MITIGATING FILTERING-INDUCED BIAS IN OFFLINE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Value-based reinforcement learning can suffer from overestimation bias when noisy value estimates are repeatedly used in bootstrapped updates. This issue becomes particularly pronounced with large discount factors or long decision horizons, where estimation errors can persist and accumulate through repeated Bellman backups. We analyze this bias propagation mechanism and further show that, even without explicit action maximization, adaptive filtering can introduce bias when filtering decisions become coupled with value-estimation errors. To mitigate this coupling, we propose two mechanisms: delayed filtering and decoupled filtering. We instantiate these mechanisms in IQL, yielding DD-IQL. Empirically, DD-IQL exhibits reduced filter–error correlation and improved value stability. Across a range of offline RL environments, DD-IQL also improves policy performance over standard IQL, with particularly large gains under high bias-stress settings.

1 INTRODUCTION

Reinforcement learning (RL) has shown strong potential for solving sequential decision-making problems (Schulman et al., 2015; 2017; Mnih, 2013). However, applying RL to complex real-world domains remains challenging (Kalashnikov et al., 2021; Tang & Wiens, 2021; Zhan et al., 2022). One fundamental difficulty is that value-based RL relies on recursive Bellman backups, where current learning targets are constructed from estimated future values (Sutton, 1988). When these estimates are noisy, policy improvement can preferentially select positive estimation errors and produce overestimation bias (Hasselt, 2010; Van Hasselt et al., 2016; Fujimoto et al., 2018; Kuznetsov et al., 2020).

This problem becomes particularly important in offline reinforcement learning, where the agent must learn from a fixed dataset without additional environment interaction (Siegel et al., 2020; Jaques et al., 2019; Levine et al., 2020; Prudencio et al., 2023; Ernst et al., 2005; Fujimoto et al., 2019; Agarwal et al., 2020; Kumar et al., 2019). As policy improvement progresses, the learned policy can move away from the behavior policy that generated the data and select actions that are poorly covered by the dataset. Value estimates for such actions can be unreliable, and offline RL cannot collect new data to correct them. Thus, distribution shift, value overestimation, and bootstrapping can interact to destabilize offline value learning.

A common way to reduce this instability is to avoid hard maximization over unreliable actions. Implicit Q-Learning (IQL) (Kostrikov et al., 2021), for example, learns a state-value function from in-dataset actions using expectile regression and extracts a policy through advantage-weighted behavior cloning (Nair et al., 2020). This in-sample formulation avoids explicit maximization over out-of-distribution actions and can reduce the direct selection of unreliable actions.

In this paper, we focus on the observation that the expectile update in IQL can be interpreted as an adaptive filtered expectation over dataset actions. Under this view, actions with larger estimated values receive larger filtering weights. However, when these weights are computed from the same noisy value estimates being updated, positive estimation errors can be preferentially emphasized and then propagated through bootstrapped targets. Because the resulting biased estimates are reused in subsequent filtering decisions, this effect can persist across Bellman backups, particularly for larger discount factors and longer effective horizons.

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

Based on this observation, we propose delayed and decoupled filtering as a bias-mitigation principle for adaptive filtered value learning. The key idea is to reduce the dependence between filtering decisions and the estimation errors of the value function being updated. Delayed filtering constructs filtering weights from delayed network snapshots, while decoupled filtering constructs the filtering weight using an estimator different from the one receiving the weighted update.

Our contributions are summarized as follows:

- We identify filtering-induced bias in value learning and connect it to the propagation of noisy value estimates through bootstrapped Bellman updates.
- We propose delayed and decoupled filtering, a bias-mitigation principle designed to weaken the dependence between filtering decisions and value-estimation errors.
- We instantiate these mechanisms in IQL, yielding DD-IQL, and empirically show reduced filter–error coupling, improved value stability, and stronger policy performance across a range of offline RL environments.
- As an additional extension, we apply a lightweight analogue of the proposed principle to ReBRAC (Tarasov et al., 2023) to examine whether the underlying design idea can also be useful in a policy-network-based setting.

2 Related Work

2.1 Overestimation Bias and Bias-Reduction Methods

Overestimation bias has been widely studied in value-based reinforcement learning. This bias arises because noisy value estimates are used for both evaluation and action selection, causing learning targets to favor actions with spuriously high values (Hasselt, 2010; Van Hasselt et al., 2016; Fujimoto et al., 2018; Kuznetsov et al., 2020). A classical solution is Double Q-learning, which reduces overestimation by using one action-value estimator to select the action and another to evaluate it (Hasselt, 2010; Van Hasselt et al., 2016).

In continuous-control actor-critic algorithms, clipped double Q-learning has become a standard bias-reduction technique. It constructs a conservative target by taking the minimum of two critic estimates, thereby reducing overestimation compared with using a single critic (Fujimoto et al., 2018). Due to its simplicity and empirical effectiveness, this mechanism has been widely adopted in later algorithms (Haarnoja et al., 2018; Jiang et al., 2022).

Other bias-reduction approaches have been explored from several perspectives. Some methods explicitly balance overestimation and underestimation in value learning through adaptive or regularized value estimators (Ren et al., 2021; Lyu et al., 2022; Döring et al., 2025). Ensemble-based approaches instead combine multiple value estimators to control estimation bias and stabilize target construction (Lan et al., 2020; Peer et al., 2021; Chen et al., 2021; Schmitt-Förster & Sutter, 2024). Another line of work replaces the hard maximum with smoother Bellman operators, such as softmax-based backups, to mitigate the sharp selection of spuriously overestimated actions (Song et al., 2019; Pan et al., 2020; Kim et al., 2019).

2.2 Offline Reinforcement Learning

Offline reinforcement learning aims to learn a policy from a fixed dataset without additional environment interaction (Siegel et al., 2020; Jaques et al., 2019; Levine et al., 2020; Prudencio et al., 2023; Ernst et al., 2005; Fujimoto et al., 2019; Agarwal et al., 2020; Kumar et al., 2019). A central challenge in offline RL is distribution shift between the behavior policy that generated the dataset and the learned policy. As policy improvement progresses, the learned policy may select actions that are poorly represented or absent from the dataset, making their value estimates unreliable. Since offline RL cannot collect additional data to correct these errors, overestimated values on poorly supported actions can persist and be repeatedly reused through bootstrapped Bellman updates (Fujimoto et al., 2019; Kumar et al., 2019; Levine et al., 2020).

Existing offline RL methods address this issue from several complementary directions. Policy-constraint and behavior-regularized methods restrict the learned policy to remain close to the behav-

ior distribution or to select actions well supported by the dataset (Jaques et al., 2019; Wu et al., 2019; Tarasov et al., 2023). Pessimistic and uncertainty-aware methods instead regularize value estimates by penalizing or downweighting actions whose values are unreliable (Kumar et al., 2020; Wu et al., 2021; Bai et al., 2022). Another line of work avoids explicit maximization over out-of-distribution actions by performing value learning and policy improvement over in-dataset actions. For example, IQL (Kostrikov et al., 2021) learns a state value through expectile regression over dataset actions and extracts a policy using advantage-weighted behavior cloning (Nair et al., 2020). Related in-sample methods similarly avoid querying policy-generated out-of-distribution actions (Xu et al., 2023; Chen et al., 2025; Garg et al., 2023).

Our work focuses on a source of bias that can remain even when policy improvement is restricted to in-dataset actions. In methods such as IQL, removing explicit maximization over out-of-distribution actions does not eliminate value-dependent weighting, since the adaptive filtering weights are themselves constructed from learned value estimates. The resulting filtering-induced bias is not directly targeted by policy constraints, which primarily restrict policy deviation from the data distribution, or by pessimistic value regularization, which primarily discourages overestimation of uncertain or poorly supported actions. Delayed and decoupled filtering instead target the dependence between filtering decisions and value-estimation errors.

3 Preliminaries

3.1 Markov Decision Process

We consider an infinite-horizon discounted Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma, \rho_0)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition kernel, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\gamma \in (0, 1)$ is the discount factor, and ρ_0 is the initial-state distribution. A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ induces a trajectory distribution p_π over trajectories $h = (s_0, a_0, s_1, a_1, \ldots)$, where $s_0 \sim \rho_0$, $a_t \sim \pi(\cdot \mid s_t)$, and $s_{t+1} \sim P(\cdot \mid s_t, a_t)$. The expected discounted return of π is

$$J(\pi) = \mathbb{E}_{h \sim p_\pi} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]. \tag{1}$$

The objective of reinforcement learning is to find an optimal policy $\pi^\star \in \arg\max_\pi J(\pi)$. The action-value function of a policy π is defined as $Q^\pi(s, a) = \mathbb{E}_{\pi,P}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, a_0 = a]$. The Bellman optimality operator $\mathcal{T}^\star$ is defined by

$$(\mathcal{T}^\star Q)(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[\max_{a' \in \mathcal{A}} Q(s', a') \right]. \tag{2}$$

The optimal action-value function $Q^\star$ is the fixed point of this operator, i.e., $Q^\star = \mathcal{T}^\star Q^\star$.

3.2 Offline Reinforcement Learning

In offline reinforcement learning, the agent learns from a fixed dataset $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^M$ without further interaction with the environment. The transitions in $\mathcal{D}$ are collected by an unknown behavior policy π_β, and the goal is to learn a policy using only this fixed dataset.

A central challenge in offline RL is the out-of-distribution (OOD) action problem. Since the dataset covers only a limited part of the state-action space, value estimates for rarely observed or unseen actions can be unreliable. This issue becomes particularly problematic when a Bellman backup explicitly maximizes over actions, because the maximization operator may select actions with spuriously high estimated values outside the data support.

Implicit Q-Learning (IQL) addresses this issue by avoiding explicit maximization over actions outside the dataset (Kostrikov et al., 2021). Instead of using a target of the form $\max_{a'} Q(s', a')$, IQL learns a state-value function from in-dataset actions and uses it to construct the critic target:

$$\mathcal{L}_Q(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}} \left[\left(Q_{\theta_i}(s, a) - (r + \gamma V_\psi(s')) \right)^2 \right], \qquad i \in \{1, 2\} \tag{3}$$

The value function V_ψ is trained by expectile regression:

$$\mathcal{L}_V(\psi) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[L_2^\tau\left(Q_{\bar{\theta}}^{\min}(s,a) - V_\psi(s)\right)\right], \qquad L_2^\tau(u) = |\tau - \mathbf{1}\{u<0\}|\, u^2, \tag{4}$$

where $Q_{\bar{\theta}_i}$ denotes the i-th target, or slowly updated, critic, and $Q_{\bar{\theta}}^{\min}(s,a) = \min_{i\in\{1,2\}} Q_{\bar{\theta}_i}(s,a)$. By choosing $\tau > 1/2$, the learned value function is fitted closer to higher-valued in-dataset actions while still avoiding explicit maximization over unseen actions. Thus, IQL performs an implicit form of policy improvement over dataset actions, which makes it well suited for offline RL.

4 Bias Propagation in Value Learning

4.1 Maximization-Induced Bias and Its Propagation

We first recall how overestimation bias arises in max-based value learning (Thrun & Schwartz, 2014; Hasselt, 2010); this mechanism serves as the reference point for the filtering-induced bias analyzed in Section 5. Suppose the learned critic decomposes as

$$\widehat{Q}_k(s,a) = \overline{Q}_k(s,a) + \epsilon_k(s,a), \tag{5}$$

where $\overline{Q}_k(s,a) = \mathbb{E}_\epsilon[\widehat{Q}_k(s,a)]$ and $\epsilon_k(s,a)$ is a zero-mean estimation error arising from finite samples, stochastic transitions, optimization error, or function approximation.

Although ϵ_k is zero-mean for each action, it is not zero-mean after maximization. Since the maximum is convex, Jensen's inequality gives

$$\mathbb{E}_\epsilon\Big[\max_{a\in\mathcal{A}}\big(\overline{Q}_k(s,a) + \epsilon_k(s,a)\big)\Big] \;\geq\; \max_{a\in\mathcal{A}}\overline{Q}_k(s,a), \tag{6}$$

so the maximum tends to select actions whose errors are accidentally positive. Denote the resulting one-step bias by $\eta_k(s) \geq 0$, given by the difference between the two sides of equation 6.

This bias does not stay local, because the maximized value forms a bootstrapped target. With $\mathcal{T}^\star$ the Bellman optimality operator defined in equation 2,

$$\mathbb{E}_\epsilon\big[(\mathcal{T}^\star\widehat{Q}_k)(s,a)\big] - (\mathcal{T}^\star\overline{Q}_k)(s,a) = \gamma\,\mathbb{E}_{s'\sim P(\cdot|s,a)}\big[\eta_k(s')\big] \;\geq\; 0. \tag{7}$$

Once this target is fitted, the critic inherits the bias, and the next backup bootstraps from the biased critic. The error therefore recurs across updates rather than remaining a one-step effect.

4.2 Why Bias Propagation Matters

To quantify the accumulation, assume as a first-order estimate that each backup contributes an average one-step bias $\eta_{\mathcal{A}}$. Over an effective horizon H,

$$B_H \;\approx\; \sum_{h=1}^{H}\gamma^h\eta_{\mathcal{A}} \;=\; \eta_{\mathcal{A}}\,\frac{\gamma(1-\gamma^H)}{1-\gamma}. \tag{8}$$

This expression shows that bias propagation becomes more severe when the discount factor is close to one or the effective horizon is long. The per-step bias $\eta_{\mathcal{A}}$ additionally depends on how many candidate actions the selection operator ranges over; Appendix B gives this dependence for finite action sets and discusses the continuous case. Notably, these conditions can commonly arise in real-world decision-making problems, where agents must reason over long horizons while selecting among complex actions. This motivates the need for bias-control mechanisms that improve the stability of bootstrapped value learning under such conditions.

5 Filtered Expectation and Remaining Bias

The previous section showed that hard maximization over noisy value estimates can generate overestimation bias and propagate it through Bellman bootstrapping. A natural way to mitigate this bias is to replace the hard maximum with an expectation over actions. Given a reference action distribution $\mu(\cdot \mid s)$, such as the behavior policy in offline RL, consider $F_{\text{avg}}(Q)(s) = \mathbb{E}_{a\sim\mu(\cdot|s)}\left[Q(s,a)\right]$.

Unlike the maximum operator, this expectation aggregates values across actions rather than selecting the action with the largest estimated value. As a result, zero-mean estimation errors can partially cancel instead of being systematically amplified by selection. However, a plain expectation only evaluates the reference distribution μ and does not perform policy improvement.

A filtered expectation provides a compromise between plain averaging and hard maximization. Instead of averaging all actions equally, it assigns larger weights to actions with larger estimated values:

$$F_f(Q)(s) = \frac{\mathbb{E}_{a\sim\mu(\cdot|s)}\left[f(Q(s,a))Q(s,a)\right]}{\mathbb{E}_{a\sim\mu(\cdot|s)}\left[f(Q(s,a))\right]}, \tag{9}$$

where $f : \mathbb{R} \to \mathbb{R}_+$ is a positive filtering function. When f is constant, equation 9 reduces to the plain expectation. When f assigns larger weights to higher-valued actions, the operator performs a soft form of policy improvement while avoiding the sharp selection of a hard maximum.

IQL can be interpreted as an example of this filtered-expectation principle. As described in Section 3.2, IQL avoids explicit maximization over unseen actions and learns a value function from in-dataset actions. At the optimum of the expectile value loss, the IQL value V_ψ can be written as a weighted expectation over behavior actions:

$$V_\psi(s) = \frac{\mathbb{E}_{a\sim\mu(\cdot|s)}\left[w_\tau(s,a)Q^{\min}_{\bar\theta}(s,a)\right]}{\mathbb{E}_{a\sim\mu(\cdot|s)}\left[w_\tau(s,a)\right]}, \qquad w_\tau(s,a) = \begin{cases}\tau, & Q^{\min}_{\bar\theta}(s,a) > V_\psi(s),\\ 1-\tau, & Q^{\min}_{\bar\theta}(s,a) \le V_\psi(s).\end{cases} \tag{10}$$

For $\tau > 1/2$, actions whose estimated values are above the current value estimate receive larger weights. Therefore, IQL can be viewed as an upper-tail filtered expectation over dataset actions.

Nevertheless, filtered expectation does not eliminate bias by itself. The key issue is that the filtering weights are computed from the same noisy estimates used to construct the filtered value. To see this, let $\widehat{q}(a) = q(a) + \epsilon(a)$ with $\mathbb{E}[\epsilon(a) \mid a] = 0$. The filtered value computed from the noisy estimate can be decomposed as

$$\widehat{F}_f = \frac{\mathbb{E}_{a\sim\mu}\left[f(\widehat{q}(a))\widehat{q}(a)\right]}{\mathbb{E}_{a\sim\mu}\left[f(\widehat{q}(a))\right]} = \frac{\mathbb{E}_{a\sim\mu}\left[f(\widehat{q}(a))q(a)\right]}{\mathbb{E}_{a\sim\mu}\left[f(\widehat{q}(a))\right]} + \frac{\mathbb{E}_{a\sim\mu}\left[f(\widehat{q}(a))\epsilon(a)\right]}{\mathbb{E}_{a\sim\mu}\left[f(\widehat{q}(a))\right]}. \tag{11}$$

The second term captures a direct estimation-error contribution to the filtered value. Although $\epsilon(a)$ is zero-mean before filtering, it is multiplied by $f(\widehat{q}(a))$, a weight that depends on the same noisy estimate. In particular,

$$\mathbb{E}_\epsilon\left[f(q(a)+\epsilon(a))\epsilon(a) \mid a\right] = \mathrm{Cov}_\epsilon\left(f(q(a)+\epsilon(a)), \epsilon(a) \mid a\right). \tag{12}$$

Thus, zero-mean estimation errors do not necessarily cancel after adaptive filtering. If f is increasing, positive errors tend to increase both the estimated value and its filtering weight, while negative errors tend to receive smaller weights. This covariance captures a direct filter–error coupling, while additional bias can arise through noise-dependent reweighting and normalization (Appendix A).

When the filtering weights are computed from the same noisy estimates used in the weighted update, the filtered value can therefore remain biased. Once incorporated into Bellman targets, this bias can propagate through subsequent bootstrapped updates, motivating mechanisms that reduce the dependence between filtering weights and value-estimation errors.

6 Delayed Decoupled IQL

Based on the filtering-induced bias discussed in Section 5, we introduce two bias-control mechanisms targeting distinct forms of dependence: delayed filtering and decoupled filtering. In IQL, the expectile weight is constructed from estimated action values and is used to update the value function, creating dependence between the filtering decision and value-estimation errors. Delayed filtering weakens this dependence across optimization steps, while decoupled filtering weakens it across estimators. Applying both mechanisms to IQL yields Delayed Decoupled IQL (DD-IQL).

6.1 Delayed Filtering

In standard IQL, the expectile weight is recomputed from the current critic and value estimates at every update. Therefore, the filtering weight can immediately react to transient estimation errors, assigning larger weights to actions whose values are temporarily overestimated.

Delayed filtering reduces this temporal coupling by constructing the filtering weight from delayed network snapshots. Specifically, we maintain delayed copies of the critic and value function, denoted by $\widetilde{Q}$ and $\widetilde{V}$. The delayed advantage and the corresponding expectile weight are defined as

$$\widetilde{A}(s,a) = \widetilde{Q}(s,a) - \widetilde{V}(s), \qquad \widetilde{w}_\tau(s,a) = \left|\tau - \mathbf{1}\left\{\widetilde{A}(s,a) < 0\right\}\right|. \tag{13}$$

The delayed networks are refreshed only every D update steps by setting $\widetilde{Q} \leftarrow Q_{\bar{\theta}}$ and $\widetilde{V} \leftarrow V_\psi$. Between two refreshes, the filtering weight remains fixed while the current critic and value networks continue to be updated. Thus, the filter does not immediately follow short-term fluctuations in the current bootstrapped estimates and can weaken the temporal dependence between filtering decisions and instantaneous update errors.

6.2 Decoupled Filtering

Delayed filtering weakens temporal coupling, but the same estimator may still construct the filtering weight and receive the corresponding weighted update. In this case, the filtering weight can remain correlated with the estimation error of the estimator being updated.

To further reduce this estimator-level dependence, DD-IQL separates filter construction from the weighted update. It maintains an ensemble of value estimators $\{(Q_{\theta_i}, Q_{\bar{\theta}_i}, V_{\psi_i})\}_{i=1}^N$ together with their delayed copies $\{(\widetilde{Q}_i, \widetilde{V}_i)\}_{i=1}^N$. For each estimator i, the filtering weight is constructed using a different delayed estimator $j_m(i)$, where m denotes the current delay interval:

$$\widetilde{A}_{j_m(i)}(s,a) = \widetilde{Q}_{j_m(i)}(s,a) - \widetilde{V}_{j_m(i)}(s), \qquad w_i(s,a) = \left|\tau - \mathbf{1}\left\{\widetilde{A}_{j_m(i)}(s,a) < 0\right\}\right|. \tag{14}$$

Here, $j_m(i) \neq i$, and the assignment j_m is updated every D steps together with the delayed-network refresh. Using this decoupled filtering weight, estimator i updates its value and critic networks:

$$\mathcal{L}_{V_i}(\psi_i) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[w_i(s,a)\left(Q_{\bar{\theta}_i}(s,a) - V_{\psi_i}(s)\right)^2\right], \tag{15}$$

$$\mathcal{L}_{Q_i}(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(Q_{\theta_i}(s,a) - (r + \gamma V_{\psi_i}(s'))\right)^2\right]. \tag{16}$$

Thus, using a different estimator to construct the filtering weight removes the direct self-coupling between filtering decisions and the estimator being updated. While the estimators may still have correlated errors, decoupled filtering is designed to weaken this dependence, and refreshing j_m every D steps can further reduce persistent estimator-specific correlations.

Overall, delayed filtering and decoupled filtering target distinct forms of filter–estimator dependence. Delayed filtering introduces temporal separation between the filtering reference and the evolving estimates, while decoupled filtering separates filter construction from the estimator receiving the weighted update. Together, these mechanisms preserve the filtered-expectation structure of IQL while mitigating filtering-induced bias.

6.3 Actor Update

After learning the value and critic ensemble, DD-IQL trains the policy using advantage-weighted behavior cloning, following the policy extraction step of IQL. The actor advantage is computed from the ensemble-mean critic and value estimates:

$$A_{\text{actor}}(s,a) = \frac{1}{N}\sum_{i=1}^N Q_{\bar{\theta}_i}(s,a) - \frac{1}{N}\sum_{i=1}^N V_{\psi_i}(s). \tag{17}$$

The policy is optimized with

$$\mathcal{L}_\pi(\phi) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\exp\left(\beta A_{\text{actor}}(s,a)\right)\left(-\log \pi_\phi(a \mid s)\right)\right], \tag{18}$$

where $\beta > 0$ controls the strength of advantage weighting. The full training procedure of DD-IQL is provided in Appendix D.1.

Table 1: Performance on OGBench navigation tasks under the original discount-factor settings and the bias-stress setting ($\gamma = 0.999$). Results are mean $\pm$ standard deviation over the training seeds.

	AM-L	HM-M	HM-L	AS-A	Cube-S	Cube-D	Pz-3x3	Pz-4x4	Scene	Avg
Original discount-factor settings ($\gamma = 0.99$ or $\gamma = 0.995$)										
IQL ($\tau = 0.9$)	0.14 ± 0.03	0.44 ± 0.05	0.05 ± 0.02	0.06 ± 0.03	0.84 ± 0.07	0.00 ± 0.00	0.01 ± 0.00	0.04 ± 0.01	0.11 ± 0.09	0.19
IQL ($\tau = 0.99$)	0.00 ± 0.00	0.25 ± 0.05	0.03 ± 0.01	0.03 ± 0.01	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.00	0.00 ± 0.00	0.04
ReBRAC	0.97 ± 0.02	0.16 ± 0.05	0.03 ± 0.02	0.01 ± 0.01	0.87 ± 0.05	0.08 ± 0.03	0.04 ± 0.02	0.08 ± 0.04	0.49 ± 0.07	0.30
FQL	0.83 ± 0.03	0.26 ± 0.20	0.00 ± 0.00	0.37 ± 0.03	0.95 ± 0.01	0.28 ± 0.05	0.12 ± 0.05	0.11 ± 0.02	0.77 ± 0.14	0.41
DD-IQL ($\tau = 0.9$)	0.28 ± 0.03	0.49 ± 0.00	0.07 ± 0.03	0.05 ± 0.01	0.88 ± 0.06	0.00 ± 0.00	0.01 ± 0.00	0.05 ± 0.03	0.03 ± 0.00	0.21
DD-IQL ($\tau = 0.99$)	0.50 ± 0.00	0.52 ± 0.02	0.13 ± 0.03	0.04 ± 0.00	0.93 ± 0.01	0.00 ± 0.00	0.00 ± 0.00	0.02 ± 0.01	0.38 ± 0.18	0.28
DD-ReBRAC	0.98 ± 0.01	0.39 ± 0.12	0.39 ± 0.10	0.43 ± 0.05	0.91 ± 0.02	0.32 ± 0.05	0.04 ± 0.02	0.10 ± 0.02	0.76 ± 0.05	0.48
Bias-stress setting ($\gamma = 0.999$)										
IQL ($\tau = 0.9$)	0.09 ± 0.00	0.46 ± 0.03	0.09 ± 0.02	0.02 ± 0.01	0.52 ± 0.38	0.00 ± 0.00	0.00 ± 0.00	0.04 ± 0.01	0.10 ± 0.17	0.15
IQL ($\tau = 0.99$)	0.06 ± 0.03	0.23 ± 0.05	0.04 ± 0.02	0.01 ± 0.01	0.07 ± 0.12	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.05
ReBRAC	0.87 ± 0.07	0.50 ± 0.28	0.02 ± 0.01	0.08 ± 0.09	0.91 ± 0.03	0.00 ± 0.00	0.92 ± 0.02	0.00 ± 0.00	0.69 ± 0.09	0.44
FQL	0.87 ± 0.12	0.22 ± 0.28	0.00 ± 0.00	0.18 ± 0.11	0.93 ± 0.02	0.28 ± 0.10	0.90 ± 0.06	0.00 ± 0.01	0.85 ± 0.04	0.47
DD-IQL ($\tau = 0.9$)	0.37 ± 0.03	0.63 ± 0.03	0.17 ± 0.00	0.03 ± 0.01	0.91 ± 0.00	0.01 ± 0.01	0.05 ± 0.01	0.09 ± 0.03	0.96 ± 0.03	0.36
DD-IQL ($\tau = 0.99$)	0.57 ± 0.04	0.63 ± 0.04	0.35 ± 0.01	0.09 ± 0.02	0.93 ± 0.03	0.00 ± 0.00	0.03 ± 0.01	0.07 ± 0.01	0.48 ± 0.18	0.35
DD-ReBRAC	0.83 ± 0.15	0.90 ± 0.05	0.51 ± 0.28	0.40 ± 0.03	0.93 ± 0.01	0.32 ± 0.06	0.94 ± 0.03	0.01 ± 0.01	0.83 ± 0.03	0.63

7 EXPERIMENTS

We evaluate DD-IQL on OGBench (Park et al., 2025a) environments. Our implementation follows the general code structure and hyperparameter conventions of CORL (Tarasov et al., 2022), with settings adapted from CORL and Flow Q-Learning (FQL) (Park et al., 2025b).

We compare DD-IQL with offline RL baselines, including IQL (Kostrikov et al., 2021), ReBRAC (Tarasov et al., 2023), and FQL (Park et al., 2025b). IQL and ReBRAC are Gaussian-policy baselines, while FQL is a flow-policy baseline. We also report DD-ReBRAC, which applies a lightweight analogue of the delayed-decoupled principle to ReBRAC, to examine whether the underlying design idea can also be useful in an offline RL algorithm with an explicit policy network. Details of DD-ReBRAC are provided in Appendix D.2.

Our main goal is to evaluate whether delayed-decoupled filtering can mitigate overestimation bias under challenging value-learning conditions. To this end, we construct bias-stress settings that make bias propagation more likely. We use the original hyperparameter settings from FQL (Park et al., 2025b) as our reference setting: the discount factor is $\gamma = 0.99$ or $\gamma = 0.995$ depending on the environment, and the expectile level for IQL-based algorithms is $\tau = 0.9$. We then make the value-learning setting more challenging by increasing the discount factor to $\gamma = 0.999$ and, for IQL-based algorithms, increasing the expectile level to $\tau = 0.99$.

A larger discount factor strengthens long-horizon bias propagation, while a larger expectile level places greater emphasis on the upper tail of the value distribution. As discussed in Section 4.2, these factors can make bootstrapped value learning more susceptible to accumulated estimation bias. Such conditions are also relevant to complex decision-making problems that require reasoning over long horizons. We therefore use these bias-stress settings to evaluate whether DD-IQL improves bias control under challenging offline RL conditions. All results are averaged over four training seeds, with each evaluation conducted over 50 episodes, and we report the mean and standard deviation across seeds.

7.1 MAIN RESULTS

Table 1 compares the performance of each algorithm across nine OGBench environments under the original and bias-stress discount settings. Overall, DD-IQL and DD-ReBRAC outperform their respective baselines, IQL and ReBRAC, in most environments under both settings.

The advantage of DD-IQL becomes more pronounced under the bias-stress setting. For standard IQL, increasing either the discount factor γ or the expectile level τ often leads to substantial performance degradation. This suggests that IQL becomes more vulnerable when long-horizon bootstrapping is amplified or when upper-tail filtering is strengthened. In contrast, DD-IQL remains stable and often improves as τ or γ increases.

Table 2: Filter–error coupling and filtering-signal diagnostics for AntMaze Large (AM-L) and Cube Single (Cube-S). All diagnostics are averaged over checkpoints from 5K to 100K training steps and then summarized across training seeds.

			Filter–TD $\lvert\text{corr.}\rvert$		Filter–adv. corr.		Mean weight	
Env.	γ	τ	IQL	DD-IQL	IQL	DD-IQL	IQL	DD-IQL
AM-L	0.99	0.9	0.190 ± 0.008	0.012 ± 0.003	0.548 ± 0.008	0.637 ± 0.016	0.262 ± 0.021	0.429 ± 0.005
	0.99	0.99	0.108 ± 0.012	0.024 ± 0.003	0.370 ± 0.031	0.628 ± 0.005	0.058 ± 0.011	0.304 ± 0.005
	0.999	0.9	0.159 ± 0.011	0.013 ± 0.002	0.578 ± 0.012	0.663 ± 0.008	0.258 ± 0.017	0.412 ± 0.007
	0.999	0.99	0.088 ± 0.008	0.018 ± 0.002	0.370 ± 0.021	0.604 ± 0.016	0.053 ± 0.009	0.295 ± 0.004
Cube-S	0.99	0.9	0.209 ± 0.013	0.039 ± 0.006	0.407 ± 0.008	0.487 ± 0.006	0.267 ± 0.014	0.385 ± 0.012
	0.99	0.99	0.149 ± 0.006	0.017 ± 0.002	0.261 ± 0.006	0.474 ± 0.008	0.053 ± 0.002	0.245 ± 0.008
	0.999	0.9	0.198 ± 0.015	0.040 ± 0.005	0.394 ± 0.009	0.467 ± 0.005	0.268 ± 0.014	0.385 ± 0.009
	0.999	0.99	0.146 ± 0.011	0.017 ± 0.002	0.251 ± 0.012	0.445 ± 0.008	0.053 ± 0.006	0.243 ± 0.002

These results indicate that larger discount factors and higher expectile levels are not inherently harmful. A larger γ can provide useful long-horizon learning signals, and a higher τ can encourage stronger policy improvement toward high-valued in-dataset actions. However, without sufficient bias control, these signals can be dominated by overestimation bias. DD-IQL mitigates this issue by reducing the dependence between filtering weights and estimation errors, allowing it to use stronger learning signals more reliably than standard IQL. Additional results on non-default task variants are provided in Appendix E.5.

7.2 Reduced Filter–Error Coupling while Preserving the Filtering Signal

To examine whether DD-IQL reduces undesirable filter–error dependence while retaining the intended value-dependent filtering behavior, we consider three complementary diagnostics. Let w_i denote the filtering weight applied to estimator i, let e_i^{TD} denote its TD residual, and let A_i^{filt} denote the advantage estimate used to construct w_i. For DD-IQL, $A_i^{\mathrm{filt}} = \widetilde{A}_{j_m(i)}$ is obtained from a different delayed estimator, whereas standard IQL constructs the filtering weight from the current advantage. We measure

$$C_{\mathrm{err}} = \left|\mathrm{Corr}\left(w_i, e_i^{\mathrm{TD}}\right)\right|, \qquad C_{\mathrm{sig}} = \mathrm{Corr}\left(w_i, A_i^{\mathrm{filt}}\right), \tag{19}$$

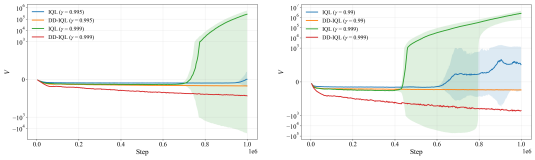
Here, C_{err} measures the dependence between the filtering decision and a Bellman-consistency proxy of the estimator receiving the weighted update. The TD residual is not an oracle measure of $Q - Q^\star$, but a smaller C_{err} indicates weaker coupling between the filter and this error proxy. In contrast, C_{sig} measures an intended dependence: the filtering weight should remain responsive to the advantage signal from which it is constructed. A substantial positive C_{sig} therefore indicates that the intended value-dependent filtering behavior remains active. We use absolute correlation for the error proxy because coupling strength is relevant regardless of sign, while retaining the sign for the advantage correlation because positive alignment represents the intended filtering behavior. We additionally report $\mathbb{E}[w_i]$ as a descriptive statistic of the filter's operating point.

As shown in Table 2, DD-IQL exhibits substantially smaller filter–TD correlation than IQL across all reported settings for both AM-L and Cube-S. At the same time, its filter–advantage correlation remains strongly positive and is not weakened relative to IQL in the reported settings. Thus, the weaker filter–error coupling is not accompanied by a loss of responsiveness to the advantage signal used for filtering. These results are consistent with the intended effect of delayed and decoupled filtering: DD-IQL weakens dependence between filtering decisions and the error proxy of the estimator being updated while retaining the value-dependent signal that drives adaptive filtering.

7.3 Value Stability under High-Discount Bias Stress

To further examine value stability under stronger bootstrapping, we track the learned value estimates of IQL and DD-IQL during training with $\tau = 0.9$ under both the original and high-discount settings. As shown in Fig. 1, standard IQL exhibits pronounced upward value drift, with the estimated value scale growing by several orders of magnitude. The instability becomes particularly severe under the high-discount setting, where longer-horizon bootstrapping allows estimation errors to propagate over a larger effective horizon. In contrast, DD-IQL maintains a substantially more stable value scale and avoids the explosive upward drift observed in IQL, even when $\gamma = 0.999$.

These results are consistent with the view that repeated bootstrapping can accumulate positive estimation errors when adaptive filtering remains coupled to noisy value estimates. By weakening this coupling through delayed and decoupled filtering, DD-IQL is designed to mitigate such accumulation, which is accompanied here by substantially improved value stability under bias-stress conditions.



(a) AS-A (b) Scene

Figure 1: Mean value estimates during training under the original discount setting and the high-discount bias-stress setting. Solid lines indicate the mean across training seeds, and shaded regions denote standard deviation.

7.4 Ablating Delayed and Decoupled Filtering

Finally, we ablate the two mechanisms underlying DD-IQL: delayed filtering and decoupled filtering. For a fair comparison with standard IQL, we use two critic estimators for all methods in this study.

We consider three variants. Delayed-Filtering IQL (DF-IQL) applies delayed filtering directly to the standard IQL architecture; when $D = 1$, it reduces to the standard IQL update. Self-Delayed IQL (SD-IQL) retains the estimator-wise update structure of DD-IQL but removes cross-estimator decoupling by setting $j_m(i) = i$, so that each estimator constructs its filtering weight from its own delayed copy. DD-IQL combines both mechanisms, using a delayed filtering signal from a different estimator, $j_m(i) \neq i$, for the update of estimator i.

Table 3: Ablation across delay periods D. The first two blocks use the original discount setting, and the last two use the severe-discount setting. Results report the mean $\pm$ standard deviation across training seeds after averaging performance over the nine OGBench environments within each seed.

Method	$D = 1$	$D = 10$	$D = 100$	$D = 1000$
Original discount factor γ, $\tau = 0.9$				
DF-IQL	0.19±0.02	0.20±0.01	0.17±0.00	0.16±0.01
SD-IQL	0.09±0.02	0.08±0.01	0.19±0.01	0.20±0.01
DD-IQL	0.16±0.01	0.16±0.01	0.16±0.00	0.15±0.01
Original discount factor γ, $\tau = 0.99$				
DF-IQL	0.04±0.00	0.03±0.00	0.12±0.03	0.17±0.00
SD-IQL	0.00±0.00	0.00±0.00	0.00±0.00	0.10±0.00
DD-IQL	0.24±0.01	0.24±0.02	0.25±0.01	0.24±0.00
High discount factor γ, $\tau = 0.9$				
DF-IQL	0.15±0.05	0.17±0.05	0.17±0.04	0.27±0.01
SD-IQL	0.09±0.01	0.09±0.01	0.10±0.04	0.19±0.00
DD-IQL	0.28±0.01	0.28±0.01	0.28±0.01	0.28±0.00
High discount factor γ, $\tau = 0.99$				
DF-IQL	0.05±0.01	0.03±0.00	0.08±0.05	0.17±0.03
SD-IQL	0.00±0.00	0.00±0.00	0.00±0.00	0.11±0.01
DD-IQL	0.30±0.01	0.31±0.02	0.29±0.01	0.30±0.01

Table 3 reports the average performance over the nine OGBench environments used in Table 1 across different delay periods D. Under the bias-stress settings, DF-IQL and SD-IQL generally benefit from larger delays, suggesting that temporal separation can improve performance when filtering remains coupled to the same estimator. In contrast, DD-IQL performs strongly and remains relatively stable across the tested delay periods, indicating that its performance is not highly sensitive to the choice of D. With sufficiently large D, all three variants outperform standard IQL under the bias-stress settings while using the same number of critic estimators. Overall, these results suggest that delayed and decoupled filtering improve performance under stronger bias settings.

8 Conclusion

In this paper, we studied filtering-induced bias in offline value learning. We first showed how estimation bias can enter Bellman targets and be repeatedly reused through bootstrapped updates. We then interpreted IQL as an adaptive filtered expectation over dataset actions and showed that, even without explicit action maximization, filtering can introduce bias when filtering decisions become coupled with value-estimation errors. To mitigate this dependence, we proposed delayed and decoupled filtering, which introduce temporal and estimator-level separation between filtering decisions and the value updates they affect. Applying these mechanisms to IQL yields DD-IQL. Empirically, DD-IQL exhibits weaker filter–error correlation and more stable value estimates, together with improved policy performance across a range of offline RL environments, with particularly pronounced gains under bias-stress settings. Overall, these results support delayed and decoupled filtering as a practical approach for mitigating filtering-induced bias in offline value learning.

AI USE STATEMENT

We used generative AI tools to assist with polishing the writing and improving the clarity of the manuscript, and for retrieval and discovery of potentially relevant prior work. We also used generative AI tools to help check mathematical claims and derivations that had already been developed by the authors. All AI-assisted edits and mathematical feedback were reviewed and verified by the authors, and any retrieved or suggested references were checked against the original sources before being included in the paper. The authors take full responsibility for the final content of the manuscript, including all mathematical claims, proofs, citations, and conclusions.

REFERENCES

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International conference on machine learning*, pp. 104–114. PMLR, 2020.

Chenjia Bai, Lingxiao Wang, Zhuoran Yang, Zhihong Deng, Animesh Garg, Peng Liu, and Zhaoran Wang. Pessimistic bootstrapping for uncertainty-driven offline reinforcement learning. *arXiv preprint arXiv:2202.11566*, 2022.

Tianyuan Chen, Ronglong Cai, Faguo Wu, and Xiao Zhang. Active: Offline reinforcement learning via adaptive imitation and in-sample v-ensemble. In *International Conference on Learning Representations*, volume 2025, pp. 77269–77287, 2025.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. Randomized ensembled double q-learning: Learning fast without a model. *arXiv preprint arXiv:2101.05982*, 2021.

Leif Döring, Benedikt Wille, Maximilian Birr, Mihail Bîrsan, and Martin Slowik. Addq: Adaptive distributional double q-learning. *arXiv preprint arXiv:2506.19478*, 2025.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6, 2005.

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

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019.

Divyansh Garg, Joey Hejna, Matthieu Geist, and Stefano Ermon. Extreme q-learning: Maxent rl without entropy. *arXiv preprint arXiv:2301.02328*, 2023.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

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

Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

Haobo Jiang, Guangyu Li, Jin Xie, and Jian Yang. Action candidate driven clipped double q-learning for discrete and continuous action tasks. *IEEE Transactions on Neural Networks and Learning Systems*, 35(4):5269–5279, 2022.

Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

Seungchan Kim, Kavosh Asadi, Michael Littman, and George Konidaris. Deepmellow: removing the need for a target network in deep q-learning. In *Proceedings of the twenty eighth international joint conference on artificial intelligence*, 2019.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in neural information processing systems*, 32, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry Vetrov. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In *International Conference on Machine Learning*, pp. 5556–5566. PMLR, 2020.

Qingfeng Lan, Yangchen Pan, Alona Fyshe, and Martha White. Maxmin q-learning: Controlling the estimation bias of q-learning. *arXiv preprint arXiv:2002.06487*, 2020.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Jiafei Lyu, Xiaoteng Ma, Jiangpeng Yan, and Xiu Li. Efficient continuous control with double actors and regularized critics. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 7655–7663, 2022.

Volodymyr Mnih. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Ling Pan, Qingpeng Cai, and Longbo Huang. Softmax deep double deterministic policy gradients. *Advances in neural information processing systems*, 33:11767–11777, 2020.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations*, volume 2025, pp. 94937–94982, 2025a.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48104–48127. PMLR, 13–19 Jul 2025b.

Oren Peer, Chen Tessler, Nadav Merlis, and Ron Meir. Ensemble bootstrapping for q-learning. In *International conference on machine learning*, pp. 8454–8463. PMLR, 2021.

Rafael Figueiredo Prudencio, Marcos ROA Maximo, and Esther Luna Colombini. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE transactions on neural networks and learning systems*, 35(8):10237–10257, 2023.

Zhizhou Ren, Guangxiang Zhu, Hao Hu, Beining Han, Jianglun Chen, and Chongjie Zhang. On the estimation bias in double q-learning. *Advances in Neural Information Processing Systems*, 34: 10246–10259, 2021.

Peter Schmitt-Förster and Tobias Sutter. Regularized q-learning through robust averaging. *arXiv preprint arXiv:2405.02201*, 2024.

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

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

Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

Zhao Song, Ron Parr, and Lawrence Carin. Revisiting the softmax bellman operator: New benefits and new perspective. In *International conference on machine learning*, pp. 5916–5925. PMLR, 2019.

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

Shengpu Tang and Jenna Wiens. Model selection for offline reinforcement learning: Practical considerations for healthcare settings. In *Machine Learning for Healthcare Conference*, pp. 2–35. PMLR, 2021.

Denis Tarasov, Alexander Nikulin, Dmitry Akimov, Vladislav Kurenkov, and Sergey Kolesnikov. CORL: Research-oriented deep offline reinforcement learning library. In *3rd Offline RL Workshop: Offline RL as a "Launchpad"*, 2022. URL https://openreview.net/forum?id=SyAS49bBcv.

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:11592–11620, 2023.

Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 connectionist models summer school*, pp. 255–263. Psychology Press, 2014.

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

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Yue Wu, Shuangfei Zhai, Nitish Srivastava, Joshua Susskind, Jian Zhang, Ruslan Salakhutdinov, and Hanlin Goh. Uncertainty weighted actor-critic for offline reinforcement learning. *arXiv preprint arXiv:2105.08140*, 2021.

Haoran Xu, Li Jiang, Jianxiong Li, Zhuoran Yang, Zhaoran Wang, Victor Wai Kin Chan, and Xianyuan Zhan. Offline rl with no ood actions: In-sample learning via implicit value regularization. *arXiv preprint arXiv:2303.15810*, 2023.

Xianyuan Zhan, Haoran Xu, Yue Zhang, Xiangyu Zhu, Honglei Yin, and Yu Zheng. Deepthermal: Combustion optimization for thermal power generating units using offline reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 4680–4688, 2022.

A Additional Analysis of Filtering-Induced Bias

This appendix provides additional analysis supporting the discussion in Section 5. Throughout this section, we focus on a fixed state s and write $q(a)$ for a noise-free reference action-value profile over actions $a \sim \mu(\cdot \mid s)$. We use $\widehat{q}(a) = q(a) + \epsilon(a)$ to denote a noisy value estimate, where $\mathbb{E}_\epsilon[\epsilon(a) \mid a] = 0$. The analysis below is intended to isolate simple mechanisms by which adaptive filtering can interact with estimation noise.

A.1 Exact Decomposition of Filtering-Induced Bias

Let $f : \mathbb{R} \to (0, \infty)$ be a filtering function and define the corresponding noise-free filtered value

$$F_f(q) := \frac{\mathbb{E}_{a\sim\mu}\left[f(q(a))q(a)\right]}{\mathbb{E}_{a\sim\mu}\left[f(q(a))\right]}. \tag{20}$$

For a realization of the estimation noise, define

$$w_\epsilon(a) := f(q(a) + \epsilon(a)), \qquad Z_\epsilon := \mathbb{E}_{a\sim\mu}\left[w_\epsilon(a)\right], \tag{21}$$

and

$$S_\epsilon := \mathbb{E}_{a\sim\mu}\left[w_\epsilon(a)q(a)\right], \qquad R_\epsilon := \mathbb{E}_{a\sim\mu}\left[w_\epsilon(a)\epsilon(a)\right]. \tag{22}$$

Assuming the relevant expectations are finite, the filtered value constructed from the noisy estimates can be written exactly as

$$\widehat{F}_f = \frac{S_\epsilon}{Z_\epsilon} + \frac{R_\epsilon}{Z_\epsilon}. \tag{23}$$

Let

$$Z_0 := \mathbb{E}_{a\sim\mu}\left[f(q(a))\right], \qquad S_0 := \mathbb{E}_{a\sim\mu}\left[f(q(a))q(a)\right], \tag{24}$$

so that $F_f(q) = S_0/Z_0$. The expected noise-induced deviation from the noise-free filtered value therefore admits the exact decomposition

$$\begin{aligned} B_f^{\mathrm{noise}} &:= \mathbb{E}_\epsilon\left[\widehat{F}_f\right] - F_f(q) \\ &= \underbrace{\mathbb{E}_\epsilon\left[\frac{S_\epsilon}{Z_\epsilon}\right] - \frac{S_0}{Z_0}}_{\text{noise-induced change in value-based reweighting}} + \underbrace{\mathbb{E}_\epsilon\left[\frac{R_\epsilon}{Z_\epsilon}\right]}_{\text{direct estimation-error contribution}}. \end{aligned} \tag{25}$$

Here, B_f^{noise} measures deviation relative to the same filtering operator applied to the noise-free reference values; it should not be interpreted as bias relative to the optimal value function.

The second term contains the direct interaction between the filtering weight and estimation error. In particular, for each fixed action,

$$\mathbb{E}_\epsilon\left[f(q(a) + \epsilon(a))\epsilon(a) \mid a\right] = \mathrm{Cov}_\epsilon\left(f(q(a) + \epsilon(a)), \epsilon(a) \mid a\right), \tag{26}$$

where we used $\mathbb{E}_\epsilon[\epsilon(a) \mid a] = 0$. This covariance isolates a direct filter–error coupling. Equation equation 25 also makes clear, however, that the total noise-induced deviation need not be determined by this covariance alone, because estimation noise can additionally change the relative action weights and the random normalization.

A.2 A Stylized Upper-Expectile Example

The expectile weighting used by IQL is threshold-based rather than a smooth filtering function. We therefore give a separate stylized example showing that upper-expectile weighting can produce an upward shift even when the underlying variation is symmetric and zero-mean.

For a random variable X with finite second moment, let $e_\tau(X)$ denote its τ-expectile:

$$e_\tau(X) := \arg\min_v \mathbb{E}\left[|\tau - \mathbf{1}\{X - v < 0\}|\,(X - v)^2\right]. \tag{27}$$

When $\tau = 1/2$, the expectile coincides with the mean, while $\tau > 1/2$ assigns greater weight to positive residuals.

Proposition 1 (A positive upper-expectile shift)**.** *Let*

$$X = q_0 + E, \tag{28}$$

where E has finite second moment, is symmetric about zero, and satisfies $\mathbb{P}(E \neq 0) > 0$. If $\tau > 1/2$, then

$$e_\tau(X) > q_0. \tag{29}$$

Proof. By location equivariance of expectiles,

$$e_\tau(q_0 + E) = q_0 + e_\tau(E), \tag{30}$$

so it suffices to show that $e_\tau(E) > 0$. The first-order condition for the expectile can be written as

$$g(v) := \tau\mathbb{E}\left[(E - v)_+\right] - (1 - \tau)\mathbb{E}\left[(v - E)_+\right] = 0. \tag{31}$$

By symmetry,

$$\mathbb{E}[E_+] = \mathbb{E}[(-E)_+], \tag{32}$$

and hence

$$g(0) = \tau\mathbb{E}[E_+] - (1 - \tau)\mathbb{E}[(-E)_+] = (2\tau - 1)\mathbb{E}[E_+] > 0. \tag{33}$$

The expectile objective is convex and has a unique minimizer under the stated conditions, while $g(v)$ decreases through its zero. Therefore, the solution satisfies $e_\tau(E) > 0$, which implies $e_\tau(X) > q_0$. □

Proposition 1 is intended only as a mechanism-isolating example. It shows that an upper expectile of a symmetric zero-mean distribution can lie above its mean even when there is no underlying positive shift. In other words, asymmetric expectile weighting can convert symmetric zero-mean variation into an upward shift in the filtered value. This provides a simple example of how upper-tail filtering can introduce directional value distortion even in the absence of systematic positive estimation error.

B Additional Analysis of Maximization Bias in Continuous Actions

The main text uses the finite-action approximation $\eta_{\mathcal{A}}(s) \approx \sigma\sqrt{2\log|\mathcal{A}|}$ only as an intuition for the dependence of maximization bias on the number of noisy alternatives. Here we state the conditions under which this approximation is appropriate and clarify its interpretation in continuous action spaces.

For a fixed state, let

$$\widehat{q}_i = q_i + \sigma Z_i, \qquad i = 1, \ldots, M, \tag{34}$$

where q_i is the mean value of action i, $\sigma > 0$ is a common error scale, and $\mathbb{E}[Z_i] = 0$. Let

$$q^\star := \max_i q_i, \qquad \Delta_i := q^\star - q_i. \tag{35}$$

The one-step maximization bias can be written exactly as

$$\eta_{\mathcal{A}}(s) := \mathbb{E}\left[\max_i \left(q_i + \sigma Z_i\right)\right] - q^\star = \sigma\mathbb{E}\left[\max_i \left(Z_i - \frac{\Delta_i}{\sigma}\right)\right]. \tag{36}$$

Thus, actions with gaps large relative to the noise scale contribute less to the maximization effect. When the relevant action values are approximately equal and the Z_i are independent standard Gaussian variables,

$$\eta_{\mathcal{A}}(s) \approx \sigma\mathbb{E}\left[\max_{1\le i\le M} Z_i\right], \tag{37}$$

and the standard Gaussian extreme-value asymptotic gives

$$\mathbb{E}\left[\max_{1\le i\le M} Z_i\right] \sim \sqrt{2\log M} \qquad \text{as } M \to \infty. \tag{38}$$

The approximation $\eta_{\mathcal{A}}(s) \approx \sigma\sqrt{2\log|\mathcal{A}|}$ should therefore be understood as a scaling intuition under approximately equal action values and approximately independent Gaussian errors, rather than as a general characterization.

For continuous actions, let

$$q^\star(s) := \sup_{a\in\mathcal{A}} \overline{Q}_k(s,a), \qquad \Delta(s,a) := q^\star(s) - \overline{Q}_k(s,a). \tag{39}$$

Whenever the relevant supremum and expectations are well defined, the continuous-action analogue is

$$\eta_k(s) := \mathbb{E}_\epsilon\left[\sup_{a\in\mathcal{A}}\left(\overline{Q}_k(s,a)+\epsilon_k(s,a)\right)\right] - q^\star(s) = \mathbb{E}_\epsilon\left[\sup_{a\in\mathcal{A}}\left(\epsilon_k(s,a)-\Delta(s,a)\right)\right]. \tag{40}$$

Equation 40 shows that the relevant quantity in continuous control is not the infinite cardinality of the action space itself. Instead, the maximization effect depends on the action-value gaps together with the magnitude and dependence structure of the estimation-error process over actions.

C Additional Interpretation of DD-IQL

This section provides simple population-level interpretations of the two mechanisms used by DD-IQL. These arguments are intended to clarify how decoupling and delay modify filter–error dependence.

C.1 Signal–Error Decomposition and Decoupled Filtering

To isolate the direct error contribution, consider a source estimator that constructs a filtering weight and a receiver estimator whose value is weighted. Let

$$\widehat{Q}_{\mathrm{src}}(s,a) = Q(s,a) + \epsilon_{\mathrm{src}}(s,a), \qquad \widehat{Q}_{\mathrm{rec}}(s,a) = Q(s,a) + \epsilon_{\mathrm{rec}}(s,a). \tag{41}$$

Let the source estimator construct a positive filtering weight

$$w_{\mathrm{src}}(s,a) = g\left(\widehat{Q}_{\mathrm{src}}(s,a), \widehat{V}_{\mathrm{src}}(s)\right), \tag{42}$$

where g denotes the corresponding weighting rule. The population weighted-value analogue for the receiver is

$$\widehat{V}_{\mathrm{src}\to\mathrm{rec}}(s) := \frac{\mathbb{E}_{a\sim\mu(\cdot|s)}\left[w_{\mathrm{src}}(s,a)\widehat{Q}_{\mathrm{rec}}(s,a)\right]}{\mathbb{E}_{a\sim\mu(\cdot|s)}\left[w_{\mathrm{src}}(s,a)\right]}. \tag{43}$$

Substituting $\widehat{Q}_{\mathrm{rec}} = Q + \epsilon_{\mathrm{rec}}$ yields

$$\widehat{V}_{\mathrm{src}\to\mathrm{rec}}(s) = \underbrace{\frac{\mathbb{E}_{a\sim\mu}\left[w_{\mathrm{src}}(s,a)Q(s,a)\right]}{\mathbb{E}_{a\sim\mu}\left[w_{\mathrm{src}}(s,a)\right]}}_{\text{selection-weighted value component}} + \underbrace{\frac{\mathbb{E}_{a\sim\mu}\left[w_{\mathrm{src}}(s,a)\epsilon_{\mathrm{rec}}(s,a)\right]}{\mathbb{E}_{a\sim\mu}\left[w_{\mathrm{src}}(s,a)\right]}}_{\text{direct receiver-error contribution}}. \tag{44}$$

A simple sufficient condition for the direct receiver-error term to vanish is conditional unbiasedness. Suppose that, for every action,

$$\mathbb{E}\left[\epsilon_{\mathrm{rec}}(s,a) \mid Q, \widehat{Q}_{\mathrm{src}}, \widehat{V}_{\mathrm{src}}\right] = 0. \tag{45}$$

Conditioned on Q, $\widehat{Q}_{\mathrm{src}}$, and $\widehat{V}_{\mathrm{src}}$, both the filtering weights and their normalization are fixed. Therefore,

$$\mathbb{E}\left[\left.\frac{\mathbb{E}_{a\sim\mu}\left[w_{\mathrm{src}}(s,a)\epsilon_{\mathrm{rec}}(s,a)\right]}{\mathbb{E}_{a\sim\mu}\left[w_{\mathrm{src}}(s,a)\right]}\right| Q, \widehat{Q}_{\mathrm{src}}, \widehat{V}_{\mathrm{src}}\right] = 0. \tag{46}$$

Thus, under the conditional-unbiasedness assumption, the direct receiver-error contribution vanishes in expectation. In practice, however, neural-network estimators trained on the same offline dataset and related bootstrapped targets can have correlated errors, so equation 45 need not hold exactly. Decoupled filtering therefore does not imply statistical independence between the source and receiver estimators. Rather, it is designed so that the receiver estimator is not itself used to construct its filtering weight, thereby avoiding direct same-estimator reuse and weakening filter–receiver error dependence.

C.2 Covariance View of Delayed Filtering

We give a simple interpretation of delayed filtering. Let $\widetilde{w}_m(s,a)$ denote a filtering weight constructed at the beginning of delay interval m and held fixed for the following D optimization steps. Let $\epsilon_{m,t}(s,a)$ denote a zero-mean current estimation error at step $t \in \{1,\dots,D\}$ within that interval. For a fixed state-action pair,

$$\mathbb{E}\left[\frac{1}{D}\sum_{t=1}^{D}\widetilde{w}_m(s,a)\epsilon_{m,t}(s,a)\right] = \frac{1}{D}\sum_{t=1}^{D}\mathrm{Cov}\left(\widetilde{w}_m(s,a),\epsilon_{m,t}(s,a)\right), \tag{47}$$

where we used $\mathbb{E}[\epsilon_{m,t}(s,a)] = 0$.

If the filtering weight is instead reconstructed from the current estimator at every step, the analogous direct coupling involves

$$\frac{1}{D}\sum_{t=1}^{D}\mathrm{Cov}\left(w_{m,t}(s,a),\epsilon_{m,t}(s,a)\right), \tag{48}$$

where $w_{m,t}$ is the current filtering weight. Holding $\widetilde{w}_m$ fixed avoids reconstructing the filtering weight from the instantaneous noisy estimate at every optimization step. If the stale filtering weight is less strongly coupled with subsequent estimation errors than a newly reconstructed current weight, the average direct weighted-error contribution over the interval can be reduced. However, when substantial temporal correlation persists, such a reduction is not guaranteed.

This covariance view isolates the direct weighted-error component of filtering-induced bias. At the same time, the filtering signal may become stale as the value estimates evolve. Consequently, the delay period introduces a trade-off between temporal separation from instantaneous estimation fluctuations and the relevance of the filtering signal to the current value estimates. Overall, delayed filtering is best interpreted as a temporal separation mechanism rather than as a guarantee of statistical decorrelation. Together with estimator-level source–receiver separation, it is designed to reduce direct reuse of estimation errors in adaptive filtering.

D Algorithm Details

This appendix provides implementation details for DD-IQL and DD-ReBRAC. We also clarify the ablation variants used in Section 7.4 and discuss the relationship between the proposed delayed-decoupled mechanism and Double Q-learning.

D.1 DD-IQL

Algorithm 1 summarizes the full training procedure of DD-IQL. The key difference from standard IQL is that each value estimator is updated using a filtering weight computed from a delayed and, when $N > 1$, different estimator.

In our implementation, the decoupling assignment $j_m(i)$ is updated cyclically every delay interval. Let m denote the delay-interval index and let r_m denote the cyclic offset. For $N > 1$, we set

$$r_m = 1 + (m \bmod (N-1)), \qquad j_m(i) = 1 + ((i - 1 + r_m) \bmod N). \tag{49}$$

This assignment ensures that $j_m(i) \neq i$ for every estimator i whenever $N > 1$. When $N = 2$, each estimator always uses the other estimator to construct its filtering weight. When $N > 2$, the filtering partner cycles over the other estimators as m increases. If $N = 1$, decoupling is not possible and we set $j_m(1) = 1$.

The target critic of each estimator is updated by a soft target update:

$$\bar{\theta}_i \leftarrow (1-\rho)\bar{\theta}_i + \rho\theta_i, \qquad i = 1,\dots,N, \tag{50}$$

where $\rho \in (0,1]$ is the target update rate.

Algorithm 1 Delayed Decoupled IQL (DD-IQL)

Require: Dataset $\mathcal{D}$, ensemble size N, delay period D, expectile level τ, discount factor γ, actor temperature β, target update rate ρ

Initialize critics $\{Q_{\theta_i}\}_{i=1}^{N}$, target critics $\{Q_{\bar{\theta}_i}\}_{i=1}^{N}$, value functions $\{V_{\psi_i}\}_{i=1}^{N}$, and policy π_ϕ
Initialize delayed copies $\{(\widetilde{Q}_i, \widetilde{V}_i)\}_{i=1}^{N}$ from $\{(Q_{\bar{\theta}_i}, V_{\psi_i})\}_{i=1}^{N}$
Initialize delay-interval index $m = 0$ and assignment j_m using equation 49
for each training step $t = 1, 2, \ldots$ **do**
  Sample a minibatch $(s, a, r, s') \sim \mathcal{D}$
  for each estimator $i = 1, \ldots, N$ **do**
    Compute the delayed decoupled filtering weight $w_i(s, a)$ using equation 14
    Update V_{ψ_i} by minimizing the value loss in equation 15
    Update Q_{θ_i} by minimizing the critic loss in equation 16
    Update the target critic $Q_{\bar{\theta}_i}$ using equation 50
  end for
  Compute the ensemble actor advantage using equation 17
  Update the policy π_ϕ by minimizing the actor loss in equation 18
  if $t \bmod D = 0$ **then**
    Refresh delayed copies: $\widetilde{Q}_i \leftarrow Q_{\bar{\theta}_i}$ and $\widetilde{V}_i \leftarrow V_{\psi_i}$ for all i
    Update $m \leftarrow m + 1$ and recompute j_m using equation 49
  end if
end for

D.2 DD-ReBRAC: A Policy-Network Instantiation

We additionally consider a lightweight policy-network instantiation of the delayed-decoupled principle based on ReBRAC (Tarasov et al., 2023). Unlike DD-IQL, which explicitly separates the estimator constructing the filtering weight from the estimator receiving the weighted value update, DD-ReBRAC does not implement the same one-to-one source–receiver separation. Instead, it introduces temporal separation in target-action construction through a hard-delayed actor and reduces reliance on any single critic through ensemble-average evaluation.

DD-ReBRAC maintains an online actor π_ϕ, a hard-delayed actor $\widetilde{\pi}$, and an ensemble of N_Q critics $\{Q_{\theta_i}\}_{i=1}^{N_Q}$ with corresponding target critics $\{Q_{\bar{\theta}_i}\}_{i=1}^{N_Q}$. The critics use standard soft target updates, whereas the delayed actor is refreshed by a hard copy from the online actor every D optimization steps:

$$\widetilde{\pi} \leftarrow \pi_\phi \qquad \text{every } D \text{ update steps.} \tag{51}$$

Between two refreshes, $\widetilde{\pi}$ remains fixed. Thus, the action used in the Bellman target is temporally separated from instantaneous changes in the online actor.

Compared with original ReBRAC, DD-ReBRAC modifies two components. First, the next action used in the critic target is constructed from the hard-delayed actor rather than the actor currently being optimized. Second, the target value is evaluated using the mean of the target-critic ensemble rather than a clipped minimum over two critics. Consequently, even when $N_Q = 2$ and $D = 1$, DD-ReBRAC does not exactly recover original ReBRAC, because the critic aggregation remains an ensemble mean rather than minimum.

Given a transition $(s, a, r, s', a'_{\mathcal{D}}) \sim \mathcal{D}$, where $a'_{\mathcal{D}}$ denotes the next dataset action, the target action is constructed as

$$\widetilde{a}' = \mathrm{clip}\left(\widetilde{\pi}(s') + \mathrm{clip}(\epsilon, -c, c), -1, 1\right), \qquad \epsilon \sim \mathcal{N}(0, \sigma_\pi^2 I), \tag{52}$$

where σ_π is the target policy smoothing noise scale and c is the noise clipping threshold.

The selected target action is evaluated using the ensemble mean of the target critics:

$$\overline{Q}_{\bar{\theta}}(s', \widetilde{a}') = \frac{1}{N_Q} \sum_{i=1}^{N_Q} Q_{\bar{\theta}_i}(s', \widetilde{a}'). \tag{53}$$

Unlike clipped double-Q targets, this aggregation does not introduce an explicit minimum across critics. The resulting critic target is

$$y_{\text{DD-ReBRAC}} = r + \gamma \left(\overline{Q}_{\bar{\theta}}(s', \widetilde{a}') - \alpha_{\text{critic}} \left\| \widetilde{a}' - a'_{\mathcal{D}} \right\|_2^2 \right), \tag{54}$$

where α_{critic} controls the critic-side behavior-regularization penalty. The critic ensemble is optimized using

$$\mathcal{L}_Q^{\text{DD-ReBRAC}} = \mathbb{E}_{(s,a,r,s',a'_{\mathcal{D}}) \sim \mathcal{D}} \left[\frac{1}{N_Q} \sum_{i=1}^{N_Q} \left(Q_{\theta_i}(s,a) - y_{\text{DD-ReBRAC}} \right)^2 \right]. \tag{55}$$

The target critics are updated by

$$\bar{\theta}_i \leftarrow (1-\rho)\bar{\theta}_i + \rho\theta_i, \qquad i = 1, \ldots, N_Q, \tag{56}$$

where $\rho \in (0, 1]$ is the target update rate. Thus, the target-action policy is updated through periodic hard copies, while the target critics follow the online critics through soft updates.

The online actor is trained using the ensemble-mean critic while remaining close to the dataset action. Define

$$a_\phi(s) = \pi_\phi(s), \qquad \overline{Q}_\theta(s, a_\phi(s)) = \frac{1}{N_Q} \sum_{i=1}^{N_Q} Q_{\theta_i}(s, a_\phi(s)). \tag{57}$$

The actor loss is

$$\mathcal{L}_\pi^{\text{DD-ReBRAC}} = \mathbb{E}_{(s,a) \sim \mathcal{D}} \left[-\lambda \overline{Q}_\theta(s, a_\phi(s)) + \alpha_{\text{actor}} \left\| a_\phi(s) - a \right\|_2^2 \right], \tag{58}$$

where α_{actor} controls the actor-side behavior regularization. Following the implementation used in our experiments, λ can be normalized by the critic scale:

$$\lambda = \text{stopgrad} \left(\frac{1}{\left| \mathbb{E}_{(s,a) \sim \mathcal{D}} \left[\overline{Q}_\theta(s, a_\phi(s)) \right] \right|} \right). \tag{59}$$

Overall, DD-ReBRAC should be viewed as a lightweight policy-network analogue of the proposed bias-mitigation principle rather than as a direct counterpart to DD-IQL, allowing us to examine whether the underlying design principle extends beyond expectile-based value learning.

D.3 Definitions of the Ablation Variants

Section 7.4 compares three variants—Delayed-Filtering IQL (DF-IQL), Self-Delayed IQL (SD-IQL), and DD-IQL—to distinguish the effects of delayed filtering and cross-estimator decoupling. In the ablation study, all methods use the same budget of two critic estimators. However, DF-IQL retains the standard IQL clipped-double-critic structure, whereas SD-IQL and DD-IQL use the estimator-wise update structure of DD-IQL. We describe these differences below.

IQL and Delayed-Filtering IQL (DF-IQL). Standard IQL commonly maintains two critics and uses their clipped minimum when fitting the expectile value function. Define

$$Q_{\bar{\theta}}^{\min}(s,a) := \min_{\ell \in \{1,2\}} Q_{\bar{\theta}_\ell}(s,a). \tag{60}$$

For standard IQL, the expectile advantage and weight are computed from the current clipped-min target critic and value function:

$$A^{\text{IQL}}(s,a) = Q_{\bar{\theta}}^{\min}(s,a) - V_\psi(s), \qquad w^{\text{IQL}}(s,a) = \left| \tau - \mathbf{1}\left\{ A^{\text{IQL}}(s,a) < 0 \right\} \right|. \tag{61}$$

DF-IQL applies delayed filtering directly to this standard IQL structure. Let

$$\widetilde{Q}^{\min}(s,a) := \min_{\ell \in \{1,2\}} \widetilde{Q}_\ell(s,a), \tag{62}$$

Algorithm 2 Delayed Decoupled ReBRAC (DD-ReBRAC)

Require: Dataset $\mathcal{D}$, number of critics N_Q, delay period D, discount factor γ, actor update frequency K_π, actor behavior coefficient α_{actor}, critic behavior coefficient α_{critic}, target update rate ρ, target policy smoothing parameters (σ_π, c)

Initialize critics $\{Q_{\theta_i}\}_{i=1}^{N_Q}$, target critics $\{Q_{\bar{\theta}_i}\}_{i=1}^{N_Q}$, actor π_ϕ, and delayed actor $\widetilde{\pi}$
Initialize target critics by $\bar{\theta}_i \leftarrow \theta_i$ for all i
Initialize delayed actor by $\widetilde{\pi} \leftarrow \pi_\phi$
for each training step $t = 1, 2, \ldots$ **do**
  Sample a minibatch $(s, a, r, s', a'_{\mathcal{D}}) \sim \mathcal{D}$
  Construct the delayed-policy target action $\widetilde{a}'$ using equation 52
  Compute the ensemble-mean target value using equation 53
  Construct the critic target $y_{\text{DD-ReBRAC}}$ using equation 54
  Update critics $\{Q_{\theta_i}\}_{i=1}^{N_Q}$ by minimizing equation 55
  if $t \bmod K_\pi = 0$ **then**
    Update the actor π_ϕ by minimizing equation 58
    Soft-update target critics using equation 56
  end if
  if $t \bmod D = 0$ **then**
    Hard-update the delayed actor using equation 51
  end if
end for

where $\widetilde{Q}_\ell$ and $\widetilde{V}$ denote delayed copies of the critic and value networks. DF-IQL constructs the filtering weight from the delayed advantage

$$\widetilde{A}^{\text{DF-IQL}}(s,a) = \widetilde{Q}^{\min}(s,a) - \widetilde{V}(s), \qquad \widetilde{w}^{\text{DF-IQL}}(s,a) = \left|\tau - \mathbf{1}\left\{\widetilde{A}^{\text{DF-IQL}}(s,a) < 0\right\}\right|. \quad (63)$$

The value regression target itself remains the current clipped-min target critic. Thus,

$$\mathcal{L}_V^{\text{IQL}} = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[w^{\text{IQL}}(s,a)\left(Q_{\bar{\theta}}^{\min}(s,a) - V_\psi(s)\right)^2\right],$$
$$\mathcal{L}_V^{\text{DF-IQL}} = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\widetilde{w}^{\text{DF-IQL}}(s,a)\left(Q_{\bar{\theta}}^{\min}(s,a) - V_\psi(s)\right)^2\right]. \quad (64)$$

The critic target has the same form in both methods:

$$y^{\text{IQL}}(s,a,r,s') = y^{\text{DF-IQL}}(s,a,r,s') = r + \gamma V_\psi(s'), \quad (65)$$

and the two critics are fitted by

$$\mathcal{L}_Q = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\frac{1}{2}\sum_{\ell=1}^{2}\left(Q_{\theta_\ell}(s,a) - y(s,a,r,s')\right)^2\right]. \quad (66)$$

Under the refresh convention used in our implementation, when $D = 1$ the delayed copies are refreshed at every update step and coincide with the corresponding current estimates when the filtering weight is constructed. Therefore, DF-IQL reduces to the standard IQL update at $D = 1$.

Self-Delayed IQL (SD-IQL). SD-IQL is an ablation of DD-IQL that retains delayed filtering but removes cross-estimator decoupling. Unlike DF-IQL, SD-IQL uses the estimator-wise structure of DD-IQL. For $N = 2$, each estimator $i \in \{1, 2\}$ maintains

$$\left(Q_{\theta_i}, Q_{\bar{\theta}_i}, V_{\psi_i}, \widetilde{Q}_i, \widetilde{V}_i\right). \quad (67)$$

SD-IQL uses self-delayed filtering by setting

$$j_m(i) = i. \quad (68)$$

Thus, estimator i constructs its filtering weight from its own delayed copy:

$$\widetilde{A}_i^{\text{SD-IQL}}(s,a) = \widetilde{Q}_i(s,a) - \widetilde{V}_i(s), \qquad w_i^{\text{SD-IQL}}(s,a) = \left|\tau - \mathbf{1}\left\{\widetilde{A}_i^{\text{SD-IQL}}(s,a) < 0\right\}\right|. \quad (69)$$

The corresponding value loss is

$$\mathcal{L}_{V_i}^{\text{SD-IQL}} = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[w_i^{\text{SD-IQL}}(s,a)\left(Q_{\bar{\theta}_i}(s,a) - V_{\psi_i}(s)\right)^2\right]. \tag{70}$$

The estimator-specific critic target is

$$y_i^{\text{SD-IQL}}(s,a,r,s') = r + \gamma V_{\psi_i}(s'), \tag{71}$$

and the critic is fitted by

$$\mathcal{L}_{Q_i}^{\text{SD-IQL}} = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(Q_{\theta_i}(s,a) - y_i^{\text{SD-IQL}}(s,a,r,s')\right)^2\right]. \tag{72}$$

Hence, SD-IQL retains temporal separation through delayed filtering, but the source and receiver indices are the same. It therefore removes the cross-estimator decoupling used by DD-IQL.

Delayed Decoupled IQL (DD-IQL). DD-IQL uses the same estimator-wise value and critic structure as SD-IQL, but constructs the filtering weight for estimator i from a different delayed source estimator, $j_m(i) \neq i$, as described in Section 6.2. Thus, SD-IQL isolates delayed filtering by setting $j_m(i) = i$, whereas DD-IQL combines delayed filtering with cross-estimator decoupling. DF-IQL, by contrast, applies delayed filtering directly to the standard clipped-double-critic IQL architecture.

D.4 Relation to Double Q-learning and Clipped Double Q-learning

Double Q-learning (Hasselt, 2010) and clipped double Q-learning (Fujimoto et al., 2018) are two widely used approaches for mitigating overestimation in value-based reinforcement learning. Although DD-IQL shares with these methods the broad goal of reducing bias arising from noisy value estimates, the source of bias being addressed and the location at which the bias-control mechanism is applied are different.

Double Q-learning. Double Q-learning addresses maximization bias by separating action selection from action evaluation. In its classical form, one estimator selects the greedy action, while another estimator evaluates the selected action. For example, an update of Q^A can use the target

$$y^A = r + \gamma Q^B\left(s', \arg\max_{a'} Q^A(s',a')\right), \tag{73}$$

with an analogous update obtained by exchanging the roles of Q^A and Q^B. The cross-estimator interaction therefore occurs directly in the Bellman target: one estimator determines which action is selected, and the other provides the value assigned to that selected action. This avoids using the same noisy estimator both to select the maximizing action and to evaluate it.

Clipped double Q-learning. Clipped double Q-learning uses a different strategy. Rather than separating action selection and evaluation, it constructs a conservative target using the minimum of multiple critic estimates. Schematically,

$$y_{\text{clip}} = r + \gamma \min_{\ell\in\{1,2\}} Q_{\bar{\theta}_\ell}(s',a'), \tag{74}$$

where a' denotes the action used in the target backup. Taking the minimum can reduce positive overestimation when one critic is spuriously high, and this mechanism is widely used in modern actor–critic methods (Fujimoto et al., 2018). At the same time, the minimum operator is conservative and can produce underestimation when one critic is substantially lower than the other. Thus, clipped double Q-learning controls overestimation through pessimistic aggregation.

DD-IQL. DD-IQL targets filtering-induced bias by separating the construction of an adaptive filtering weight from the estimator receiving the corresponding weighted update. Under the ensemble-based formulation, the filtering weight for estimator i is constructed from a different estimator $j_m(i) \neq i$, while estimator i retains its own value-regression residual, as defined in equation 14 and equation 15. The estimator assignment is periodically cycled across delay intervals, avoiding persistent pairing between the same estimators and potentially weakening estimator-specific correlations over training.

DD-IQL additionally constructs the filtering weight from delayed estimates. This introduces temporal separation between the filtering decision and the current value update, complementing the estimator-level separation between the estimator constructing the filtering weight and the estimator being updated. The delayed and decoupled components are therefore designed to weaken the dependence between filtering decisions and value-estimation errors across optimization steps and across estimators, respectively.

DD-IQL also differs from clipped double Q-learning in how multiple critics are used. Clipped double Q-learning controls overestimation through an explicit minimum across critic estimates, providing conservative target evaluation but potentially introducing underestimation when one critic is spuriously low. DD-IQL does not use an explicit minimum across its estimator-specific value updates as its primary bias-control mechanism; instead, each estimator retains its own value and critic updates as in equation 15 and equation 16. When the estimators are aggregated for policy extraction, DD-IQL uses their ensemble mean as in equation 17, rather than a clipped minimum. This avoids introducing additional pessimism explicitly through a minimum operator, although it does not imply that DD-IQL is free from estimation bias or underestimation.

E Additional experiments

This section provides additional experimental results that complement the main-text evaluation. We first summarize the environment and task abbreviations used throughout the result tables. We then provide complementary diagnostics for the temporal and estimator-level effects of delayed and decoupled filtering, followed by the full per-environment ablation results and additional evaluations on non-default OGBench tasks.

E.1 Environment and Task Notation

To keep the result tables compact, we use abbreviated names for the OGBench environments throughout the paper and appendix. Table 4 summarizes the abbreviations together with the corresponding environment names and default single-task settings.

Unless a task suffix is explicitly specified, an abbreviation refers to the default single-task setting of the corresponding OGBench environment. For results on non-default tasks, we append Tn to the environment abbreviation, where n denotes the task index. For example, HM-L denotes the default task of HumanoidMaze-Large, whereas HM-L-T2 denotes task 2 of the same environment.

Table 4: Abbreviations for the OGBench environments used in our experiments. An abbreviation without a task suffix denotes the default single-task setting of OGBench. Non-default tasks are indicated by appending Tn, where n is the task index (e.g., T2 denotes task 2 of HumanoidMaze-Large).

Abbreviation	OGBench environment	Default task	Task-specific notation
AM-L	AntMaze-Large	T1	AM-L-Tn
HM-M	HumanoidMaze-Medium	T1	HM-M-Tn
HM-L	HumanoidMaze-Large	T1	HM-L-Tn
AS-A	AntSoccer-Arena	T4	AS-A-Tn
Cube-S	Cube-Single	T2	Cube-S-Tn
Cube-D	Cube-Double	T2	Cube-D-Tn
Pz-3x3	Puzzle-3x3	T4	Pz-3x3-Tn
Pz-4x4	Puzzle-4x4	T4	Pz-4x4-Tn
Scene	Scene	T2	Scene-Tn

E.2 Temporal Effect of Delayed Filtering

Delayed filtering is intended to prevent filtering decisions from immediately tracking changes in the estimators being updated. To characterize this temporal separation during training, we measure how the filtering weight applied to estimator i is associated with the drift of that estimator's current target critic from its own delayed snapshot. Specifically, we define

$$\Delta_i^{\mathrm{temp}}(s,a) = Q_{\bar{\theta}_i}(s,a) - \widetilde{Q}_i(s,a), \qquad C_{\mathrm{temp}} = \mathrm{Corr}\big(w_i, \Delta_i^{\mathrm{temp}}\big). \tag{75}$$

Table 5: Signed correlation between the filtering weight and critic temporal drift across a delay interval for DD-IQL with $D = 100$ on AntMaze Large (AM-L) and Cube Single (Cube-S). Delay age denotes the number of optimization steps since the most recent delayed-network refresh. Diagnostics are computed over training steps 5K–100K, averaged within each seed, and then summarized across training seeds.

Env.	γ	τ	Age 25	Age 50	Age 75
AM-L	0.99	0.9	-0.032 ± 0.003	-0.029 ± 0.004	-0.030 ± 0.004
	0.99	0.99	-0.049 ± 0.005	-0.062 ± 0.004	-0.069 ± 0.003
	0.999	0.9	-0.041 ± 0.006	-0.041 ± 0.008	-0.043 ± 0.008
	0.999	0.99	-0.077 ± 0.004	-0.091 ± 0.005	-0.099 ± 0.005
Cube-S	0.99	0.9	-0.055 ± 0.009	-0.050 ± 0.006	-0.049 ± 0.005
	0.99	0.99	-0.050 ± 0.002	-0.065 ± 0.003	-0.074 ± 0.003
	0.999	0.9	-0.057 ± 0.007	-0.053 ± 0.003	-0.051 ± 0.005
	0.999	0.99	-0.058 ± 0.003	-0.073 ± 0.004	-0.083 ± 0.003

Here, w_i is constructed from the delayed advantage $\widetilde{A}_{j_m(i)}$, whereas $\widetilde{Q}_i$ serves as the delayed reference for measuring the temporal drift of estimator i. We evaluate C_{temp} at ages 25, 50, and 75 within the $D = 100$ delay interval for the same two representative environments used in the main text. Immediately after a refresh, the current and delayed critics coincide by construction, so Δ_i^{temp} is identically zero and the corresponding Pearson correlation is undefined; we therefore omit age 0.

As the delayed snapshot ages, the current critic evolves relative to its delayed reference, and nonzero temporal correlations emerge between the filtering weights and the resulting critic drift. The correlations are predominantly negative across the reported settings. For $\tau = 0.99$, their magnitude also tends to increase at later delay ages, whereas the pattern is weaker or less monotonic for $\tau = 0.9$. These results characterize the temporal separation induced by delayed filtering.

E.3 Estimator-Level Decoupling

We additionally examine whether the filtering weight remains directly coupled to the estimator receiving the weighted update. We use the leave-one-out ensemble disagreement

$$\epsilon_i^{\text{LOO}} = Q_i - \frac{1}{N-1} \sum_{k \neq i} Q_k \tag{76}$$

as an estimator-specific disagreement proxy rather than an oracle value error. Table 6 compares the absolute correlation of the filtering weight with the LOO disagreement of the delayed estimator used to construct the filter and with that of the estimator receiving the weighted update. Across all reported settings, the latter correlation remains close to zero and is substantially smaller than the former. This pattern is consistent with the intended estimator-level separation of DD-IQL: the estimator receiving the weighted update is not itself used to construct its filtering weight.

Table 6: Estimator-level decoupling diagnostic for DD-IQL. We report the absolute correlation between the filtering weight and the LOO disagreement proxy of the delayed estimator used for filtering (Filtering est.) and that of the estimator receiving the weighted update (Updated est.).

Env.	γ	τ	Filtering est.	Updated est.
AM-L	0.99	0.9	0.194 ± 0.004	0.012 ± 0.001
	0.99	0.99	0.127 ± 0.003	0.011 ± 0.002
	0.999	0.9	0.154 ± 0.002	0.011 ± 0.002
	0.999	0.99	0.087 ± 0.005	0.011 ± 0.002
Cube-S	0.99	0.9	0.094 ± 0.001	0.005 ± 0.000
	0.99	0.99	0.066 ± 0.000	0.006 ± 0.000
	0.999	0.9	0.084 ± 0.001	0.005 ± 0.000
	0.999	0.99	0.058 ± 0.002	0.007 ± 0.001

E.4 Full Results for the Delayed and Decoupled Filtering Ablation

Section 7.4 reports the delayed and decoupled filtering ablation using performance averaged over the nine OGBench environments. Here, we provide the complete per-environment results underlying those averages. We evaluate Delayed-Filtering IQL (DF-IQL), Self-Delayed IQL (SD-IQL), and DD-IQL across delay periods $D \in \{1, 10, 100, 1000\}$ and expectile levels $\tau \in \{0.9, 0.99\}$. As in the main-text ablation, all methods use the same budget of two critic estimators so that the comparison focuses on the effects of delayed filtering and cross-estimator decoupling.

Table 7 reports the results under the original environment-dependent discount-factor settings ($\gamma = 0.99$ or $\gamma = 0.995$), while Table 8 reports the corresponding results under the severe-bias setting with $\gamma = 0.999$.

The per-environment results are consistent with the trends observed in the averaged results of Section 7.4. DF-IQL and SD-IQL are more sensitive to the choice of delay period, and larger delays improve performance in several challenging settings. DD-IQL remains comparatively stable across the tested values of D, particularly at the higher expectile level $\tau = 0.99$ and discount factor $\gamma = 0.999$. These results further illustrate the complementary effects of temporal delay and cross-estimator decoupling under stronger bias-propagation conditions.

Table 7: Full per-environment performance for the delayed and decoupled filtering ablation on OGBench navigation tasks under the original discount-factor settings ($\gamma = 0.99$ or $\gamma = 0.995$, depending on the environment). Results are mean $\pm$ standard deviation over four seeds.

Algorithm	D	τ	AM-L	HM-M	HM-L	AS-A	Cube-S	Cube-D	Pz-3x3	Pz-4x4	Scene	Avg
DF-IQL	10	0.9	0.12 ± 0.02	0.45 ± 0.02	0.04 ± 0.01	0.08 ± 0.04	0.89 ± 0.04	0.01 ± 0.01	0.01 ± 0.01	0.04 ± 0.01	0.18 ± 0.08	0.20
	10	0.99	0.00 ± 0.00	0.28 ± 0.02	0.00 ± 0.00	0.02 ± 0.01	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.01	0.00 ± 0.00	0.03
	100	0.9	0.13 ± 0.04	0.40 ± 0.02	0.02 ± 0.01	0.06 ± 0.02	0.86 ± 0.06	0.00 ± 0.00	0.01 ± 0.00	0.03 ± 0.02	0.04 ± 0.01	0.17
	100	0.99	0.00 ± 0.00	0.26 ± 0.06	0.00 ± 0.01	0.02 ± 0.02	0.76 ± 0.26	0.00 ± 0.00	0.00 ± 0.00	0.02 ± 0.02	0.01 ± 0.02	0.12
	1000	0.9	0.08 ± 0.02	0.26 ± 0.06	0.02 ± 0.01	0.06 ± 0.02	0.92 ± 0.03	0.00 ± 0.00	0.02 ± 0.00	0.03 ± 0.02	0.08 ± 0.01	0.16
	1000	0.99	0.00 ± 0.00	0.35 ± 0.04	0.02 ± 0.01	0.03 ± 0.02	0.91 ± 0.04	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.01	0.19 ± 0.04	0.17
SD-IQL	10	0.9	0.00 ± 0.00	0.55 ± 0.03	0.07 ± 0.03	0.01 ± 0.01	0.08 ± 0.13	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.01	0.00 ± 0.00	0.08
	10	0.99	0.00 ± 0.00	0.00 ± 0.01	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00
	100	0.9	0.12 ± 0.02	0.58 ± 0.07	0.05 ± 0.02	0.05 ± 0.03	0.89 ± 0.06	0.00 ± 0.00	0.01 ± 0.01	0.02 ± 0.01	0.03 ± 0.03	0.19
	100	0.99	0.00 ± 0.00	0.01 ± 0.01	0.00 ± 0.00	0.01 ± 0.01	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00
	1000	0.9	0.22 ± 0.03	0.52 ± 0.03	0.05 ± 0.00	0.06 ± 0.02	0.89 ± 0.05	0.00 ± 0.00	0.01 ± 0.01	0.02 ± 0.01	0.04 ± 0.02	0.20
	1000	0.99	0.00 ± 0.00	0.08 ± 0.02	0.00 ± 0.00	0.03 ± 0.01	0.78 ± 0.05	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.10
DD-IQL	10	0.9	0.08 ± 0.04	0.24 ± 0.03	0.01 ± 0.01	0.07 ± 0.02	0.94 ± 0.01	0.00 ± 0.00	0.01 ± 0.01	0.06 ± 0.01	0.06 ± 0.03	0.16
	10	0.99	0.21 ± 0.02	0.36 ± 0.05	0.02 ± 0.02	0.06 ± 0.02	0.93 ± 0.04	0.00 ± 0.00	0.02 ± 0.02	0.05 ± 0.01	0.56 ± 0.15	0.24
	100	0.9	0.08 ± 0.03	0.24 ± 0.02	0.00 ± 0.00	0.07 ± 0.02	0.91 ± 0.02	0.00 ± 0.01	0.03 ± 0.02	0.07 ± 0.02	0.05 ± 0.03	0.16
	100	0.99	0.22 ± 0.03	0.33 ± 0.04	0.01 ± 0.01	0.06 ± 0.02	0.96 ± 0.02	0.00 ± 0.00	0.00 ± 0.00	0.05 ± 0.03	0.63 ± 0.11	0.25
	1000	0.9	0.08 ± 0.02	0.17 ± 0.02	0.01 ± 0.01	0.06 ± 0.00	0.91 ± 0.05	0.00 ± 0.00	0.02 ± 0.01	0.05 ± 0.02	0.06 ± 0.02	0.15
	1000	0.99	0.19 ± 0.05	0.27 ± 0.05	0.01 ± 0.01	0.05 ± 0.02	0.95 ± 0.03	0.00 ± 0.00	0.02 ± 0.02	0.03 ± 0.02	0.63 ± 0.08	0.24

Table 8: Full per-environment performance for the delayed and decoupled filtering ablation on OGBench navigation tasks under the severe-bias discount setting ($\gamma = 0.999$). Results are mean $\pm$ standard deviation over four seeds.

Algorithm	D	τ	AM-L	HM-M	HM-L	AS-A	Cube-S	Cube-D	Pz-3x3	Pz-4x4	Scene	Avg
DF-IQL	10	0.9	0.09 ± 0.04	0.45 ± 0.03	0.07 ± 0.01	0.00 ± 0.00	0.66 ± 0.39	0.00 ± 0.00	0.01 ± 0.01	0.06 ± 0.01	0.16 ± 0.17	0.17
	10	0.99	0.04 ± 0.03	0.16 ± 0.02	0.01 ± 0.01	0.01 ± 0.01	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.02 ± 0.01	0.00 ± 0.00	0.03
	100	0.9	0.10 ± 0.04	0.44 ± 0.05	0.06 ± 0.02	0.01 ± 0.01	0.74 ± 0.30	0.00 ± 0.00	0.02 ± 0.01	0.03 ± 0.01	0.11 ± 0.11	0.17
	100	0.99	0.01 ± 0.00	0.22 ± 0.12	0.02 ± 0.00	0.01 ± 0.01	0.46 ± 0.45	0.00 ± 0.00	0.00 ± 0.00	0.02 ± 0.02	0.00 ± 0.00	0.08
	1000	0.9	0.13 ± 0.02	0.38 ± 0.04	0.04 ± 0.01	0.00 ± 0.01	0.92 ± 0.02	0.00 ± 0.00	0.03 ± 0.01	0.04 ± 0.02	0.86 ± 0.06	0.27
	1000	0.99	0.00 ± 0.00	0.20 ± 0.03	0.04 ± 0.01	0.03 ± 0.02	0.90 ± 0.03	0.00 ± 0.01	0.00 ± 0.00	0.02 ± 0.02	0.33 ± 0.29	0.17
SD-IQL	10	0.9	0.00 ± 0.01	0.57 ± 0.05	0.12 ± 0.04	0.02 ± 0.01	0.02 ± 0.02	0.00 ± 0.00	0.00 ± 0.00	0.04 ± 0.01	0.00 ± 0.00	0.09
	10	0.99	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00
	100	0.9	0.01 ± 0.02	0.52 ± 0.01	0.08 ± 0.01	0.02 ± 0.02	0.21 ± 0.36	0.00 ± 0.00	0.00 ± 0.00	0.05 ± 0.03	0.00 ± 0.00	0.10
	100	0.99	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.01	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00
	1000	0.9	0.00 ± 0.01	0.53 ± 0.04	0.10 ± 0.02	0.03 ± 0.01	0.88 ± 0.02	0.00 ± 0.00	0.01 ± 0.01	0.03 ± 0.02	0.10 ± 0.08	0.19
	1000	0.99	0.00 ± 0.00	0.13 ± 0.04	0.00 ± 0.00	0.01 ± 0.01	0.87 ± 0.05	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.11
DD-IQL	10	0.9	0.12 ± 0.01	0.45 ± 0.04	0.04 ± 0.02	0.03 ± 0.01	0.95 ± 0.03	0.00 ± 0.00	0.03 ± 0.02	0.07 ± 0.04	0.86 ± 0.06	0.28
	10	0.99	0.18 ± 0.03	0.51 ± 0.06	0.06 ± 0.01	0.13 ± 0.01	0.96 ± 0.01	0.00 ± 0.00	0.02 ± 0.00	0.05 ± 0.01	0.90 ± 0.08	0.31
	100	0.9	0.07 ± 0.02	0.38 ± 0.06	0.05 ± 0.01	0.01 ± 0.01	0.91 ± 0.06	0.00 ± 0.00	0.04 ± 0.02	0.08 ± 0.03	0.97 ± 0.02	0.28
	100	0.99	0.15 ± 0.03	0.40 ± 0.03	0.05 ± 0.01	0.08 ± 0.02	0.96 ± 0.02	0.00 ± 0.00	0.01 ± 0.01	0.04 ± 0.02	0.89 ± 0.05	0.29
	1000	0.9	0.09 ± 0.01	0.36 ± 0.02	0.04 ± 0.01	0.01 ± 0.01	0.93 ± 0.03	0.00 ± 0.00	0.03 ± 0.01	0.09 ± 0.01	0.95 ± 0.03	0.28
	1000	0.99	0.16 ± 0.05	0.40 ± 0.05	0.06 ± 0.02	0.10 ± 0.04	0.95 ± 0.02	0.00 ± 0.00	0.01 ± 0.01	0.08 ± 0.04	0.91 ± 0.02	0.30

E.5 Results on Additional OGBench Tasks

To complement the default-task evaluation in the main text, we additionally evaluate the methods on the remaining non-default task variants of the same OGBench environments. These experiments use the same training protocol and hyperparameter configurations as the corresponding default-task experiments. Tables 9 and 10 report the results under the original environment-dependent discount factors and the high-discount setting $\gamma = 0.999$, respectively.

Under the original discount-factor settings, DD-IQL achieves average scores of 0.21 and 0.20 for $\tau = 0.9$ and $\tau = 0.99$, compared with 0.18 and 0.03 for the corresponding IQL baselines. The improvement is more pronounced under $\gamma = 0.999$, where DD-IQL obtains average scores of 0.27 and 0.24, compared with 0.12 and 0.03 for IQL. The gains vary across individual task variants, but the aggregate results indicate that the advantage of DD-IQL over standard IQL is not restricted to the default tasks used in the main evaluation and becomes particularly visible under the high-discount setting.

As an auxiliary extension, DD-ReBRAC also improves the average score over ReBRAC in both settings, from 0.35 to 0.51 under the original discount factors and from 0.36 to 0.50 under $\gamma = 0.999$. These additional results provide broader empirical coverage of the proposed design principle across task variants.

Table 9: Performance on non-default OGBench tasks under the original discount-factor settings ($\gamma = 0.99$ or $\gamma = 0.995$, depending on the environment). Task-specific environment abbreviations follow the notation defined in Table 4. Results are mean $\pm$ standard deviation over four seeds.

Environment	IQL ($\tau = 0.9$)	IQL ($\tau = 0.99$)	**DD-IQL** ($\tau = 0.9$)	**DD-IQL** ($\tau = 0.99$)	ReBRAC	**DD-ReBRAC**	FQL
AM-L-T2	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.87 ± 0.03	0.90 ± 0.03	0.49 ± 0.10
AM-L-T3	0.21 ± 0.05	0.00 ± 0.00	0.18 ± 0.05	0.14 ± 0.03	0.82 ± 0.06	0.72 ± 0.09	0.92 ± 0.02
AM-L-T4	0.02 ± 0.01	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.86 ± 0.06	0.64 ± 0.36	0.81 ± 0.04
AM-L-T5	0.15 ± 0.06	0.00 ± 0.00	0.09 ± 0.01	0.16 ± 0.04	0.86 ± 0.07	0.90 ± 0.01	0.84 ± 0.07
HM-M-T2	0.59 ± 0.03	0.38 ± 0.02	0.47 ± 0.02	0.53 ± 0.06	0.26 ± 0.13	0.96 ± 0.02	0.84 ± 0.22
HM-M-T3	0.26 ± 0.07	0.04 ± 0.02	0.20 ± 0.07	0.15 ± 0.01	0.46 ± 0.23	0.95 ± 0.02	0.53 ± 0.26
HM-M-T4	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.28 ± 0.18	0.45 ± 0.20	0.02 ± 0.02
HM-M-T5	0.34 ± 0.03	0.05 ± 0.02	0.12 ± 0.03	0.10 ± 0.02	0.32 ± 0.08	0.97 ± 0.01	0.97 ± 0.02
HM-L-T2	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.00	0.00 ± 0.00
HM-L-T3	0.06 ± 0.02	0.01 ± 0.00	0.10 ± 0.01	0.10 ± 0.03	0.21 ± 0.09	0.35 ± 0.20	0.10 ± 0.03
HM-L-T4	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.01	0.01 ± 0.00	0.01 ± 0.00	0.17 ± 0.04	0.00 ± 0.00
HM-L-T5	0.02 ± 0.01	0.00 ± 0.00	0.02 ± 0.01	0.01 ± 0.01	0.01 ± 0.01	0.05 ± 0.02	0.00 ± 0.00
AS-A-T1	0.19 ± 0.09	0.05 ± 0.01	0.27 ± 0.02	0.25 ± 0.02	0.01 ± 0.01	0.76 ± 0.03	0.81 ± 0.02
AS-A-T2	0.14 ± 0.08	0.02 ± 0.02	0.08 ± 0.03	0.12 ± 0.03	0.42 ± 0.25	0.94 ± 0.02	0.92 ± 0.02
AS-A-T3	0.19 ± 0.02	0.04 ± 0.04	0.04 ± 0.01	0.05 ± 0.01	0.00 ± 0.00	0.59 ± 0.03	0.55 ± 0.05
AS-A-T5	0.09 ± 0.04	0.04 ± 0.02	0.01 ± 0.01	0.02 ± 0.01	0.00 ± 0.00	0.61 ± 0.05	0.33 ± 0.02
Cube-S-T1	0.63 ± 0.36	0.00 ± 0.01	0.88 ± 0.04	0.88 ± 0.06	0.89 ± 0.04	0.89 ± 0.04	0.93 ± 0.04
Cube-S-T3	0.08 ± 0.12	0.01 ± 0.00	0.89 ± 0.04	0.93 ± 0.03	0.91 ± 0.02	0.94 ± 0.01	0.98 ± 0.02
Cube-S-T4	0.78 ± 0.04	0.02 ± 0.01	0.83 ± 0.02	0.82 ± 0.08	0.85 ± 0.02	0.87 ± 0.02	0.90 ± 0.04
Cube-S-T5	0.66 ± 0.14	0.34 ± 0.32	0.83 ± 0.03	0.76 ± 0.02	0.84 ± 0.03	0.83 ± 0.04	0.84 ± 0.04
Cube-D-T1	0.17 ± 0.06	0.00 ± 0.00	0.05 ± 0.02	0.06 ± 0.06	0.37 ± 0.08	0.56 ± 0.12	0.54 ± 0.07
Cube-D-T3	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.02 ± 0.01	0.32 ± 0.04	0.20 ± 0.01
Cube-D-T4	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.01	0.02 ± 0.01	0.02 ± 0.02
Cube-D-T5	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.01	0.49 ± 0.05	0.14 ± 0.05
Pz-3x3-T1	0.49 ± 0.19	0.00 ± 0.00	0.67 ± 0.06	0.79 ± 0.05	0.98 ± 0.01	0.96 ± 0.02	0.91 ± 0.04
Pz-3x3-T2	0.08 ± 0.06	0.00 ± 0.00	0.02 ± 0.03	0.00 ± 0.01	0.05 ± 0.04	0.05 ± 0.04	0.14 ± 0.02
Pz-3x3-T3	0.02 ± 0.01	0.00 ± 0.00	0.01 ± 0.01	0.00 ± 0.01	0.05 ± 0.01	0.03 ± 0.01	0.11 ± 0.03
Pz-3x3-T5	0.02 ± 0.02	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.01	0.07 ± 0.04	0.07 ± 0.01	0.18 ± 0.08
Pz-4x4-T1	0.11 ± 0.04	0.06 ± 0.03	0.13 ± 0.02	0.17 ± 0.04	0.20 ± 0.03	0.32 ± 0.05	0.32 ± 0.03
Pz-4x4-T2	0.04 ± 0.01	0.03 ± 0.02	0.04 ± 0.01	0.02 ± 0.01	0.09 ± 0.02	0.12 ± 0.03	0.15 ± 0.04
Pz-4x4-T3	0.12 ± 0.04	0.09 ± 0.04	0.17 ± 0.02	0.17 ± 0.02	0.14 ± 0.04	0.18 ± 0.05	0.18 ± 0.01
Pz-4x4-T5	0.03 ± 0.01	0.01 ± 0.01	0.03 ± 0.02	0.01 ± 0.01	0.07 ± 0.02	0.09 ± 0.01	0.07 ± 0.01
Scene-T1	0.78 ± 0.29	0.00 ± 0.00	0.98 ± 0.01	0.89 ± 0.05	0.97 ± 0.02	0.96 ± 0.02	1.00 ± 0.00
Scene-T3	0.31 ± 0.07	0.00 ± 0.00	0.30 ± 0.07	0.11 ± 0.05	0.71 ± 0.07	0.80 ± 0.05	0.95 ± 0.02
Scene-T4	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.03 ± 0.02	0.01 ± 0.01	0.09 ± 0.04
Scene-T5	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Avg	0.18	0.03	0.21	0.20	0.35	0.51	0.47

Table 10: Performance on non-default OGBench tasks under the severe-bias setting ($\gamma = 0.999$). Default tasks are excluded because they are reported separately in the main results. Task-specific environment abbreviations follow the notation defined in Table 4. Results are mean $\pm$ standard deviation over four seeds.

Environment	IQL ($\tau = 0.9$)	IQL ($\tau = 0.99$)	**DD-IQL** ($\tau = 0.9$)	**DD-IQL** ($\tau = 0.99$)	ReBRAC	**DD-ReBRAC**	FQL
AM-L-T2	0.01 ± 0.01	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.30 ± 0.29	0.02 ± 0.03	0.04 ± 0.05
AM-L-T3	0.35 ± 0.08	0.04 ± 0.05	0.14 ± 0.03	0.15 ± 0.04	0.77 ± 0.08	0.79 ± 0.06	0.88 ± 0.05
AM-L-T4	0.16 ± 0.07	0.00 ± 0.00	0.01 ± 0.01	0.00 ± 0.00	0.41 ± 0.33	0.03 ± 0.05	0.00 ± 0.00
AM-L-T5	0.25 ± 0.15	0.00 ± 0.00	0.40 ± 0.08	0.40 ± 0.08	0.60 ± 0.23	0.43 ± 0.25	0.59 ± 0.39
HM-M-T2	0.58 ± 0.06	0.41 ± 0.03	0.63 ± 0.03	0.68 ± 0.03	0.61 ± 0.30	0.98 ± 0.01	0.82 ± 0.16
HM-M-T3	0.15 ± 0.06	0.03 ± 0.02	0.17 ± 0.03	0.15 ± 0.04	0.51 ± 0.10	0.97 ± 0.00	0.00 ± 0.01
HM-M-T4	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.22 ± 0.25	0.29 ± 0.27	0.00 ± 0.00
HM-M-T5	0.50 ± 0.07	0.07 ± 0.02	0.21 ± 0.03	0.09 ± 0.01	0.27 ± 0.19	0.98 ± 0.01	0.57 ± 0.30
HM-L-T2	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.07 ± 0.08	0.00 ± 0.00
HM-L-T3	0.08 ± 0.02	0.03 ± 0.01	0.09 ± 0.01	0.10 ± 0.02	0.17 ± 0.25	0.29 ± 0.08	0.04 ± 0.02
HM-L-T4	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.24 ± 0.17	0.00 ± 0.00
HM-L-T5	0.04 ± 0.04	0.00 ± 0.00	0.17 ± 0.05	0.15 ± 0.02	0.00 ± 0.00	0.41 ± 0.15	0.00 ± 0.00
AS-A-T1	0.04 ± 0.03	0.05 ± 0.03	0.30 ± 0.02	0.34 ± 0.03	0.00 ± 0.00	0.83 ± 0.02	0.63 ± 0.10
AS-A-T2	0.15 ± 0.12	0.02 ± 0.01	0.09 ± 0.03	0.25 ± 0.03	0.13 ± 0.11	0.89 ± 0.03	0.66 ± 0.20
AS-A-T3	0.17 ± 0.04	0.06 ± 0.02	0.15 ± 0.04	0.21 ± 0.03	0.06 ± 0.03	0.56 ± 0.04	0.39 ± 0.09
AS-A-T5	0.08 ± 0.04	0.03 ± 0.02	0.04 ± 0.01	0.12 ± 0.02	0.04 ± 0.04	0.33 ± 0.04	0.11 ± 0.02
Cube-S-T1	0.01 ± 0.01	0.00 ± 0.00	0.92 ± 0.03	0.88 ± 0.04	0.86 ± 0.03	0.89 ± 0.04	0.93 ± 0.02
Cube-S-T3	0.23 ± 0.40	0.01 ± 0.00	0.92 ± 0.03	0.94 ± 0.01	0.88 ± 0.04	0.87 ± 0.08	0.96 ± 0.02
Cube-S-T4	0.26 ± 0.28	0.00 ± 0.00	0.84 ± 0.05	0.79 ± 0.07	0.86 ± 0.04	0.87 ± 0.01	0.89 ± 0.07
Cube-S-T5	0.10 ± 0.12	0.20 ± 0.27	0.88 ± 0.04	0.80 ± 0.02	0.78 ± 0.05	0.80 ± 0.03	0.81 ± 0.04
Cube-D-T1	0.02 ± 0.02	0.00 ± 0.00	0.20 ± 0.03	0.21 ± 0.08	0.07 ± 0.03	0.23 ± 0.04	0.34 ± 0.06
Cube-D-T3	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.14 ± 0.07	0.15 ± 0.04
Cube-D-T4	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.02 ± 0.01	0.01 ± 0.01
Cube-D-T5	0.00 ± 0.00	0.00 ± 0.00	0.02 ± 0.01	0.03 ± 0.01	0.00 ± 0.00	0.30 ± 0.02	0.28 ± 0.08
Pz-3x3-T1	0.13 ± 0.19	0.00 ± 0.00	0.40 ± 0.06	0.47 ± 0.04	0.98 ± 0.03	0.98 ± 0.01	1.00 ± 0.00
Pz-3x3-T2	0.04 ± 0.07	0.00 ± 0.00	0.18 ± 0.03	0.19 ± 0.07	0.91 ± 0.05	0.95 ± 0.05	0.98 ± 0.01
Pz-3x3-T3	0.02 ± 0.01	0.00 ± 0.00	0.08 ± 0.04	0.07 ± 0.03	0.89 ± 0.06	0.93 ± 0.02	0.89 ± 0.01
Pz-3x3-T5	0.02 ± 0.02	0.00 ± 0.00	0.13 ± 0.04	0.08 ± 0.02	0.94 ± 0.02	0.95 ± 0.03	0.94 ± 0.02
Pz-4x4-T1	0.12 ± 0.03	0.00 ± 0.00	0.22 ± 0.04	0.19 ± 0.04	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.01
Pz-4x4-T2	0.06 ± 0.01	0.00 ± 0.00	0.05 ± 0.02	0.02 ± 0.01	0.01 ± 0.01	0.01 ± 0.00	0.00 ± 0.01
Pz-4x4-T3	0.10 ± 0.02	0.03 ± 0.04	0.29 ± 0.03	0.21 ± 0.04	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Pz-4x4-T5	0.03 ± 0.02	0.00 ± 0.00	0.05 ± 0.02	0.04 ± 0.02	0.00 ± 0.00	0.01 ± 0.00	0.01 ± 0.00
Scene-T1	0.25 ± 0.42	0.10 ± 0.10	0.88 ± 0.03	0.91 ± 0.07	0.80 ± 0.05	0.92 ± 0.03	1.00 ± 0.00
Scene-T3	0.23 ± 0.13	0.00 ± 0.00	0.16 ± 0.06	0.11 ± 0.05	0.48 ± 0.16	0.70 ± 0.18	0.89 ± 0.12
Scene-T4	0.00 ± 0.00	0.00 ± 0.00	0.67 ± 0.19	0.03 ± 0.04	0.24 ± 0.24	0.00 ± 0.00	0.24 ± 0.21
Scene-T5	0.00 ± 0.00	0.00 ± 0.00	0.58 ± 0.12	0.02 ± 0.02	0.07 ± 0.11	0.30 ± 0.11	0.00 ± 0.00
Avg	0.12	0.03	0.27	0.24	0.36	0.50	0.42

F Hyperparameters and Computational Resources

This section summarizes the hyperparameter configurations and computational resources used in our experiments. We first describe the common and environment-specific hyperparameters for the evaluated methods, including the additional settings introduced for DD-IQL and DD-ReBRAC. We then report the hardware and software environment used for training and provide a runtime and memory profile under a representative OGBench task.

F.1 Hyperparameter Settings

We follow established implementations and previously tuned OGBench configurations for all baseline methods rather than introducing additional environment-specific tuning. For IQL, our implementation follows CORL (Tarasov et al., 2022), while the OGBench-specific hyperparameters are adopted from the IQL baseline configuration used in FQL (Park et al., 2025b). For ReBRAC (Tarasov et al., 2023) and FQL, we follow the corresponding implementations and OGBench configurations from FQL. Thus, the baseline-specific parameters reported below are inherited from existing configurations rather than selected specifically for our experiments.

Table 11: Common hyperparameters used in the state-based OGBench experiments. DD-IQL and DD-ReBRAC inherit the corresponding baseline hyperparameters from IQL and ReBRAC, respectively. For DD-IQL, the ensemble size denotes the number of paired Q and V estimators; for the other ensemble-based methods, it denotes the number of critic estimates.

Hyperparameter	**IQL**	**DD-IQL**	**ReBRAC**	**DD-ReBRAC**	**FQL**
Training updates			10^6		
Batch size			256		
Learning rate			3×10^{-4}		
Soft target update rate ρ			0.005		
Evaluation interval			10^5		
Evaluation episodes			50		
State normalization	Yes	Yes	Yes	Yes	No
Reward normalization	No	No	No	No	No
Hidden activation	ReLU	ReLU	GELU	GELU	GELU
Actor layer normalization	No	No	No	No	No
Critic/value layer normalization	No	No	Yes	Yes	Yes
Target policy noise	–	–	0.2	0.2	–
Target noise clip	–	–	0.5	0.5	–
Actor update interval	1	1	2	2	–
Flow steps	–	–	–	–	10
Ensemble size	2	$N = 20$	2	$N_Q = 50$	2
Delay period D	–	100	–	100	–

For the nine state-based OGBench tasks considered in this work, all methods are trained for 10^6 gradient updates with a batch size of 256. Policies are evaluated every 10^5 updates using 50 episodes. Unless otherwise specified, all learned components use a learning rate of 3×10^{-4}, and target networks are updated with a soft-update rate of $\rho = 0.005$. The results reported in the tables are averaged over the final three evaluation checkpoints, with each checkpoint computed from 50 evaluation episodes. The common method-level hyperparameters are summarized in Table 11, and the environment-specific parameters are reported in Table 12.

DD-IQL inherits all environment-specific hyperparameters from IQL. Its only additional method-specific hyperparameters are the estimator ensemble size and delay period, which are fixed to $N = 20$ and $D = 100$, respectively, in all main experiments. Likewise, DD-ReBRAC inherits the corresponding ReBRAC hyperparameters and uses a critic ensemble of size $N_Q = 50$ with a delayed-actor update period of $D = 100$. We perform no additional environment-specific tuning for either proposed variant.

For the original setting, we use the environment-dependent discount factors γ_{orig} listed in Table 12. To examine longer effective horizons, we additionally evaluate $\gamma = 0.999$ while keeping the remaining hyperparameters fixed. For IQL and DD-IQL, the standard expectile level is $\tau = 0.9$, and we additionally consider $\tau = 0.99$ to study a more aggressive upper-expectile setting. Accordingly, the IQL-based experiments use the four combinations $(\gamma_{\text{orig}}, 0.9)$, $(\gamma_{\text{orig}}, 0.99)$, $(0.999, 0.9)$, and $(0.999, 0.99)$.

F.2 Computational Resources and Runtime

We additionally profile the computational cost of each method on the Cube Single environment. All methods are evaluated for 20,000 gradient updates with a batch size of 256 on a single NVIDIA RTX A6000 GPU with 48 GB of memory. The host machine is equipped with two Intel Xeon Gold 6240R CPUs and approximately 504 GiB of RAM. The software environment uses Python 3.10.21, JAX/JAXlib 0.4.1, and Flax 0.6.1. We verified on fixed inputs that the profiling implementation produces the same updated JAX training state as the original implementation.

Table 13 reports the measured wall-clock time for 20,000 gradient updates and the peak GPU memory usage. DD-IQL with $N = 2$ requires 63.65 seconds, compared with 61.16 seconds for IQL, while using similar GPU memory. Under the main configuration with $N = 20$, DD-IQL requires 102.60 seconds and 0.779 GiB of peak GPU memory. The increase in computational cost from $N = 2$ to $N = 20$ is therefore primarily associated with the larger estimator ensemble rather than the delayed-update mechanism itself.

Table 12: Environment-specific hyperparameters for the nine state-based OGBench tasks. γ_{orig} denotes the discount factor used in the original setting. For IQL and DD-IQL, β is the inverse temperature used for advantage-weighted policy extraction. For ReBRAC and DD-ReBRAC, α_{actor} and α_{critic} are the actor and critic behavior-regularization coefficients. For FQL, α denotes its method-specific coefficient, and Q-aggregation indicates the aggregation rule used across value estimates.

Environment	γ_{orig}	**IQL/DD-IQL network**	β	α_{actor}	α_{critic}	**FQL** α	**FQL** Q**-agg.**
AntMaze Large	0.99	4×512	10	0.003	0.01	10	Min
HumanoidMaze Medium	0.995	4×512	10	0.01	0.01	30	Mean
HumanoidMaze Large	0.995	4×512	10	0.01	0.01	30	Mean
AntSoccer Arena	0.995	4×512	1	0.01	0.01	10	Mean
Cube Single Play	0.99	4×256	1	1.0	0.0	300	Mean
Cube Double Play	0.99	4×256	0.3	0.1	0.0	300	Mean
Scene Play	0.99	4×256	10	0.1	0.01	300	Mean
Puzzle 3×3	0.99	4×256	10	0.3	0.01	1000	Mean
Puzzle 4×4	0.99	4×256	3	0.3	0.01	1000	Mean

Table 13: Computational cost on Cube Single for 20,000 gradient updates with batch size 256. DD-IQL uses $D = 100$ with either $N = 2$ or $N = 20$, and DD-ReBRAC uses $N_Q = 50$ and $D = 100$.

Method	**20K runtime (s)**	**Peak GPU memory**
IQL	61.16	0.418 GiB
DD-IQL ($N = 2$)	63.65	0.404 GiB
DD-IQL ($N = 20$)	102.60	0.779 GiB
ReBRAC	70.04	0.529 GiB
DD-ReBRAC ($N_Q = 50$)	257.18	4.365 GiB
FQL	68.08	0.529 GiB