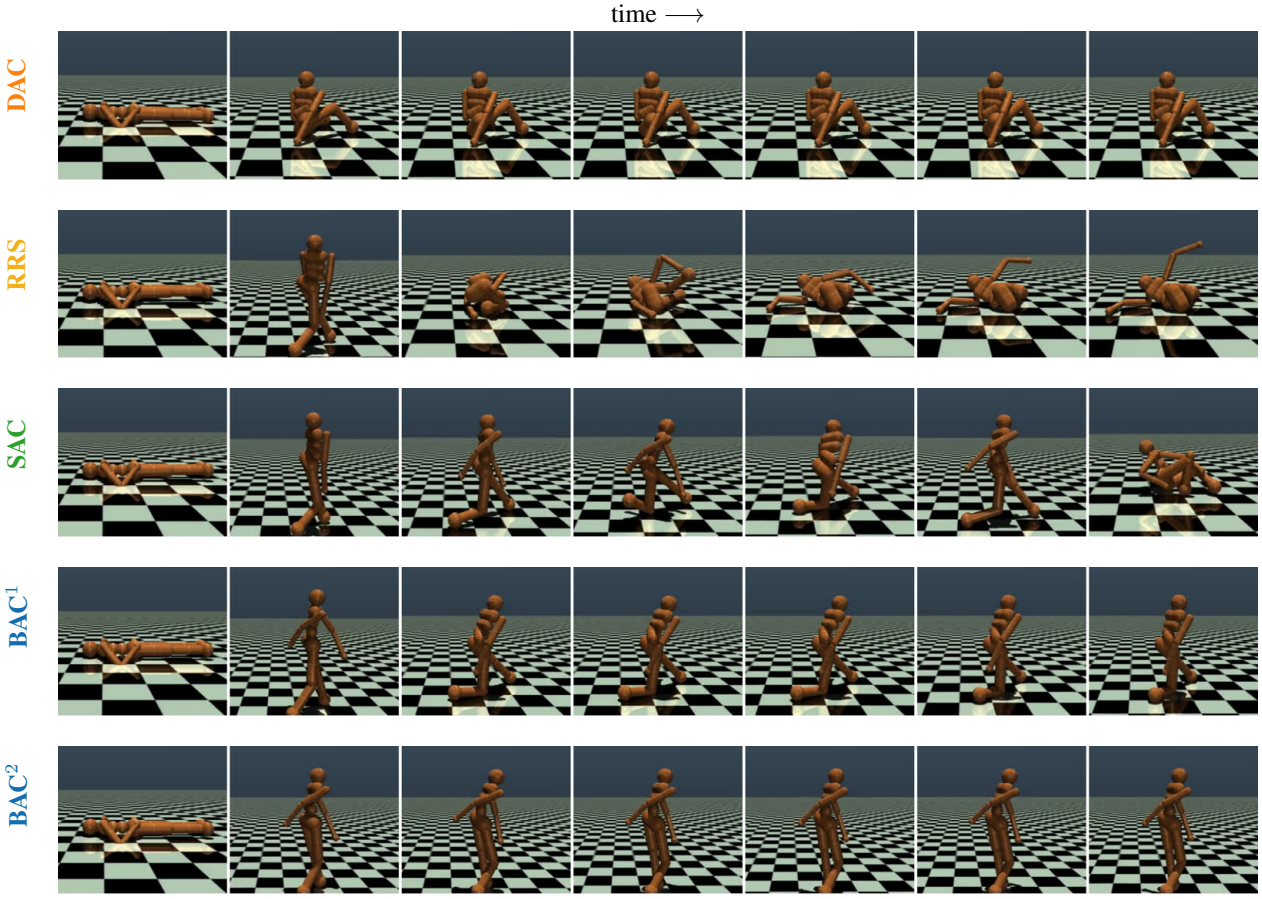


Figure 45. Visualization on HumanoidStandup task. BAC^1 is the visualization using the learned policy at 2.5M steps, and BAC^2 reveals the behaviors learned at 5M steps. For DAC, RRS, and SAC, we visualize the learned policy at 5M steps.

DogRun. DogRun, provided by the DMControl (Tunyasuvunakool et al., 2020), is a challenging task with a high-dimensional action space ($\mathcal{A} \in \mathbb{R}^{38}$). The task is based on a sophisticated model of a Pharaoh Dog, including intricate kinematics, skinning weights, collision geometry, as well as muscle and tendon attachment points. This complexity makes the DogRun task extremely difficult for algorithms to learn and control effectively.

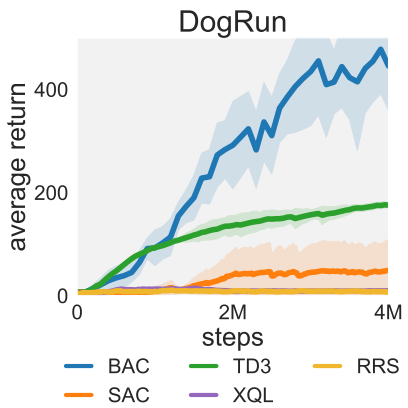


Figure 46. Learning curves BAC and other baselines on DogRun task.

We conducted extensive experiments in the DogRun task to compare the performance of BAC against other state-of-the-art algorithms. Here, we include Extreme Q-Learning (XQL) (Garg et al., 2023) as our baseline, which falls into the MaxEntropy RL framework but directly models the maximal Q value. The results, depicted in Figure 46, reveal that BAC significantly surpasses its counterparts, attaining higher average returns in fewer interactions. It demonstrates a remarkable capability of learning to control the high-dimensional, complex robot, such as facilitating the dog's run. To the best of our knowledge, ***it is the first documented result of model-free methods effectively tackling the challenging DogRun task.***

In addition to the quantitative results, we also offer a visualization of keyframes in the trajectory in Figure 47. Here, the superior performance of BAC becomes even more apparent. While competing algorithms struggle to prevent the dog from falling, BAC successfully achieves a running motion. This aptitude for handling complex, high-dimensional tasks further reaffirms the efficacy and robustness of BAC when dealing with failure-prone scenarios.

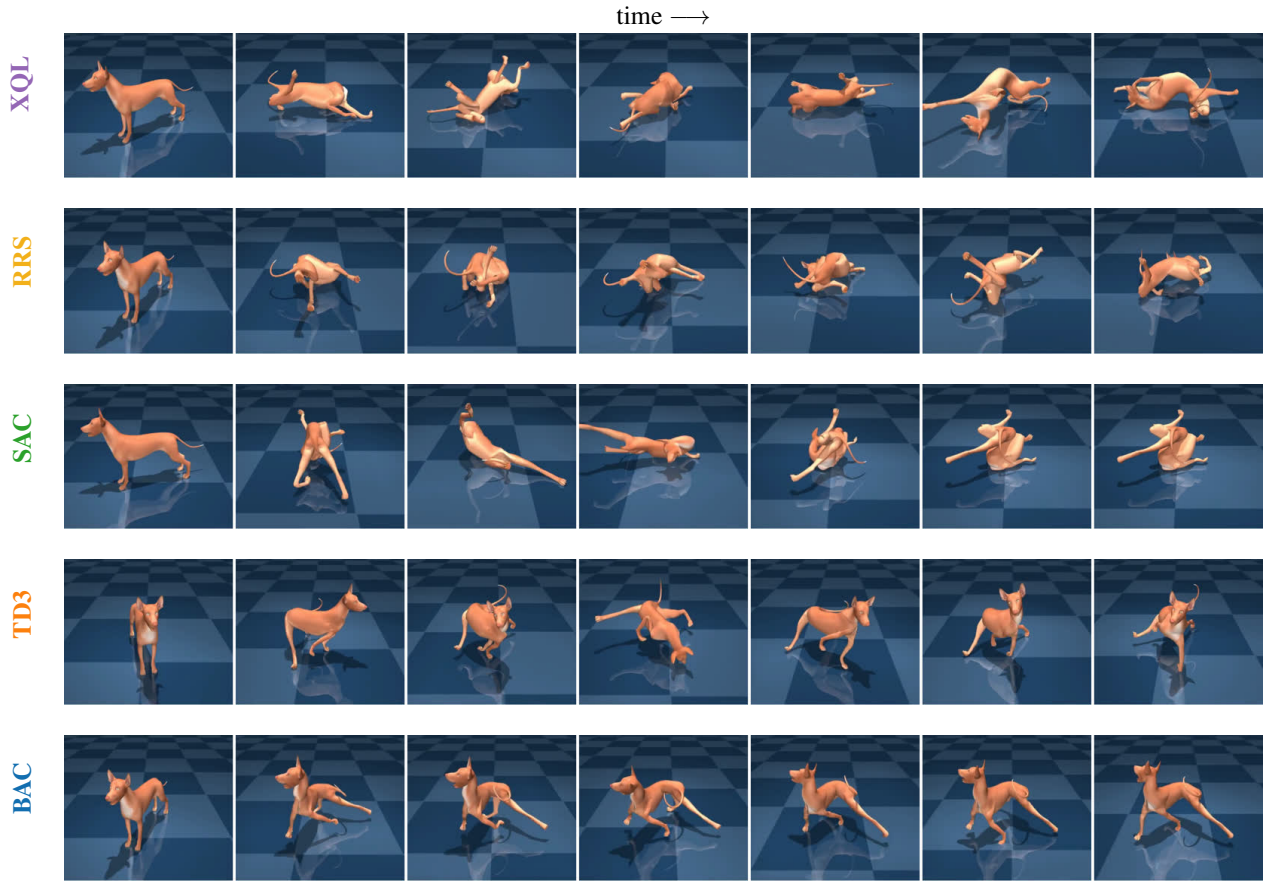


Figure 47. Visualization on DogRun task. We visualize the keyframes of the trajectories induced by the learned policy of each algorithm at 4M steps.

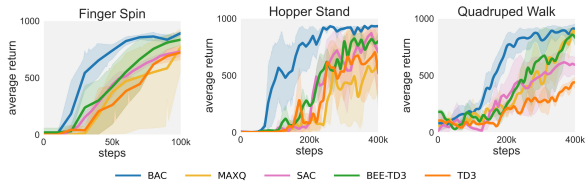
I. Additional Baselines Comparison

Our paper spotted the underestimation phenomenon, and other options may fix it.

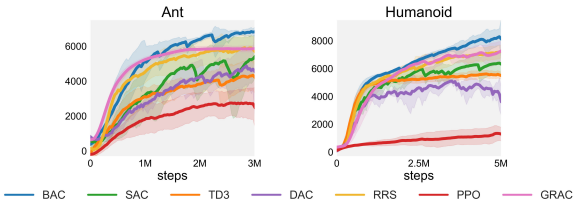
Here, we compare two baselines, $\max(Q(s', \pi(s')), Q(s', a'))$ and GRAC (Shao et al., 2022), as shown in Figure 48. However, both the MAXQ and GRAC lag behind BAC for a crucial reason: while they address underestimation, they inadvertently introduce extra overestimation, which can hinder performance. Interestingly, GRAC does outperform standard AC algorithms, underscoring the advantage of addressing underestimation to boost performance—one of the key contributions of our paper. Here's a deeper dive into these comparisons:

- Comparison to MAXQ: The MAXQ method, although simple, faces notable instability issues. When $Q(s', a')$ gets erroneously overestimated, a common issue in early training, it leads to overly optimistic outcomes from $\max(Q(s', \pi(s')), Q(s', a'))$, potentially leading to more severe overestimation issues.
- Comparison to GRAC: GRAC, by selecting the maximum Q-values around the policy π, does introduce a degree of optimism. However, this approach suffers from considerable overestimation (extrapolation error), especially when it encounters actions that are out of the distribution during training (Fujimoto et al., 2019). Such overestimation can negatively affect performance. Also, the additional computational load required for action sampling could be a drawback in resource-constrained environments.

BEE does not handle it one-sidedly. While the BEE seeks to alleviate underestimation, it might not incite extra overestimation. As the exploitation operator only relies on real experience and may lead to a more stable estimation, which would help to reduce overestimation practically.



(a) **Comparison with MAXQ.** Training curves of BAC, MAXQ, BEE-TD3, SAC, TD3 in DMControl benchmark tasks. Solid curves depict the mean of ten trials and shaded regions correspond to the one standard deviation.



(b) textbfComparison with GRAC. Training curves of BAC, GRAC, and other standard AC algorithms(SAC, TD3, DAC, RRS, and PPO) in MuJoCo benchmark tasks. Solid curves depict the mean of ten trials and shaded regions correspond to the one standard deviation. GRAC does outperform standard AC algorithms, underscoring the advantage of addressing underestimation to boost performance.

Figure 48. Performance comparison to two further baselines.

J. More Benchmark Results

Given that MuJoCo benchmark tasks have been solved well by popular baselines, we conduct experiments on the more complex locomotion and manipulation tasks from **DMControl** (Tunyasuvunakool et al., 2020), **Meta-World** (Yu et al., 2019), **Adroit** (Rajeswaran et al., 2017), **ManiSkill2** (Gu et al., 2023), **Shadow Dexterous Hand** (Plappert et al., 2018) and **MyoSuite** (Vittorio et al., 2022) for further evaluation of BAC and the baselines. Currently, several tasks in these benchmarks pose a formidable challenge that stumps most model-free methods. Notably, BAC has demonstrated its effectiveness by successfully solving many of these challenging tasks.

J.1. Evaluation on DMControl benchmark tasks

We tested BAC and its variant, BEE-TD3, on 15 continuous control tasks from DMControl. BAC successfully solves many challenging tasks like HumanoidRun, DogWalk, and DogRun, where both SAC and TD3 fail. Also, BEE-TD3 boosts TD3's performance by a large margin, demonstrating the generalizability of the BEE operator.

Trajectory Visualizations. Figure 49 provides visualizations of trajectories generated by BAC on five tasks from DMControl. For each trajectory, we display seven keyframes.

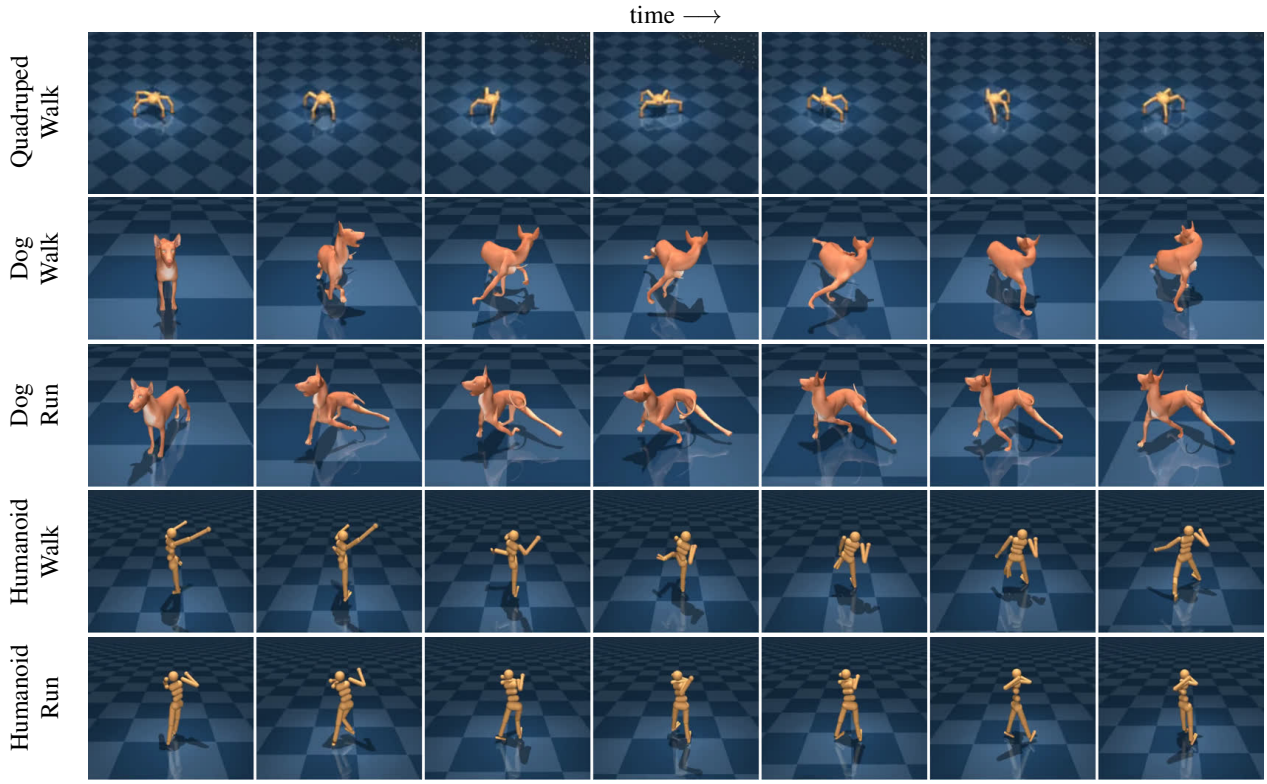


Figure 49. **Trajectory Visualizations.** Visualizations of the learned policy of BAC on five DMControl benchmark tasks.

Rliable metrics. We report additional (aggregate) performance metrics of BAC and SAC on the set of 15 DMControl tasks using the `rliable` toolkit (Agarwal et al., 2021). As shown in Figure 50, BAC outperforms SAC in terms of Median, interquantile mean (IQM), Mean, and Optimality Gap.

Performance comparison. Training curves for 15 DMControl tasks are shown in Figure 51. For simple locomotion/manipulation tasks (*e.g.*, HopperStand, WalkerStand, CupCatch), we generally find that while SAC's eventual performance is competitive with BAC, BAC shows better sample efficiency. In the more complex, failure-prone tasks (*e.g.*, HumanoidWalk, HumanoidRun, DogWalk, and DogRun), BAC significantly surpasses SAC. As shown in the visualizations[2],

[2]Please refer to `https://jity16.github.io/BEE/` for videos or Section D.4 for key frames.

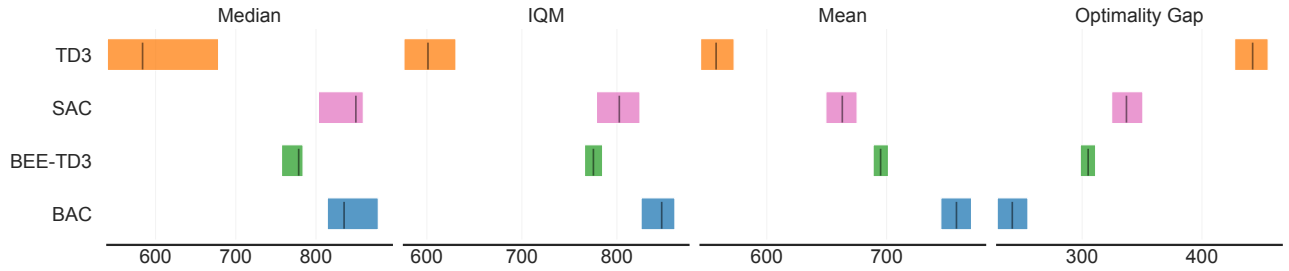


Figure 50. **Reliable metrics on DMControl tasks.** Median, IQM, Mean (higher values are better), and Optimality Gap (lower values are better) performance of BAC , BEE-TD3 and baselines (SAC, TD3) on the 15 DMControl tasks.

SAC struggles to learn meaningful behaviors in Dog Run, whereas the BAC agent yields superior performance.

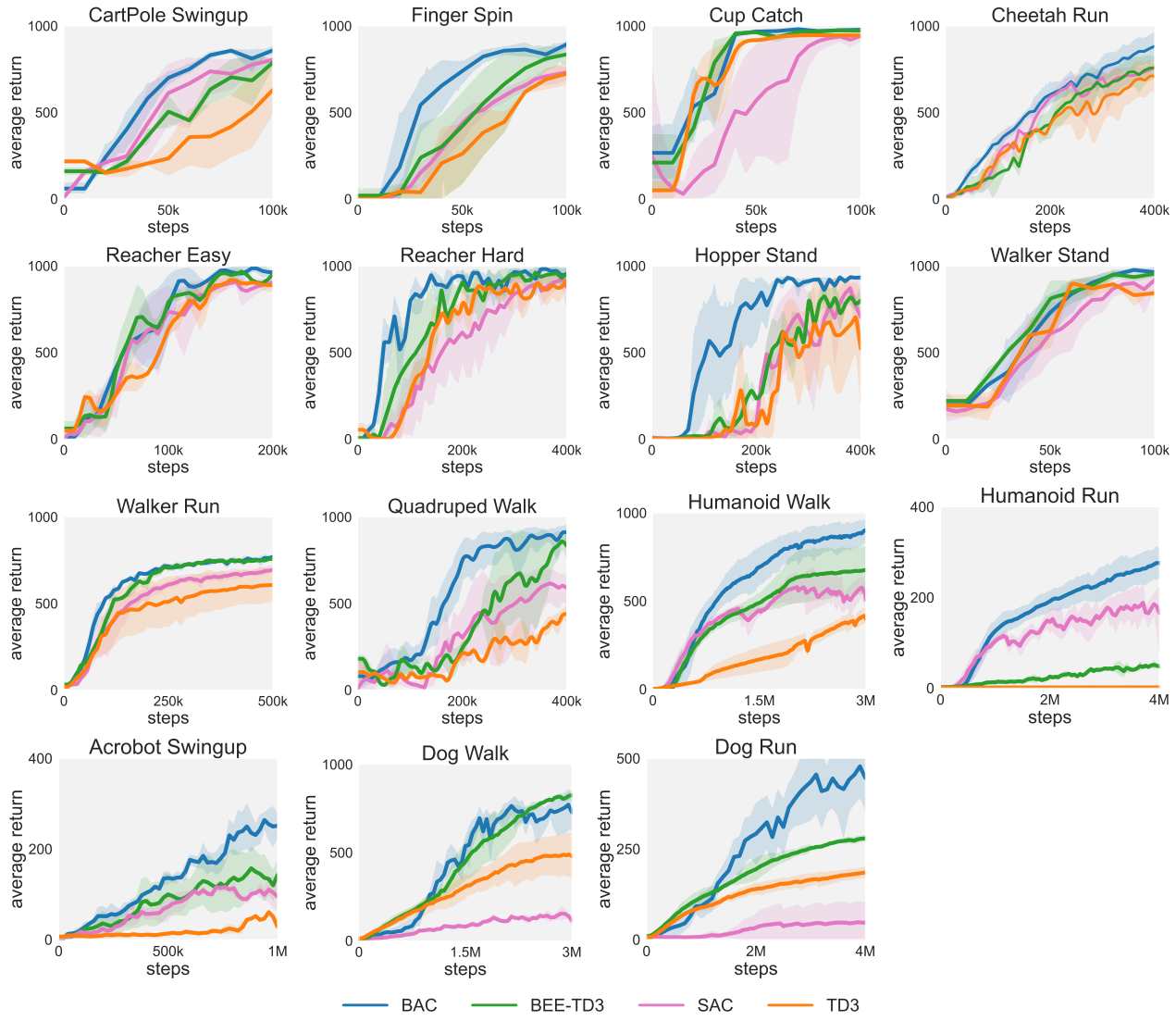


Figure 51. **DMControl tasks.** Training curves of BAC , BEE-TD3, SAC, TD3 in DMControl benchmark tasks. Solid curves depict the mean of ten trials and shaded regions correspond to the one standard deviation.

J.2. Evaluation on Meta-World benchmark tasks

Performance comparison. In Figure 53, we present learning curves of both the success rate and average return for twelve individual Meta-World tasks. Note that we conduct experiments on the goal-conditioned versions of the tasks from Meta-World-v2, which are considered harder than the single-goal variant.

In tasks typically categorized as simple, where both SAC and TD3 succeed within 1M steps, it is noteworthy that BAC still outperforms in terms of sample efficiency.

In tasks involving complex manipulation, such as pick place, basketball, hand insert, coffee push and hammer, BAC exhibits strong performance. Consider the hammer task, while SAC and TD3 occasionally achieve serendipitous successes before reaching 500K steps, their Q-value estimations are susceptible to misguidance by the inferior follow-up actions that occur frequently, resulting in a sustained low success rate. In contrast, BAC efficiently exploits the value of success and mitigates the impact of inferior samples on the Q-value, leading to a significant performance improvement beyond 500K steps, and finally surpasses SAC and TD3 by a large margin in terms of success rate.

These results highlight the promising potential of BAC in manipulation tasks.

Trajectory Visualizations. Successful trajectories for one simple task and five aforementioned complex tasks are visualized in Figure 52. For each trajectory, we display seven keyframes.

Figure 52. **Trajectory Visualizations.** We visualize trajectories generated by BAC on six Meta-World tasks. Our method (BAC) is capable of solving each of these tasks within 1M steps.

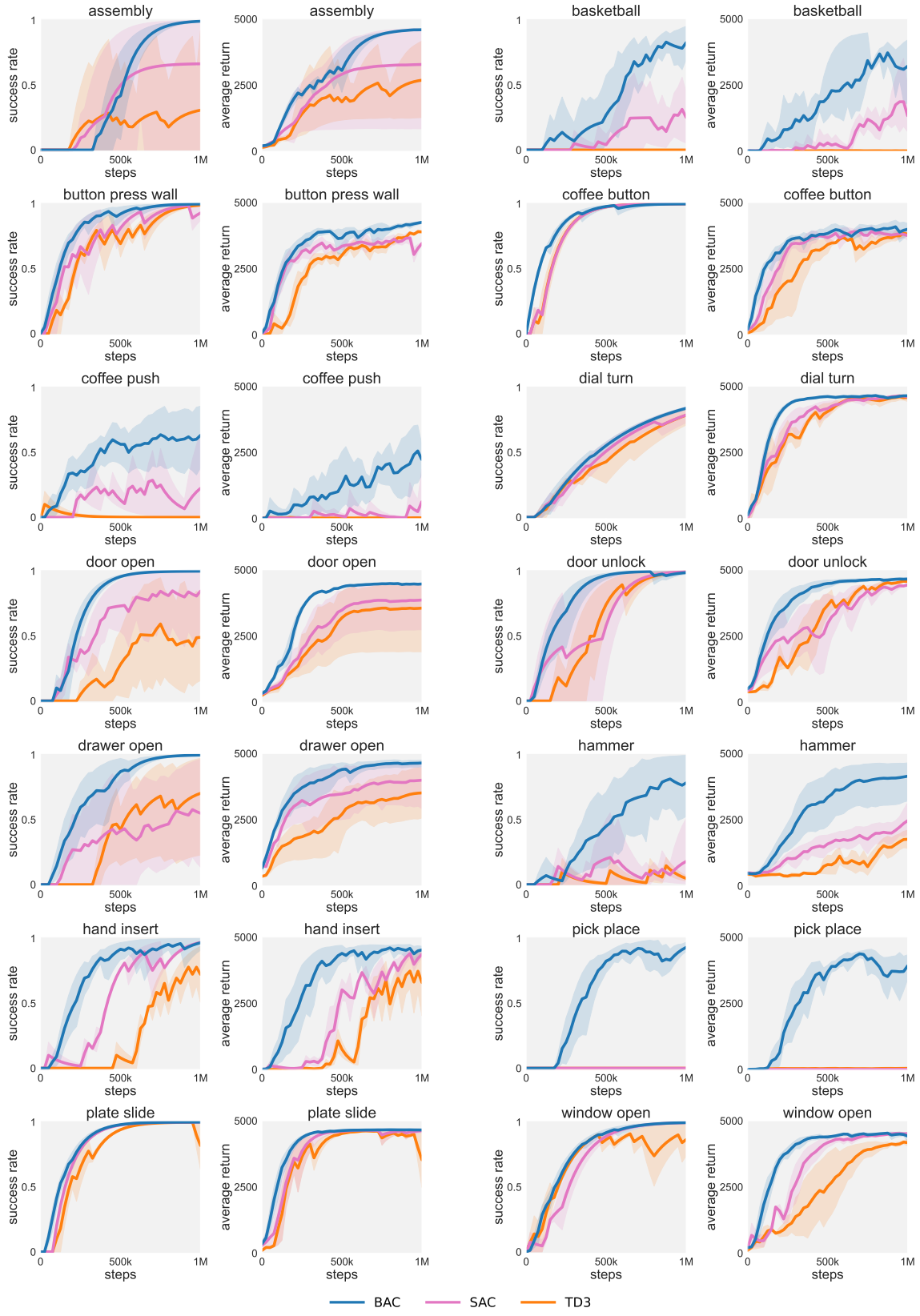


Figure 53. **Individual Meta-World tasks.** Success rate and average return of BAC , SAC, TD3 on twelve manipulation tasks from MetaWorld (sorted alphabetically). Solid curves depict the mean of ten trials, and shaded regions correspond to the one standard deviation.

J.3. Evaluation on Adroit benchmark tasks

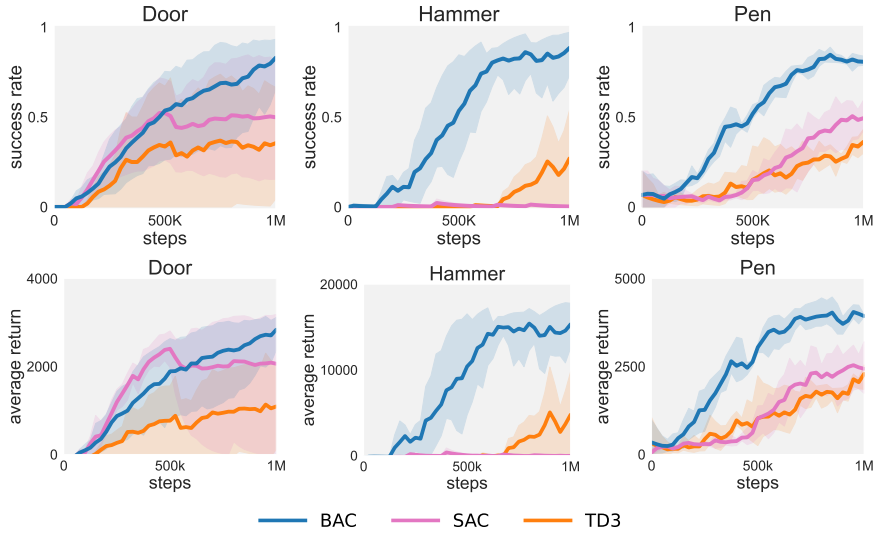


Figure 54. **Adroit tasks.** Success rate and average return of BAC , SAC, TD3 on Adroit benchmark tasks. Solid curves depict the mean of ten trials, and shaded regions correspond to the one standard deviation.

J.4. Evaluation on MyoSuite benchmark tasks

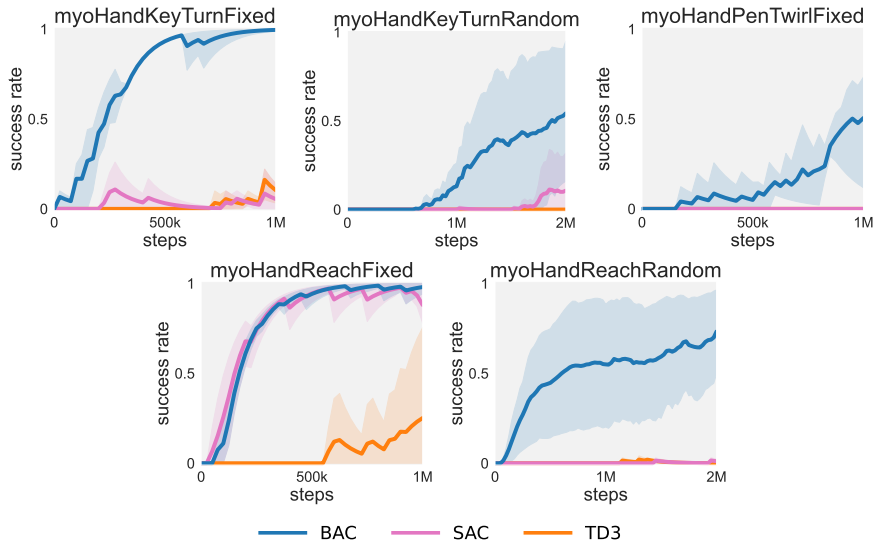


Figure 55. **MyoSuite tasks.** Success rate of BAC , SAC, TD3 on MyoSuite benchmark tasks. Solid curves depict the mean of ten trials, and shaded regions correspond to the one standard deviation.

J.5. Evaluation on ManiSkill2 benchmark tasks

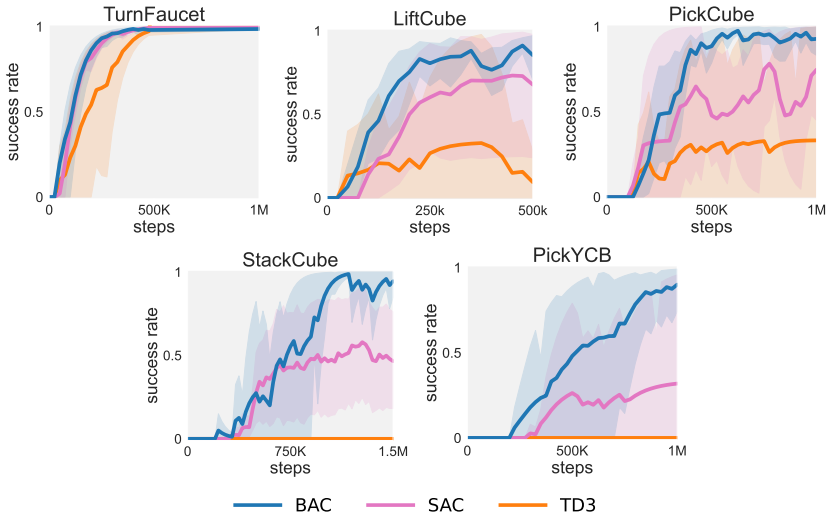


Figure 56. **ManiSkill2 tasks.** Success rate of BAC , SAC, TD3 on MainSkill2 benchmark tasks. Solid curves depict the mean of ten trials, and shaded regions correspond to the one standard deviation.

J.6. Evaluation on Shadow Dexterous Hand benchmark tasks

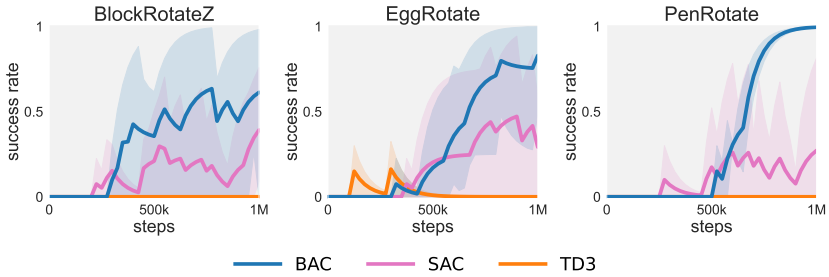


Figure 57. **Shadow Dexterous Hand tasks.** Success rate of BAC , SAC, TD3 on Shadow Dexterous Hand benchmark tasks. Solid curves depict the mean of ten trials, and shaded regions correspond to the one standard deviation.

J.7. Evaluation on Swimmer benchmark tasks with an extremely high γ.

In our main paper, we adopt the same $\gamma = 0.99$ parameters for all our tasks adhering to the community's advocacy for using a uniform set of parameters for all tasks. Simply applying the trick - setting $\gamma = 0.9999$ recommended by Franceschetti et al. (2022), we observe BAC, SAC, and TD3 all surpass 300 before reaching 1M steps. Notably, BAC still outperforms.

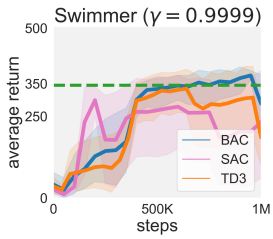


Figure 58. Performance curves of BAC, SAC, TD3 with the discount factor $\gamma = 0.9999$ for Swimmer-v2 task. Run on 10 seeds.