Mildly Conservative Q-Learning for Offline Reinforcement Learning

Jiafei Lyu[1,*], **Xiaoteng Ma**[2*], **Xiu Li**[1†], **Zongqing Lu**[3†]
[1]Tsinghua Shenzhen International Graduate School, Tsinghua University
[2]Department of Automation, Tsinghua Unversity
[3]School of Computer Science, Peking University
{lvjf20,ma-xt17}@mails.tsinghua.edu.cn,
li.xiu@sz.tsinghua.edu.cn, zongqing.lu@pku.edu.cn

Abstract

Offline reinforcement learning (RL) defines the task of learning from a static logged dataset without continually interacting with the environment. The distribution shift between the learned policy and the behavior policy makes it necessary for the value function to stay conservative such that out-of-distribution (OOD) actions will not be severely overestimated. However, existing approaches, penalizing the unseen actions or regularizing with the behavior policy, are too pessimistic, which suppresses the generalization of the value function and hinders the performance improvement. This paper explores mild but enough conservatism for offline learning while not harming generalization. We propose Mildly Conservative Q-learning (MCQ), where OOD actions are actively trained by assigning them proper pseudo Q values. We theoretically show that MCQ induces a policy that behaves at least as well as the behavior policy and no erroneous overestimation will occur for OOD actions. Experimental results on the D4RL benchmarks demonstrate that MCQ achieves remarkable performance compared with prior work. Furthermore, MCQ shows superior generalization ability when transferring from offline to online, and significantly outperforms baselines. Our code is publicly available at https://github.com/dmksjfl/MCQ.

1 Introduction

Continually interacting with the environment of online reinforcement learning (RL) is often infeasible and unrealistic, since the data collection process of the agent may be expensive, difficult, or even dangerous, especially in real-world applications. Offline RL, instead, aims at learning from a static dataset that was previously collected by some unknown process [36], hence eliminating the need for environmental interactions during training.

The main challenge of offline RL is the distribution shift of state-action visitation frequency between the learned policy and the behavior policy. The evaluation of out-of-distribution (OOD) actions causes extrapolation error [14], which can be exacerbated through bootstrapping [34] and result in severe overestimation errors. Thus, keeping conservatism in value estimation is necessary in offline RL [24, 50, 60]. Previous methods achieve the conservatism by compelling the learned policy to be close to the behavior policy [14, 58, 34, 13, 57], by penalizing the learned value functions from being over-optimistic upon out-of-distribution (OOD) actions [35, 33, 59], or by learning without querying OOD samples [56, 8, 62, 32, 40].

[*]Equal Contribution
[†]Corresponding Authors

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

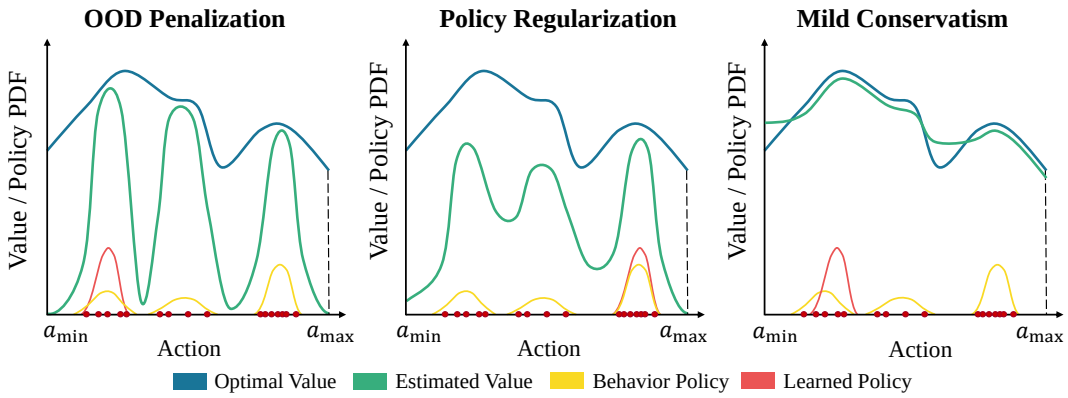


Figure 1: Comparison of prior methods against mild conservatism. The red spots represent the dataset samples. The left figure shows that penalizing OOD actions makes the value function drop sharply at the boundary of the dataset's support, which barriers policy learning. The central figure depicts that policy regularization keeps the policy near behavior policy, leading to undesired performance if the behavior policy is unsatisfying. On the right side, we illustrate the basic idea of mild conservatism. The estimated values for OOD actions are allowed to be high as long as it does not affect the learning for the optimal policy supported by the dataset, i.e., $Q(s, a^{\text{ood}}) < \max_{a \in \text{Support}(\mu)} Q(s, a)$.

In practice, we rely on neural networks to extract knowledge from the dataset and generalize it to the nearby unseen states and actions when facing continuous state and action spaces. In other words, we need the networks to "stitch" the suboptimal trajectories to generate the best possible trajectory supported by the dataset. Unfortunately, there is no free lunch. Conservatism, which offline RL celebrates, often limits the generalization and impedes the performance of the agent. Existing approaches are still inadequate in balancing conservatism and generalization. As illustrated in Figure 1, policy regularization is unreliable for offline RL when the data-collecting policy is poor, and value penalization methods often induce unnecessary pessimism in both the in-dataset region and OOD region. We argue that *the proper conservatism should be as mild as possible*. As depicted in Figure 1, we aim at well estimating the value function in the support of the dataset, and allowing value estimates upon OOD actions to be high (even higher than their optimal values) as long as $Q(s, a^{\text{ood}}) < \max_{a \in \text{Support}(\mu)} Q(s, a)$ is satisfied. The mild conservatism benefits generalization since value estimates upon OOD actions are slightly optimistic instead of being overly conservative.

To fulfill that, we propose a novel *Mildly Conservative Bellman* (MCB) operator for offline RL, where we *actively* train OOD actions and query their Q values. We theoretically analyze the convergence property of the MCB operator under the tabular MDP setting. We show that the policy induced by the MCB operator is guaranteed to behave better than the behavior policy, and can consistently improve the policy with a tighter lower bound compared with policy constraint methods or value penalization methods like CQL [35]. For practical usage, we propose the practical MCB operator and illustrate its advantages by theoretically showing that *erroneous overestimation error will not occur* with it. We then estimate the behavior policy with a conditional variational autoencoder (CVAE) [29, 51], and integrate the practical MCB operator with the Soft Actor-Critic (SAC) [20] algorithm. To this end, we propose our novel offline RL algorithm, Mildly Conservative Q-learning (MCQ).

Experimental results on the D4RL MuJoCo locomotion tasks demonstrate that MCQ surpasses recent strong baseline methods on most of the tasks, especially on non-expert datasets. Meanwhile, MCQ shows superior generalization capability when transferring from offline to online, validating our claims that mild pessimism is of importance to offline learning.

2 Preliminaries

We consider a Markov Decision Process (MDP) specified by a tuple $\langle \mathcal{S}, \mathcal{A}, r, \rho_0, p, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $r(s, a) : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function, $\rho_0(s)$ is the initial state distribution, $p(s'|s, a) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ is the transition probability, $\gamma \in [0, 1)$ is the discount

factor. Reinforcement learning (RL) aims at finding a policy $\pi(\cdot|s)$ such that the expected cumulative long-term rewards $J(\pi) = \mathbb{E}_{s_0\sim\rho_0(\cdot),a_t\sim\pi(\cdot|s_t),s_{t+1}\sim p(\cdot|s_t,a_t)}[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)]$ are maximized. The state-action function $Q(s,a)$ measures the discounted return starting from state s and action a, and following the policy π. We assume that the reward function $r(s,a)$ is bounded, i.e., $|r(s,a)| \leq r_{\max}$. Given a policy $\pi(\cdot|s)$, the Bellman backup for obtaining the corresponding Q function gives:

$$\mathcal{T}^{\pi}Q(s,a) := r(s,a) + \gamma\mathbb{E}_{s'}\mathbb{E}_{a'\sim\pi(\cdot|s')}[Q(s',a')]. \tag{1}$$

The Q function of the optimal policy satisfies the following Bellman optimal operator:

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

In offline RL setting, the online interaction is infeasible, and we can only have access to previously collected datasets $\mathcal{D} = \{s_i, a_i, r_i, s'_{i+1}, d_i\}_{i=1}^{N}$, where d is the done flag. We denote the behavior policy as $\mu(\cdot|s)$. The Bellman backup relies on actions sampled from the learned policy, $a' \sim \pi(\cdot|s')$. However, a' can lie outside of the support of μ due to the distribution shift between π and μ. The value estimates upon a' can then be arbitrarily wrong, resulting in bad policy training. Unlike prior work, we *actively* train OOD actions by constructing them pseudo target values. In this way, we retain pessimism while enjoying better generalization.

3 Mildly Conservative Q-Learning

In this section, we first formally define the MCB operator and characterize its dynamic programming properties in the tabular MDP setting. We further give a practical version of the MCB operator. We show that no erroneous overestimation will occur with the MCB operator. Finally, we incorporate the MCB operator with SAC [20] and present our novel offline RL algorithm.

3.1 Mildly Conservative Bellman (MCB) Operator

Definition 1. *The Mildly Conservative Bellman (MCB) operator is defined as*

$$\mathcal{T}_{\mathrm{MCB}}Q(s,a) = (\mathcal{T}_1\mathcal{T}_2)Q(s,a), \tag{3}$$

where

$$\mathcal{T}_1Q(s,a) = \begin{cases} Q(s,a), & \mu(a|s) > 0. \\ \max_{a'\sim\mathrm{Support}(\mu(\cdot|s))} Q(s,a') - \delta, & \textit{else}. \end{cases} \tag{4}$$

$$\mathcal{T}_2Q(s,a) = \begin{cases} r(s,a) + \gamma\mathbb{E}_{s'}\left[\max_{a'\in\mathcal{A}} Q(s',a')\right], & \mu(a|s) > 0, \\ Q(s,a), & \textit{else}. \end{cases} \tag{5}$$

The basic idea behind this novel operator is that if the learned policy outputs actions that lie in the support region of μ, then we go for backup; while if OOD actions are generated, we deliberately replace their value estimates with $\max_{a'\sim\mathrm{Support}(\mu(\cdot|s))} Q(s,a') - \delta$, where $\delta > 0$ can be arbitrarily small. That is, different from standard Bellman backup, we set up a *checking procedure* (i.e., $\mathcal{T}_1$) of whether the previous backup (i.e., $\mathcal{T}_2$) involves OOD actions for the update. Intrinsically, we construct pseudo target values for OOD actions. We subtract a small positive δ such that OOD actions will not be chosen when executing policy via $\arg\max_{a\in\mathcal{A}} Q(s,a)$.

For a better understanding of the MCB operator, we theoretically analyze its dynamic programming properties in the tabular MDP setting. All proofs are deferred to Appendix A.

Proposition 1. *In the support region of the behavior policy, i.e.,* $\mathrm{Support}(\mu)$*, the MCB operator is a γ-contraction operator in the $\mathcal{L}_\infty$ norm, and any initial Q function can converge to a unique fixed point by repeatedly applying $\mathcal{T}_{\mathrm{MCB}}$.*

Proposition 2 (Behave at least as well as behavior policy)**.** *Denote Q_{MCB} as the unique fixed point acquired by the MCB operator, then in* $\mathrm{Support}(\mu)$ *we have: $Q_\mu \leq Q_{\mathrm{MCB}} \leq Q_{\mu^*}$, where Q_μ is the Q function of the behavior policy and Q_{μ^*} is the Q function of the optimal policy in the batch.*

Proposition 2 indicates that the policy induced by the MCB operator can behave at least as well as the behavior policy, and can approximate the optimal batch-constraint policy. Apart from this advantage, we further show that the MCB operator results in milder conservatism. We start by observing that

value penalization method, like CQL [35], guarantees that the learned value function $\hat{Q}^\pi(s,a)$ is a lower bound of its true value $Q^\pi(s,a)$. It is also ensured that following such conservative update leads to a safe policy improvement, i.e., $J(\pi_{\text{CQL}}) \geq J(\mu) - \mathcal{O}(\frac{1}{(1-\gamma)^2})$ (Theorem 3.6 in [35]). For explicit policy constraint methods, e.g., TD3+BC [13], the learned policy π_p mimics the behavior policy μ, and can hardly behave significantly better than μ. We show in Proposition 3 that explicit policy constraint methods also exhibit a safe policy improvement, $J(\pi_p) \geq J(\mu) - \mathcal{O}(\frac{1}{(1-\gamma)^2})$, while the MCB operator can consistently improve the policy with a tighter lower bound.

Proposition 3 (Milder Pessimism)**.** *Suppose there exists an explicit policy constraint offline reinforcement learning algorithm such that the KL-divergence of the learned policy $\pi_p(\cdot|s)$ and the behavior policy $\mu(\cdot|s)$ is optimized to guarantee $\max\left(\text{KL}(\mu,\pi_p), \text{KL}(\pi_p,\mu)\right) \leq \epsilon$, $\forall s$. Denote $\epsilon_\mu^{\pi_p} = \max_s |\mathbb{E}_{a\sim\pi_p} A^\mu(s,a)|$, then*

$$J(\pi_p) \geq J(\mu) - \frac{\sqrt{2}\gamma\epsilon_\mu^{\pi_p}}{(1-\gamma)^2}\sqrt{\epsilon}, \tag{6}$$

while for the policy π_{MCB} learned by applying the MCB operator, we have

$$J(\pi_{\text{MCB}}) \geq J(\mu). \tag{7}$$

In summary, the MCB operator benefits the offline learning in two aspects: (1) the operator is a contraction, and any initial Q functions are guaranteed to converge to a unique fixed point; (2) the learned policy of the MCB operator is ensured to be better than the behavior policy, and reserve milder pessimism compared with policy constraint methods or CQL.

3.2 Practical MCB Operator

In practice, it is intractable to acquire $\max_{a'\sim\text{Support}(\mu(\cdot|s))} Q(s,a')$ in $\mathcal{T}_1$ of Eq. (4) in continuous control domains, and the behavior policy is often unknown. Thus, we fit an empirical behavior policy $\hat{\mu}$ with supervised learning based on the static dataset. The pseudo target values for the OOD actions are then computed by sampling N actions from $\hat{\mu}$, and taking maximum over their value evaluation. Formally, we define the practical MCB operator below, accompanied by the theoretical analysis.

Definition 2. *The practical Mildly Conservative Bellman (MCB) operator is defined as*

$$\hat{\mathcal{T}}_{\text{MCB}} Q(s,a) = (\hat{\mathcal{T}}_1 \mathcal{T}_2) Q(s,a), \tag{8}$$

where

$$\hat{\mathcal{T}}_1 Q(s,a) = \begin{cases} Q(s,a), & \mu(a|s) > 0. \\ \mathbb{E}_{\{a_i'\}^N\sim\hat{\mu}(\cdot|s)}\left[\max_{a'\sim\{a_i'\}^N} Q(s,a')\right], & \textit{else}. \end{cases} \tag{9}$$

Compared with Eq. (4), we make a small modification of $\mathcal{T}_1$, and keep $\mathcal{T}_2$ unchanged. There is no need to subtract δ here as generally $\mathbb{E}_{\{a_i'\}^N\sim\hat{\mu}(\cdot|s)}\left[\max_{a'\sim\{a_i'\}^N} Q(s,a')\right] \leq \max_{a'\sim\text{Support}(\mu)} Q(s,a')$. The practical MCB operator is much easier to implement in practice. We show that the practical MCB operator is still a γ-contraction in the support region of the behavior policy μ.

Proposition 4. *Proposition 1 still holds for the practical MCB operator.*

Since we fit the empirical distribution $\hat{\mu}$ of the behavior policy μ, there may exist a shift between $\hat{\mu}$ and μ, especially when we represent the policy via neural networks. That suggests that OOD actions a' can still be sampled from $\hat{\mu}$ such that $a' \notin \text{Support}(\mu(\cdot|s))$. Our last main result reveals that *erroneous overestimation issue will not occur* with the aid of the practical MCB operator.

Proposition 5 (No erroneous overestimation will occur)**.** *Assuming that $\sup_s D_{\text{TV}}(\hat{\mu}(\cdot|s) \parallel \mu(\cdot|s)) \leq \epsilon < \frac{1}{2}$, we have*

$$\mathbb{E}_{\{a_i'\}^N\sim\hat{\mu}(\cdot|s)}\left[\max_{a'\in\{a_i'\}^N} Q(s,a')\right] \leq \max_{a'\in\text{Support}(\mu(\cdot|s))} Q(s,a') + (1-(1-2\epsilon)^N)\frac{r_{\max}}{1-\gamma}.$$

Remark: This proposition generally requires a comparatively well-fitted empirical behavior policy $\hat{\mu}$. In practice, we model $\hat{\mu}$ with a CVAE. In most cases, CVAE can already fit the dataset well and guarantee a good performance. Whereas there may exist some situations, e.g., the dataset is highly

multi-modal, then one can replace the CVAE as the conditional GAN (CGAN) to better capture the different modes in the dataset as depicted in [61]. We believe generative models like CGAN will be a good choice by then.

Intuitively, the above conclusion says that if the empirical behavior policy $\hat{\mu}$ well fits μ, i.e., ϵ is small enough, then regardless of how $\{a'_i\}^N$ are sampled, the pseudo target value will approximate the maximum Q-value within the dataset's support with high probability. The extrapolation error is under the scale of $(1-(1-2\epsilon)^N)\frac{r_{\max}}{1-\gamma}$. We expect a good empirical behavior policy such that most of the actions sampled from it will be in-distribution. However, if ϵ is large, N can act as a trade-off parameter. The smaller N we use, the more conservative we are. Fortunately, we find empirically that our method performs well in a large interval of N over different tasks (see Section 4.2). Hence, it is safe to fix a N in practice.

3.3 Algorithm

As aforementioned, we often cannot get prior information about the behavior policy μ. Thus, we need to empirically fit a behavior policy $\hat{\mu}$ with supervised learning for applying the practical MCB operator. Our algorithm, Mildly Conservative Q-learning (MCQ), trains an additional generative model, which is also adopted by many prior work [14, 17, 33, 67]. We build our novel offline algorithm upon an off-the-shelf off-policy online RL algorithm, Soft Actor-Critic (SAC) [20].

Modelling the behavior policy with the CVAE. We utilize a conditional variational autoencoder (CVAE) [29, 51, 14] to model the behavior policy μ. Given a fixed logged dataset, the goal of the CVAE is to reconstruct actions conditioned on the states such that the reconstructed actions come from the same distribution as the actions in the dataset, i.e., $\mu(\cdot|s)$. That generally satisfies the assumption we make in Proposition 5. As concerned by [32], training a generative model like CVAE still may produce out-of-dataset actions, which leads to extrapolation error since undefined Q values can be possibly queried. Prior methods, like BCQ [14], do not well address such issue. While for our algorithm, such concern is mitigated because overestimation error is actually under control as is guaranteed by Proposition 5.

The CVAE $G_\omega(s)$ parameterized by ω is made up of an encoder $E_\xi(s,a)$ and a decoder $D_\psi(s,z)$ parameterized by ξ, ψ respectively, $\omega = \{\xi, \psi\}$. The CVAE is optimized by maximizing its variational lower bound, which is equivalent to minimizing the following objective function.

$$\mathcal{L}_{\text{CVAE}} = \mathbb{E}_{(s,a)\sim\mathcal{D}, z\sim E_\xi(s,a)} \left[(a - D_\psi(s,z))^2 + \text{KL}\left(E_\xi(s,a), \mathcal{N}(0,\mathbf{I})\right) \right], \tag{10}$$

where $\text{KL}(p,q)$ denotes the KL-divergence between probability distribution $p(\cdot)$ and $q(\cdot)$, and $\mathbf{I}$ is the identity matrix. When sampling actions from the CVAE, we first sample a latent variable z from the prior distribution, which is set to be multivariate normal distribution $\mathcal{N}(0,\mathbf{I})$, and then pass it in conjunction with the state s into the decoder $D_\psi(s,z)$ to get the desired decoded action.

It is also worth noting that we do not choose GAN [19] as the generative model because it is known to suffer from training instability and mode collapse [52, 6, 5]. Also, GAN consumes much more time and memories to train compared with the CVAE.

Loss functions. In deep RL, the Q function is represented with a neural network parameterized by θ and is updated via minimizing the temporal difference (TD) loss $\mathbb{E}_{s,a,r,s'}[(Q_\theta(s,a) - \mathcal{T}Q(s,a))^2]$. We actually are performing the regression task $(s,a) \mapsto \mathcal{T}Q(s,a)$ to train the Q function. The target value $\mathcal{T}Q(s,a)$ is usually computed by utilizing a lagging target network parameterized by θ' without gradient backpropagation. As a typical actor-critic [30, 31, 53] algorithm, SAC uses its critic networks to perform value estimation and uses a separate actor network for policy improvement. In order to incorporate the MCB operator with the off-the-shelf SAC algorithm, we need to check whether the sampled action $a' \sim \pi(\cdot|s)$ lies outside of the behavior policy's support, i.e., whether $\mu(a'|s) > 0$. However, such a criterion is not reliable, because the true behavior policy μ is unknown and it is difficult to examine whether $\mu(a'|s) > 0$ in practice. It is also problematic if we rely on the empirical behavior policy $\hat{\mu}$ to check whether a' is OOD as $\hat{\mu}$ itself can produce OOD actions.

We then resort to constructing an auxiliary loss for OOD actions and integrating it with the standard Bellman error. Specifically, we sample a^{ood} from the learned policy $\pi(\cdot|s^{\text{in}})$ based on the sampled state $s^{\text{in}} \sim \mathcal{D}$ from the dataset and assign them pseudo target values based on the practical MCB operator. Note that the superscript ood is used to distinguish from the in-dataset real actions, and a^{ood} is not necessarily an OOD action. We remark that if $a^{\text{ood}} \in \text{Support}(\mu(\cdot|s))$, the pseudo Q

Algorithm 1 Mildly Conservative Q-learning (MCQ)

Initialize CVAE G_ω, critic networks $Q_{\theta_1}, Q_{\theta_2}$ and actor network π_ϕ with random parameters
Initialize target networks $\theta_1' \leftarrow \theta_1, \theta_2' \leftarrow \theta_2$ and offline replay buffer $\mathcal{D}$.
for t = 1 to T **do**
  Sample a mini-batch $B = \{s, a, r, s', d\}$ from $\mathcal{D}$, where d is the done flag
  Train CVAE via minimizing Eq. (10)
  Get target value: $y = r(s,a) + \gamma \left[\min_{i=1,2} Q_{\theta_i'}(s',a') - \alpha \log \pi_\phi(a'|s')\right], a' \sim \pi_\phi(\cdot|s')$
  Sample N actions from π based on each s and s', set $s^{\text{in}} = \{s, s'\}$
  Compute the target value for the OOD actions via Eq. (13)
  Update critic θ_i with gradient descent via minimizing Eq. (11)
  Update actor ϕ with gradient ascent via Eq. (14)
  Update target networks: $\theta_i' \to \tau\theta_i + (1-\tau)\theta_i', \ i = 1, 2$
end for

value will not negatively affect the evaluation upon it, because in-distribution actions are still trained to approximate the optimal batch-constraint Q value. In this way, we *actively* train both possible OOD actions and in-distribution actions simultaneously via *OOD sampling*. The resulting objective function for the critic networks is presented in Eq. (11).

$$\mathcal{L}_{\text{critic}} = \lambda \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}} \left[(Q_{\theta_i}(s,a) - y)^2\right] + (1-\lambda)\mathbb{E}_{s^{\text{in}}\sim\mathcal{D}, a^{\text{ood}}\sim\pi} \left[(Q_{\theta_i}(s^{\text{in}}, a^{\text{ood}}) - y')^2\right], \quad (11)$$

where the target value for the in-distribution actions gives

$$y = r(s,a) + \gamma \left[\min_{i=1,2} Q_{\theta_i'}(s',a') - \alpha \log \pi_\phi(a'|s')\right], \alpha \in \mathbb{R}_+, \quad (12)$$

which follows the standard target value of vanilla SAC. The hyperparameter λ balances the in-distribution data training and OOD action training. Following the formulas of the practical MCB operator in Eq. (9), the pseudo target value for the OOD action is computed by:

$$y' = \min_{j=1,2} \mathbb{E}_{\{a_i'\}^N \sim \hat{\mu}} \left[\max_{a' \sim \{a_i'\}^N} Q_{\theta_j}(s^{\text{in}}, a')\right]. \quad (13)$$

Note that we experimentally find that replacing the min operator with a mean operator does not raise much difference in performance. We hence take advantage of the min operator to fulfill the pseudo clipped double Q-learning for OOD actions.

The policy is then optimized by solving the following optimization problem:

$$\pi_\phi := \max_\phi \mathbb{E}_{s\sim\mathcal{D}, a\sim\pi_\phi(\cdot|s)} \left[\min_{i=1,2} Q_{\theta_i}(s,a) - \alpha \log \pi_\phi(\cdot|s)\right]. \quad (14)$$

We detail the learning procedure of our MCQ in Algorithm 1. Different from [13], our method does not require normalization over states or value functions. The only change we make to the vanilla SAC algorithm is an extra auxiliary loss term (blue term in Eq. (11)) such that OOD actions are actively and properly trained. The additional critic loss term can also be plugged into other off-policy online RL algorithms directly. As an evidence, we combine the MCB operator with TD3 [15], yielding a deterministic version of MCQ. Please refer to Appendix B for more details.

4 Experiments

In this section, we first empirically demonstrate the effectiveness and advantages of our proposed MCQ algorithm on D4RL benchmarks [12]. We then conduct a detailed parameter study to show the hyperparameter sensitivity of MCQ. We also experimentally illustrate that the value estimation of MCQ will not incur severe overestimation and pessimistic value estimates are witnessed in practice. Finally, we show the superior offline-to-online fine-tuning benefits of MCQ on some MuJoCo datasets.

4.1 Results on MuJoCo Datasets

We experimentally compare our MCQ against behavior cloning (BC), SAC, and several recent strong baseline methods, CQL [35], UWAC [59], TD3+BC [13], and IQL [32], on D4RL [12] benchmarks.

We choose these methods as they typically represent different categories of model-free offline RL, i.e., CQL is a value penalization method, TD3+BC involves explicit policy constraint (BC loss), UWAC relies on uncertainty estimation for training, and IQL learns without querying OOD samples.

We conduct experiments on MuJoCo locomotion tasks, which are made up of five types of datasets (random, medium, medium-replay, medium-expert, and expert), yielding a total of 15 datasets. We use the most recently released "-v2" datasets for performance evaluation. The results of BC and SAC are acquired by using our implemented code. The results of CQL and UWAC are obtained by running their official codes, because the reported scores in their papers are not obtained on MuJoCo "-v2" datasets. We take the results of TD3+BC from its original paper (Table 7 in [13]). Since the IQL paper does not report its performance on MuJoCo *random* and *expert* datasets, we run IQL using the official codebase on them and take the results on medium, medium-replay, medium-expert datasets from its original paper directly. All methods are run for 1M gradient steps.

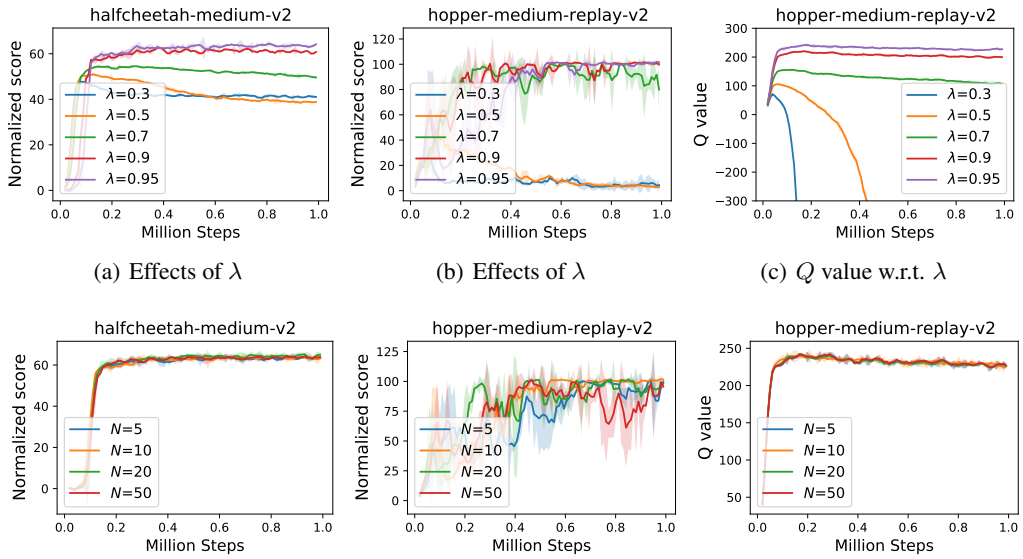
Table 1: Normalized average score comparison of MCQ against baseline methods on D4RL benchmarks over the final 10 evaluations. 0 corresponds to a random policy and 100 corresponds to an expert policy. The experiments are run on MuJoCo "-v2" datasets over 4 random seeds. r = random, m = medium, m-r = medium-replay, m-e = medium-expert, e = expert. We **bold** the highest mean.

Task Name	BC	SAC	CQL	UWAC	TD3+BC	IQL	MCQ (ours)
halfcheetah-r	2.2±0.0	**29.7**±1.4	17.5±1.5	2.3±0.0	11.0±1.1	13.1±1.3	28.5±0.6
hopper-r	3.7±0.6	9.9±1.5	7.9±0.4	2.7±0.3	8.5±0.6	7.9±0.2	**31.8**±0.5
walker2d-r	1.3±0.1	0.9±0.8	5.1±1.3	2.0±0.4	1.6±1.7	5.4±1.2	**17.0**±3.0
halfcheetah-m	43.2±0.6	55.2±27.8	47.0±0.5	42.2±0.4	48.3±0.3	47.4±0.2	**64.3**±0.2
hopper-m	54.1±3.8	0.8±0.0	53.0±28.5	50.9±4.4	59.3±4.2	66.2±5.7	**78.4**±4.3
walker2d-m	70.9±11.0	-0.3±0.2	73.3±17.7	75.4±3.0	83.7±2.1	78.3±8.7	**91.0**±0.4
halfcheetah-m-r	37.6±2.1	0.8±1.0	45.5±0.7	35.9±3.7	44.6±0.5	44.2±1.2	**56.8**±0.6
hopper-m-r	16.6±4.8	7.4±0.5	88.7±12.9	25.3±1.7	60.9±18.8	94.7±8.6	**101.6**±0.8
walker2d-m-r	20.3±9.8	-0.4±0.3	81.8±2.7	23.6±6.9	81.8±5.5	73.8±7.1	**91.3**±5.7
halfcheetah-m-e	44.0±1.6	28.4±19.4	75.6±25.7	42.7±0.3	**90.7**±4.3	86.7±5.3	87.5±1.3
hopper-m-e	53.9±4.7	0.7±0.0	105.6±12.9	44.9±8.1	98.0±9.4	91.5±14.3	**111.2**±0.1
walker2d-m-e	90.1±13.2	1.9±3.9	107.9±1.6	96.5±9.1	110.1±0.5	109.6±1.0	**114.2**±0.7
Average Above	36.5	11.3	59.1	37.0	58.2	59.9	**72.8**
halfcheetah-e	91.8±1.5	-0.8±1.8	96.3±1.3	92.9±0.6	**96.7**±1.1	95.0±0.5	96.2±0.4
hopper-e	107.7±0.7	0.7±0.0	96.5±28.0	110.5±0.5	107.8±7	109.4±0.5	**111.4**±0.4
walker2d-e	106.7±0.2	0.7±0.3	108.5±0.5	108.4±0.4	**110.2**±0.3	109.9±1.2	107.2±1.1
Total Average	49.6	9.0	67.3	50.4	67.6	68.9	**79.2**

In our experiments, we set the number of sampled actions $N = 10$ by default and tune the weighting coefficient λ. We report the λ used for all tasks in Appendix C, along with details on the experiments and implementation. We summarize the normalized average score comparison of MCQ against recent baselines in Table 1. Unsurprisingly, we observe that MCQ behaves better than BC on all of the tasks, which is consistent with our theoretical analysis in Proposition 2 and 3. MCQ also significantly outperforms the base SAC algorithm. Prior offline RL methods struggle for good performance on non-expert datasets like random and medium-replay, while MCQ surpasses them with a remarkable margin on many non-expert datasets. We attribute the less satisfying performance of prior offline RL methods to their *strict conservatism*, which restricts their generalization beyond the support of the dataset and leads to limited performance. The results, therefore, validate our claim that milder pessimism is more we need for offline learning. Furthermore, MCQ is also competitive to baselines on expert datasets. MCQ achieves the best performance on 11 out of 15 datasets, yielding a total average score of **72.8** on non-expert datasets, and an average score of **79.2** on all 15 datasets. Whereas the second best method, IQL, has an average score of 59.9 on non-expert datasets and a total average score of 68.9 across all tasks.

4.2 Parameter Study

In this subsection, we conduct a detailed parameter study on MCQ. MCQ generally contains two hyperparameters, weighting coefficient λ and number of sampled actions N. To demonstrate the parameter sensitivity of MCQ, we choose two datasets from MuJoCo locomotion tasks and conduct



(a) Effects of λ (b) Effects of λ (c) Q value w.r.t. λ

(d) Effects of N (e) Effects of. N (f) Q value w.r.t. N

Figure 2: Parameter study and Q function estimation on halfcheetah-medium-v2 and hopper-medium-replay-v2. The shaded region captures the standard deviation.

experiments on them, halfcheetah-medium-v2, and hopper-medium-replay-v2. The experiments are run for 1M gradient steps over 4 different random seeds.

Weighting coefficient λ. The weighting coefficient λ is a critical hyperparameter for MCQ, which directly controls the balance between in-distribution actions training and OOD actions training. If we set $\lambda = 1$, then MCQ degenerates into the base SAC algorithm. If λ leans towards 0, the critics will be overwhelmed by OOD actions. Intuitively, one ought not to use small λ, because more weights are desired for standard Bellman error such that in-distribution state-action pairs can be well-trained. We observe significant performance drop with smaller λ in Figure 2(a) and 2(b). Also, we find that choosing $0.7 \leq \lambda < 1$ generally induces good performance.

Number of sampled actions N. N works as a regularizer to control the potential extrapolation error. In case the behavior policy μ is known, we require N to be as large as possible to better estimate the maximum Q value. While in practice, we leverage the CVAE to approximate μ, from which OOD actions can be sampled. N then plays a role to balance pessimism and generalization. To see the influence of N, we fix $\lambda = 0.95$ for the two datasets. Experimental results in Figure 2(d) and 2(e) indicate that MCQ is insensitive to N for a wide range of N. We therefore set $N = 10$ by default.

Value estimation. We present the Q value estimates with respect to (w.r.t.) λ and N in Figure 2(c) and 2(f). The Q estimation is calculated via $\mathbb{E}_{i=1,2}\mathbb{E}_{(s,a)\sim\mathcal{D}}[Q_{\theta_i}(s,a)]$. The results illustrate that (1) smaller λ will incur severe underestimation issue (as depicted by Figure 2(c), Q values collapse with $\lambda = 0.5$ or $\lambda = 0.3$); (2) no overestimation is observed, even with a large $\lambda = 0.95$, which validates the theoretical result in Proposition 5; (3) the Q estimates resemble each other under different N. We conclude that MCQ ensures a stable and good value estimation with a proper λ.

4.3 Offline-to-online Fine-tuning

We examine the offline-to-online fine-tuning capability of MCQ against some prior strong offline RL baselines, CQL [35], TD3+BC [13], IQL [32]. We additionally compare against AWAC [45], which is designed intrinsically for offline-to-online adaptation. We conduct experiments on MuJoCo *random* and *medium-replay* datasets. It is challenging to train on these datasets for both offline and offline-to-online fine-tuning as they are non-expert, or even contain many bad transitions. We first train baselines and MCQ for 1M gradient steps offline and then perform online fine-tuning for another 100K gradient steps. Note that IQL paper [32] adopts 1M steps for online fine-tuning. However, we argue that 1M steps of online interactions are even enough to train off-policy online RL algorithms

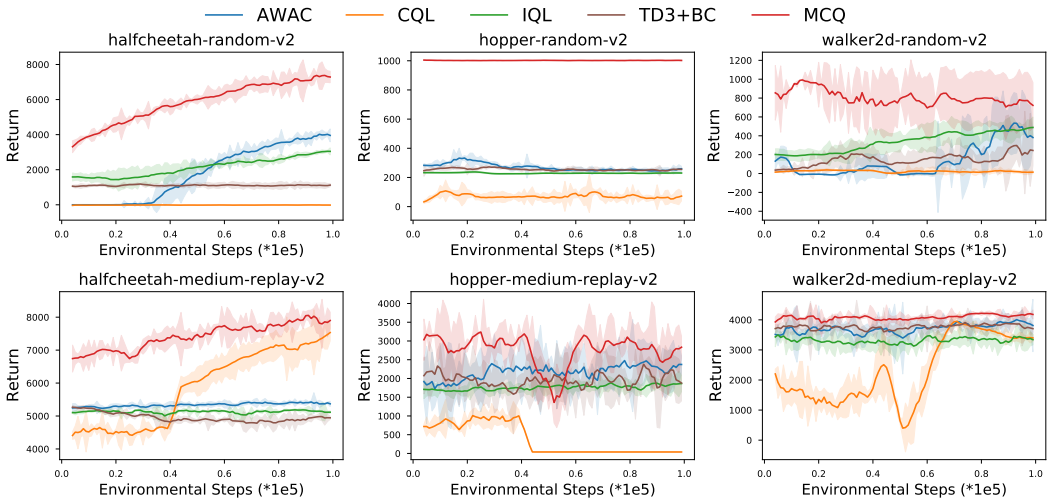


Figure 3: Offline-to-online fine-tuning results on 6 D4RL MuJoCo locomotion tasks.

from scratch to perform very well. We thus believe 100K steps is more reasonable for the online interaction. All methods are run over 4 random seeds. The results are shown in Figure 3, where the shaded region denotes the standard deviation. As expected, we observe that MCQ consistently outperforms prior offline RL methods as well as AWAC on all of the datasets, often surpassing all of them with a large margin. The mild pessimism of MCQ makes it adapt faster, or keep the offline good performance during online interactions. Other prior offline RL methods, unfortunately, fail in achieving satisfying performance during online interaction due to strict conservatism and lack of generalization ability.

5 Related Work

Model-free offline RL. Prior model-free offline RL methods are typically designed to restrict the learned policy from producing OOD actions. They usually achieve this by leveraging importance sampling [47, 54, 39, 44, 16], incorporating explicit policy constraints [34, 58, 17, 13, 11], learning latent actions [67, 3], penalizing learned value functions such that low values are assigned to unseen actions [35, 33, 41], using adaptive methods [18], and uncertainty quantification [59, 65, 4]. Another line of the methods, instead, resorts to learning without querying OOD actions [56, 8, 32]. By doing so, they constrain the learning process within the support of the dataset. Nevertheless, existing methods may induce unnecessarily over-pessimistic value functions, and their performance is largely confined by how well the behavior policy is [45, 38, 4]. That partly explains why these methods are not satisfiable when trained on non-expert datasets (e.g., random datasets). MCQ keeps milder conservatism and better generalization ability as OOD actions are *actively* trained with proper targets.

Model-based offline RL. Model-based offline RL methods, in contrast, learn the dynamics model in a supervised manner, and leverage the learned dynamics for policy optimization. Advances in this field include uncertainty quantification [46, 64, 27, 10], learning conservative value functions [63], representation learning [37, 48], constraining the learned policy with a behavior cloning loss [42], and sequential modelling [7, 23, 43]. However, there is no guarantee that the trained dynamics models are reliable, e.g., poor transitions can be generated, especially in complex high-dimensional environments [22]. Meanwhile, training dynamics models raises extra computation costs.

Offline-to-online RL. There are some efforts on accelerating online interactions with the aid of offline logged data, which is also referred to as learning from demonstration [21, 26, 49]. Offline-to-online RL, instead, aims at enhancing the well-trained offline policy via online interactions. To ensure a fast adaptation and stable policy improvement, many techniques are adopted, such as model ensemble [38], explicit policy constraints [45, 66]. Offline-to-online fine-tuning will be difficult if the trained value function or policy is overly pessimistic, which may lead to a suboptimal policy.

6 Conclusion

In this paper, we propose Mildly Conservative Q-learning (MCQ) to alleviate the over pessimism in existing offline RL methods. MCQ actively train OOD actions by constructing them proper pseudo target values following the guidance of the practical Mildly Conservative Bellman (MCB) operator. We theoretically illustrate that the policy induced by the MCB operator behaves at least as well as the behavior policy, and no erroneous overestimation will occur for the practical MCB operator. Furthermore, we extensively compare MCQ against recent strong baselines on MuJoCo locomotion tasks. Experimental results show that MCQ surpasses these baselines with a large margin on many non-expert datasets, and is also competitive with baselines on expert datasets. Moreover, we demonstrate the superior generalization capability of MCQ when transferring from offline to online. These altogether reveal that mild conservatism is critical for offline learning. We hope this work can promote the offline RL towards mild pessimism, and bring new insights into the community.

One drawback of our current algorithm lies in the need of tuning the weighting coefficient λ. However, we empirically find that $0.7 \leq \lambda < 1$ can usually induce satisfying performance. We leave the automatic tuning of λ as future work.

Acknowledgments and Disclosure of Funding

This work was supported in part by the Science and Technology Innovation 2030-Key Project under Grant 2021ZD0201404, in part by the NSF China under Grant 61872009. The authors would like to thank the anonymous reviewers for their valuable comments and advice.

References

[1] J. Achiam, D. Held, A. Tamar, and P. Abbeel. Constrained Policy Optimization. In *International Conference on Machine Learning*, 2017.

[2] R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville, and M. G. Bellemare. Deep Reinforcement Learning at the Edge of the Statistical Precipice. In *Advances in Neural Information Processing Systems*, 2021.

[3] A. Ajay, A. Kumar, P. Agrawal, S. Levine, and O. Nachum. OPAL: Offline Primitive Discovery for Accelerating Offline Reinforcement Learning. In *International Conference on Learning Representations*, 2021.

[4] C. Bai, L. Wang, Z. Yang, Z.-H. Deng, A. Garg, P. Liu, and Z. Wang. Pessimistic Bootstrapping for Uncertainty-Driven Offline Reinforcement Learning. In *International Conference on Learning Representations*, 2022.

[5] D. Bang and H. Shim. MGGAN: Solving Mode Collapse Using Manifold-Guided Training. *IEEE/CVF International Conference on Computer Vision Workshops*, pages 2347–2356, 2021.

[6] D. Bau, J.-Y. Zhu, J. Wulff, W. S. Peebles, H. Strobelt, B. Zhou, and A. Torralba. Seeing What a GAN Cannot Generate. In *IEEE/CVF International Conference on Computer Vision*, pages 4501–4510, 2019.

[7] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision Transformer: Reinforcement Learning via Sequence Modeling. *ArXiv*, abs/2106.01345, 2021.

[8] X. Chen, Z. Zhou, Z. Wang, C. Wang, Y. Wu, Q. Deng, and K. W. Ross. BAIL: Best-Action Imitation Learning for Batch Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems*, 2020.

[9] I. Csiszár and J. Körner. *Information Theory - Coding Theorems for Discrete Memoryless Systems, Second Edition*. Cambridge University Press, 2011.

[10] C. P. Diehl, T. Sievernich, M. Krüger, F. Hoffmann, and T. Bertram. UMBRELLA: Uncertainty-Aware Model-Based Offline Reinforcement Learning Leveraging Planning. *ArXiv*, abs/2111.11097, 2021.

[11] R. Fakoor, J. Mueller, P. Chaudhari, and A. Smola. Continuous Doubly Constrained Batch Reinforcement Learning. In *Advances in Neural Information Processing Systems*, 2021.

[12] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4RL: Datasets for Deep Data-Driven Reinforcement Learning. *ArXiv*, abs/2004.07219, 2020.

[13] S. Fujimoto and S. S. Gu. A Minimalist Approach to Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems*, 2021.

[14] S. Fujimoto, D. Meger, and D. Precup. Off-Policy Deep Reinforcement Learning without Exploration. In *International Conference on Machine Learning*, 2019.

[15] S. Fujimoto, H. van Hoof, and D. Meger. Addressing Function Approximation Error in Actor-Critic Methods. In *International Conference on Machine Learning*, 2018.

[16] C. Gelada and M. G. Bellemare. Off-Policy Deep Reinforcement Learning by Bootstrapping the Covariate Shift. In *AAAI Conference on Artificial Intelligence*, volume 33, 2019.

[17] S. K. S. Ghasemipour, D. Schuurmans, and S. S. Gu. EMaQ: Expected-Max Q-Learning Operator for Simple Yet Effective Offline and Online RL. In *International Conference on Machine Learning*, 2021.

[18] D. Ghosh, A. Ajay, P. Agrawal, and S. Levine. Offline RL Policies Should be Trained to be Adaptive. *ArXiv*, abs/2207.02200, 2022.

[19] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, 2014.

[20] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *International Conference on Machine Learning*, 2018.

[21] T. Hester, M. Vecerík, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, G. Dulac-Arnold, J. P. Agapiou, J. Z. Leibo, and A. Gruslys. Deep Q-learning From Demonstrations. In *AAAI Conference on Artificial Intelligence*, 2018.

[22] M. Janner, J. Fu, M. Zhang, and S. Levine. When to Trust Your Model: Model-Based Policy Optimization. In *Advances in Neural Information Processing Systems*, 2019.

[23] M. Janner, Q. Li, and S. Levine. Offline Reinforcement Learning as One Big Sequence Modeling Problem. In *Advances in Neural Information Processing Systems*, 2021.

[24] Y. Jin, Z. Yang, and Z. Wang. Is Pessimism Provably Efficient for Offline RL? In *International Conference on Machine Learning*, pages 5084–5096. PMLR, 2021.

[25] S. M. Kakade and J. Langford. Approximately Optimal Approximate Reinforcement Learning. In *International Conference on Machine Learning*, 2002.

[26] B. Kang, Z. Jie, and J. Feng. Policy Optimization with Demonstrations. In *International Conference on Machine Learning*, 2018.

[27] R. Kidambi, A. Rajeswaran, P. Netrapalli, and T. Joachims. MOReL: Model-Based Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems*, 2020.

[28] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representation*, 2015.

[29] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. *ArXiv*, abs/1312.6114, 2013.

[30] V. R. Konda and V. S. Borkar. Actor-Critic - Type Learning Algorithms for Markov Decision Processes. *SIAM J. Control. Optim.*, 38:94–123, 1999.

[31] V. R. Konda and J. N. Tsitsiklis. Actor-Critic Algorithms. In *Advances in Neural Information Processing Systems*, 1999.

[32] I. Kostrikov, A. Nair, and S. Levine. Offline Reinforcement Learning with Implicit Q-Learning. In *International Conference on Learning Representations*, 2022.

[33] I. Kostrikov, J. Tompson, R. Fergus, and O. Nachum. Offline Reinforcement Learning with Fisher Divergence Critic Regularization. In *International Conference on Machine Learning*, 2021.

[34] A. Kumar, J. Fu, G. Tucker, and S. Levine. Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction. In *Advances in Neural Information Processing Systems*, 2019.

[35] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative Q-Learning for Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems*, 2020.

[36] S. Lange, T. Gabel, and M. A. Riedmiller. Batch Reinforcement Learning. In *Reinforcement Learning*, 2012.

[37] B.-J. Lee, J. Lee, and K.-E. Kim. Representation Balancing Offline Model-based Reinforcement Learning. In *International Conference on Learning Representations*, 2021.

[38] S. Lee, Y. Seo, K. Lee, P. Abbeel, and J. Shin. Offline-to-Online Reinforcement Learning via Balanced Replay and Pessimistic Q-Ensemble. In *Conference on Robot Learning*, 2021.

[39] Y. Liu, A. Swaminathan, A. Agarwal, and E. Brunskill. Off-Policy Policy Gradient with State Distribution Correction. *ArXiv*, abs/1904.08473, 2019.

[40] X. Ma, Y. Yang, H. Hu, J. Yang, C. Zhang, Q. Zhao, B. Liang, and Q. Liu. Offline Reinforcement Learning with Value-based Episodic Memory. In *International Conference on Learning Representations*, 2022.

[41] Y. J. Ma, D. Jayaraman, and O. Bastani. Conservative Offline Distributional Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

[42] T. Matsushima, H. Furuta, Y. Matsuo, O. Nachum, and S. Gu. Deployment-Efficient Reinforcement Learning via Model-Based Offline Optimization. In *International Conference on Learning Representations*, 2021.

[43] L. Meng, M. Wen, Y. Yang, C. Le, X. Li, W. Zhang, Y. Wen, H. Zhang, J. Wang, and B. Xu. Offline Pre-trained Multi-Agent Decision Transformer: One Big Sequence Model Tackles All SMAC Tasks. *ArXiv*, abs/2112.02845, 2021.

[44] O. Nachum, Y. Chow, B. Dai, and L. Li. DualDICE: Behavior-Agnostic Estimation of Discounted Stationary Distribution Corrections. In *Advances in Neural Information Processing Systems*, 2019.

[45] A. Nair, M. Dalal, A. Gupta, and S. Levine. Accelerating Online Reinforcement Learning with Offline Datasets. *ArXiv*, abs/2006.09359, 2020.

[46] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. V. Dillon, B. Lakshminarayanan, and J. Snoek. Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift. In *Advances in Neural Information Processing Systems*, 2019.

[47] D. Precup, R. S. Sutton, and S. Dasgupta. Off-Policy Temporal Difference Learning with Function Approximation. In *International Conference on Machine Learning*, 2001.

[48] R. Rafailov, T. Yu, A. Rajeswaran, and C. Finn. Offline Reinforcement Learning from Images with Latent Space Models. In *Learning for Dynamics and Control (L4DC)*, 2021.

[49] A. Rajeswaran, V. Kumar, A. Gupta, J. Schulman, E. Todorov, and S. Levine. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. *ArXiv*, abs/1709.10087, 2018.

[50] P. Rashidinejad, B. Zhu, C. Ma, J. Jiao, and S. Russell. Bridging Offline reinforcement Learning and Imitation Learning: A Tale of Pessimism. *Advances in Neural Information Processing Systems*, 34, 2021.

[51] K. Sohn, H. Lee, and X. Yan. Learning Structured Output Representation using Deep Conditional Generative Models. In *Advances in Neural Information Processing Systems*, 2015.

[52] A. Srivastava, L. Valkov, C. Russell, M. U. Gutmann, and C. Sutton. VEEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning. In *Advances in Neural Information Processing Systems*, 2017.

[53] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[54] R. S. Sutton, A. R. Mahmood, and M. White. An Emphatic Approach to the Problem of Off-policy Temporal-Difference Learning. *Journal of Machine Learning Research*, 17:2603–2631, 2016.

[55] J. Wang, W. Li, H. Jiang, G. Zhu, S. Li, and C. Zhang. Offline Reinforcement Learning with Reverse Model-based Imagination. In *Advances in Neural Information Processing Systems*, 2021.

[56] Q. Wang, J. Xiong, L. Han, P. Sun, H. Liu, and T. Zhang. Exponentially Weighted Imitation Learning for Batched Historical Data. In *Advances in Neural Information Processing Systems*, 2018.

[57] Z. Wang, A. Novikov, K. Zolna, J. T. Springenberg, S. E. Reed, B. Shahriari, N. Siegel, J. Merel, C. Gulcehre, N. M. O. Heess, and N. de Freitas. Critic Regularized Regression. In *Advances in Neural Information Processing Systems*, 2020.

[58] Y. Wu, G. Tucker, and O. Nachum. Behavior Regularized Offline Reinforcement Learning. *ArXiv*, abs/1911.11361, 2019.

[59] Y. Wu, S. Zhai, N. Srivastava, J. M. Susskind, J. Zhang, R. Salakhutdinov, and H. Goh. Uncertainty Weighted Actor-Critic for Offline Reinforcement Learning. In *International Conference on Machine Learning*, 2021.

[60] T. Xie, C.-A. Cheng, N. Jiang, P. Mineiro, and A. Agarwal. Bellman-Consistent Pessimism for Offline Reinforcement Learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[61] S. Yang, Z. Wang, H. Zheng, Y. Feng, and M. Zhou. A Regularized Implicit Policy for Offline Reinforcement Learning. *ArXiv*, abs/2202.09673, 2022.

[62] Y. Yang, X. Ma, L. Chenghao, Z. Zheng, Q. Zhang, G. Huang, J. Yang, and Q. Zhao. Believe what you see: Implicit constraint approach for offline multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[63] T. Yu, A. Kumar, R. Rafailov, A. Rajeswaran, S. Levine, and C. Finn. COMBO: Conservative Offline Model-Based Policy Optimization. In *Advances in Neural Information Processing Systems*, 2021.

[64] T. Yu, G. Thomas, L. Yu, S. Ermon, J. Y. Zou, S. Levine, C. Finn, and T. Ma. MOPO: Model-based Offline Policy Optimization. In *Advances in Neural Information Processing Systems*, 2020.

[65] A. Zanette, M. J. Wainwright, and E. Brunskill. Provable Benefits of Actor-Critic Methods for Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems*, 2021.

[66] Y. Zhao, R. Boney, A. Ilin, J. Kannala, and J. Pajarinen. Adaptive Behavior Cloning Regularization for Stable Offline-to-Online Reinforcement Learning, 2022.

[67] W. Zhou, S. Bajracharya, and D. Held. PLAS: Latent Action Space for Offline Reinforcement Learning. In *Conference on Robot Learning*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Missing Proofs

We recall the definition of the MCB operator below.

Definition 3. *The Mildly Conservative Bellman (MCB) operator is defined as*

$$\mathcal{T}_{\text{MCB}}Q(s,a) = (\mathcal{T}_1\mathcal{T}_2)Q(s,a), \tag{15}$$

where

$$\mathcal{T}_1Q(s,a) = \begin{cases} Q(s,a), & \mu(a|s) > 0. \\ \max_{a'\sim\text{Support}(\mu(\cdot|s))} Q(s,a') - \delta, & \textit{else}. \end{cases} \tag{16}$$

$$\mathcal{T}_2Q(s,a) = \begin{cases} r(s,a) + \gamma\mathbb{E}_{s'}\left[\max_{a'\in\mathcal{A}} Q(s',a')\right], & \mu(a|s) > 0, \\ Q(s,a), & \textit{else}. \end{cases} \tag{17}$$

Proposition 6. *In the support region of the behavior policy, i.e.,* $\text{Support}(\mu)$*, the MCB operator is a* γ*-contraction operator in the* $\mathcal{L}_\infty$ *norm, and any initial* Q *function can converge to a unique fixed point by repeatedly applying* $\mathcal{T}_{\text{MCB}}$*.*

Proof. Let Q_1 and Q_2 be two arbitrary Q functions. Since $a \in \text{Support}(\mu(\cdot|s))$, then if $a' \in \text{Support}(\mu(\cdot|s'))$, we have

$$\begin{aligned}
\|\mathcal{T}_{\text{MCB}}Q_1 - \mathcal{T}_{\text{MCB}}Q_2\|_\infty &= \|\mathcal{T}_2Q_1 - \mathcal{T}_2Q_2\|_\infty \\
&= \max_{s,a}\left|\left(r(s,a) + \gamma\mathbb{E}_{s'}\left[\max_{a'\in\mathcal{A}} Q_1(s',a')\right]\right) - \left(r(s,a) + \gamma\mathbb{E}_{s'}\left[\max_{a'\in\mathcal{A}} Q_2(s',a')\right]\right)\right| \\
&= \gamma\max_{s,a}\left|\mathbb{E}_{s'}\left[\max_{a'\sim\mathcal{A}} Q_1(s',a') - \max_{a'\sim\mathcal{A}} Q_2(s',a')\right]\right| \\
&\leq \gamma\max_{s,a}\mathbb{E}_{s'}\left|\max_{a'\sim\mathcal{A}} Q_1(s',a') - \max_{a'\sim\mathcal{A}} Q_2(s',a')\right| \\
&\leq \gamma\max_{s,a}\|Q_1 - Q_2\|_\infty \\
&= \gamma\|Q_1 - Q_2\|_\infty.
\end{aligned}$$

While if $a' \notin \text{Support}(\mu(\cdot|s'))$, then $\mathcal{T}_1$ in the MCB operator will work and assign OOD actions pseudo target values. Similarly, if $a' \notin \text{Support}(\mu(\cdot|s'))$, we have

$$\begin{aligned}
&\|\mathcal{T}_{\text{MCB}}Q_1 - \mathcal{T}_{\text{MCB}}Q_2\|_\infty \\
&= \|\mathcal{T}_1\mathcal{T}_2Q_1 - \mathcal{T}_1\mathcal{T}_2Q_2\|_\infty \\
&= \max_{s,a}\left|\left(r(s,a) + \gamma\mathbb{E}_{s'}\left[\max_{a'\in\text{Support}(\mu(\cdot|s'))} Q_1(s',a') - \delta\right]\right) - \left(r(s,a) + \gamma\mathbb{E}_{s'}\left[\max_{a'\in\text{Support}(\mu(\cdot|s'))} Q_2(s',a') - \delta\right]\right)\right| \\
&= \gamma\max_{s,a}\left|\mathbb{E}_{s'}\left[\max_{a'\in\text{Support}(\mu(\cdot|s'))} Q_1(s',a') - \max_{a'\in\text{Support}(\mu(\cdot|s'))} Q_2(s',a')\right]\right| \\
&\leq \gamma\max_{s,a}\mathbb{E}_{s'}\left|\max_{a'\in\text{Support}(\mu(\cdot|s'))} Q_1(s',a') - \max_{a'\in\text{Support}(\mu(\cdot|s'))} Q_2(s',a')\right| \\
&\leq \gamma\max_{s,a}\|Q_1 - Q_2\|_\infty \\
&= \gamma\|Q_1 - Q_2\|_\infty.
\end{aligned}$$

Combining the results together, we conclude that the MCB operator is a γ-contraction in the support region of the behavior policy, which naturally leads to the conclusion that any initial Q function will converge to a unique fixed point by repeatedly applying $\mathcal{T}_{\text{MCB}}$. □

In order to show Proposition 7, we first have the following observations.

Observation 1. *In the support region of* μ*,* Q_μ *is the fixed point of the Bellman operator* $\mathcal{T}^\mu Q(s,a) := r(s,a) + \gamma\mathbb{E}_{s'}\mathbb{E}_{a'\sim\mu(\cdot|s')}[Q(s',a')]$*.*

Proof. It is well known that the Bellman operator is a γ-contraction in the $\mathcal{L}_\infty$ norm, and hence for some arbitrarily initialized Q function, it is guaranteed to converge to a unique fixed point by repeatedly applying $\mathcal{T}^\mu$. Hence, Q_μ is the Q function of the behavior policy μ in $\text{Support}(\mu)$. □

Observation 2. *In the support region of* μ*,* Q_{μ^*} *is the fixed point of the Bellman optimal operator* $\mathcal{T}Q(s,a) := r(s,a) + \gamma\mathbb{E}_{s'}\left[\max_{a'\sim\mu(\cdot|s')} Q(s',a')\right]$*.*

Proof. The proof is similar to Observation 1. Note that the resulting fixed point Q_{μ^*} is the Q function of the optimal policy in the support region of the behavior policy. In offline RL setting, we only have a static logged dataset, then Q_{μ^*} is the Q function of the optimal batch-constraint policy. □

Proposition 7 (Behave at least as well as behavior policy)**.** *Denote Q_{MCB} as the unique fixed point acquired by the MCB operator, then in $\mathrm{Support}(\mu)$ we have: $Q_\mu \leq Q_{\mathrm{MCB}} \leq Q_{\mu^*}$, where Q_μ is the Q function of the behavior policy and Q_{μ^*} is the Q function of the optimal policy in the batch.*

Proof. Denote $\mathcal{T}Q(s,a)$ as the Bellman optimal operator and $\mathcal{T}^\mu Q(s,a)$ as the Bellman operator. Since Q_{MCB} is the fixed point of the MCB operator, then if $a' \in \mathrm{Support}(\mu)$, we have

$$\begin{aligned} Q_{\mathrm{MCB}} = \mathcal{T}_{\mathrm{MCB}} Q(s,a) &= \mathcal{T}_2 Q(s,a) \\ &= r(s,a) + \gamma \mathbb{E}_{s'}\left[\max_{a' \in \mathrm{Support}(\mu(\cdot|s'))} Q(s',a')\right] \\ &= \mathcal{T}Q(s,a) = Q_{\mu^*}(s,a). \end{aligned} \tag{18}$$

Furthermore, we have

$$\begin{aligned} Q_{\mathrm{MCB}} = \mathcal{T}_{\mathrm{MCB}} Q(s,a) &= \mathcal{T}_2 Q(s,a) \\ &= r(s,a) + \gamma \mathbb{E}_{s'}\left[\max_{a' \in \mathrm{Support}(\mu(\cdot|s'))} Q(s',a')\right] \\ &\geq r(s,a) + \gamma \mathbb{E}_{s'} \mathbb{E}_{a' \in \mathrm{Support}(\mu(\cdot|s'))}[Q(s',a')] \\ &= \mathcal{T}^\mu Q(s,a) = Q_\mu(s,a). \end{aligned} \tag{19}$$

If $a' \notin \mathrm{Support}(\mu)$, we have

$$\begin{aligned} Q_{\mathrm{MCB}} = \mathcal{T}_{\mathrm{MCB}} Q(s,a) &= \mathcal{T}_1 \mathcal{T}_2 Q(s,a) \\ &= r(s,a) + \gamma \mathbb{E}_{s'}\left[\max_{a' \in \mathrm{Support}(\mu(\cdot|s'))} Q(s',a') - \delta\right] \\ &< \mathcal{T}Q(s,a) = Q_{\mu^*}(s,a). \end{aligned} \tag{20}$$

Moreover, by setting $0 < \delta \leq \mathbb{E}_{s'}\left[\max_{a' \in \mathrm{Support}(\mu(\cdot|s'))} Q(s',a') - \mathbb{E}_{a' \sim \mathrm{Support}(\mu(\cdot|s'))}[Q(s',a')]\right]$, we have

$$\begin{aligned} Q_{\mathrm{MCB}} = \mathcal{T}_{\mathrm{MCB}} Q(s,a) &= \mathcal{T}_1 \mathcal{T}_2 Q(s,a) \\ &= r(s,a) + \gamma \mathbb{E}_{s'}\left[\max_{a' \in \mathrm{Support}(\mu(\cdot|s'))} Q(s',a') - \delta\right] \\ &\geq r(s,a) + \gamma \mathbb{E}_{s'} \mathbb{E}_{a' \in \mathrm{Support}(\mu(\cdot|s'))}[Q(s',a')] \\ &= \mathcal{T}^\mu Q(s,a) = Q_\mu(s,a). \end{aligned} \tag{21}$$

Then by combining Eq. (18) and Eq. (20), the RHS holds. By combining Eq. (19) and Eq. (21), the LHS holds, which concludes the proof. □

Remark: Note that δ is a small positive number that can be arbitrarily small. As is explained in the main text (see Section 3.1), we subtract a small positive numer such that no OOD actions will be executed, considering the fact that the action is taken by following $\arg\max_{a \in \mathcal{A}} Q(s,a)$. We actively assign pseudo target values for the OOD actions, and if the allocated value is identical to the optimal Q value, actions that lie outside of the span of the behavior policy μ may be chosen and executed, which may be unsafe. By subtracting a small δ, we ensure that the OOD action training will not affect the learning for the optimal batch-constraint policy.

Proposition 8 (Milder Pessimism)**.** *Suppose there exists an explicit policy constraint offline reinforcement learning algorithm such that the KL-divergence of the learned policy $\pi_p(\cdot|s)$ and the behavior policy $\mu(\cdot|s)$ is optimized to guarantee $\max\left(\mathrm{KL}(\mu,\pi_p), \mathrm{KL}(\pi_p,\mu)\right) \leq \epsilon$, $\forall s$. Denote $\epsilon_\mu^{\pi_p} = \max_s |\mathbb{E}_{a\sim\pi_p} A^\mu(s,a)|$, then*

$$J(\pi_p) \geq J(\mu) - \frac{\sqrt{2}\gamma \epsilon_\mu^{\pi_p}}{(1-\gamma)^2}\sqrt{\epsilon}, \tag{22}$$

while for the policy π_{MCB} learned by applying the MCB operator, we have

$$J(\pi_{\mathrm{MCB}}) \geq J(\mu). \tag{23}$$

Proof. By utilizing the performance difference lemma (Lemma 6.1 in [25]), the return difference between two policies π' and π gives:

$$J(\pi') - J(\pi) = \frac{1}{1-\gamma}\sum_s d_{\pi'}(s) \sum_a \left[\pi'(a|s) A^\pi(s,a)\right], \tag{24}$$

where $d_{\pi'}(s)$ is the probability distribution underlying the states present in the static dataset $\mathcal{D}$. Furthermore, based on the Corollary 1 in [1], we have:

$$J(\pi') - J(\pi) \geq \frac{1}{1-\gamma} \sum_s d_\pi(s) \sum_a \left[\pi'(a|s) A^\pi(s,a)\right] - \frac{2\gamma \epsilon_\pi^{\pi'}}{(1-\gamma)^2} D_{\mathrm{TV}}^{d_\pi}(\pi', \pi), \tag{25}$$

where $\epsilon_\pi^{\pi'} = \max_s |\mathbb{E}_{a\sim\pi'} A^\pi(s,a)|$ and $D_{\mathrm{TV}}^{d_\pi}(p,q)$ denotes the total variation between two distributions $p(\cdot)$ and $q(\cdot)$ over the distribution d_π, i.e., $D_{\mathrm{TV}}^{d_\pi}(p,q) = \frac{1}{2}\sum_{d_\pi}\sum_x |p(x) - q(x)|$. We use $D_{\mathrm{TV}}(p,q)$ to denote the total variation between two distributions $p(\cdot)$ and $q(\cdot)$, $D_{\mathrm{TV}}(p,q) = \frac{1}{2}\sum_x |p(x) - q(x)|$.

Following similar procedure as Proposition 2 in [56], we have

$$\begin{aligned} J(\pi_p) - J(\mu) &\geq \frac{1}{1-\gamma} \sum_s d_\mu(s) \sum_a \pi_p(a|s) A^\mu(s,a) - \frac{2\gamma \epsilon_\mu^{\pi_p}}{(1-\gamma)^2} D_{\mathrm{TV}}^{d_\mu}(\pi_p, \mu) \\ &\geq 0 - \frac{2\gamma \epsilon_\mu^{\pi_p}}{(1-\gamma)^2} D_{\mathrm{TV}}^{d_\mu}(\pi_p, \mu) \\ &= -\frac{2\gamma \epsilon_\mu^{\pi_p}}{(1-\gamma)^2} D_{\mathrm{TV}}^{d_\mu}(\pi_p, \mu), \end{aligned} \tag{26}$$

Then, we have

$$\begin{aligned} J(\pi_p) - J(\mu) &\geq -\frac{2\gamma \epsilon_\mu^{\pi_p}}{(1-\gamma)^2} D_{\mathrm{TV}}^{d_\mu}(\pi_p, \mu) \\ &= -\frac{2\gamma \epsilon_\mu^{\pi_p}}{(1-\gamma)^2} \sum_s d_\mu(s) \left[D_{\mathrm{TV}}(\pi_p(\cdot|s), \mu(\cdot|s))\right] \\ &\geq -\frac{\sqrt{2}\gamma \epsilon_\mu^{\pi_p}}{(1-\gamma)^2} \sum_s d_\mu(s) \min\left(\sqrt{\mathrm{KL}(\mu(\cdot|s), \pi_p(\cdot|s))}, \sqrt{\mathrm{KL}(\pi_p(\cdot|s), \mu(\cdot|s))}\right) \\ &\geq -\frac{\sqrt{2}\gamma \epsilon_\mu^{\pi_p}}{(1-\gamma)^2} \sum_s d_\mu(s) \max\left(\sqrt{\mathrm{KL}(\mu(\cdot|s), \pi_p(\cdot|s))}, \sqrt{\mathrm{KL}(\pi_p(\cdot|s), \mu(\cdot|s))}\right) \\ &\geq -\frac{\sqrt{2}\gamma \epsilon_\mu^{\pi_p}}{(1-\gamma)^2} \sqrt{\epsilon}. \end{aligned} \tag{27}$$

The third line above uses the Pinsker-Csiszatr inequality [9], and the fourth line holds due to the fact that $\max(a,b) \geq \min(a,b)$, and the final line holds due to the assumption we make, i.e., the KL-divergence between the learned policy and the behavior policy is optimized to guarantee $\max\left(\mathrm{KL}(\mu, \pi_p), \mathrm{KL}(\pi_p, \mu)\right) \leq \epsilon$, $\forall s$. We hence conclude that the explicit policy constraint RL algorithms induce a safe policy improvement, i.e., $J(\pi_p) \geq J(\mu) - \mathcal{O}\left(\frac{1}{(1-\gamma)^2}\right)$.

For the policy induced by the MCB operator, however, we have

$$\begin{aligned} J(\pi_{\mathrm{MCB}}) &= \mathbb{E}_{s\sim\mathcal{D}}\left[V_{\pi_{\mathrm{MCB}}}(s)\right] \\ &= \mathbb{E}_{s\sim\mathcal{D}} \mathbb{E}_{a\sim\pi_{\mathrm{MCB}}}\left[Q_{\mathrm{MCB}}(s,a)\right] \\ &= \mathbb{E}_{s\sim\mathcal{D}} \mathbb{E}_{a\sim\mathrm{Support}(\mu(\cdot|s))}\left[Q_{\mathrm{MCB}}(s,a)\right] \\ &\geq \mathbb{E}_{s\sim\mathcal{D}} \mathbb{E}_{a\sim\mathrm{Support}(\mu(\cdot|s))}\left[Q_\mu(s,a)\right] \quad \text{(By using Proposition 7)} \\ &= J(\mu). \end{aligned} \tag{28}$$

The third line holds as the policy induced by the MCB operator will not execute actions that lie outside of the support region of behavior policy. That allows us to conclude the proof. □

Remark: Note that in Theorem 3.6 of [35], CQL also guarantees a safe policy improvement, i.e., $J(\pi_{\mathrm{CQL}}) \geq J(\mu) - \mathcal{O}\left(\frac{1}{(1-\gamma)^2}\right)$, which is consistent with the lower bound of the explicit policy constraint methods. These methods are often overly pessimistic, while the policy learned by applying the MCB operator can improve the policy with a tighter lower bound.

Proposition 9. *Proposition 6 still holds for the practical MCB operator.*

Proof. The practical MCB operator only modifies $\mathcal{T}_1$, i.e., the target value for the OOD actions. We only discuss $a' \notin \mathrm{Support}(\mu(\cdot|s'))$ here because for $a' \in \mathrm{Support}(\mu(\cdot|s'))$, the proof is identical as Proposition 6. For

$a' \notin \text{Support}(\mu(\cdot|s'))$, let Q_1 and Q_2 be two arbitrary Q functions, then

$$
\begin{aligned}
&\|\hat{\mathcal{T}}_{\text{MCB}} Q_1 - \hat{\mathcal{T}}_{\text{MCB}} Q_2\|_\infty \\
&= \|\hat{\mathcal{T}}_1 \mathcal{T}_2 Q_1 - \hat{\mathcal{T}}_1 \mathcal{T}_2 Q_2\|_\infty \\
&= \max_{s,a} \Bigg| \left(r(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{D}} \mathbb{E}_{\{a'_i\}^N \sim \text{Support}(\hat{\mu})} \left[\max_{a' \in \{a'_i\}^N} Q_1(s', a') \right] \right) \\
&\qquad - \left(r(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{D}} \mathbb{E}_{\{a'_i\}^N \sim \text{Support}(\hat{\mu})} \left[\max_{a' \in \{a'_i\}^N} Q_2(s', a') \right] \right) \Bigg| \\
&= \gamma \max_{s,a} \left| \mathbb{E}_{s' \sim \mathcal{D}} \mathbb{E}_{\{a'_i\}^N \sim \text{Support}(\hat{\mu})} \left[\max_{a' \in \{a'_i\}^N} Q_1(s', a') - \max_{a' \in \{a'_i\}^N} Q_2(s', a') \right] \right| \\
&\leq \gamma \max_{s,a} \mathbb{E}_{s' \sim \mathcal{D}} \mathbb{E}_{\{a'_i\}^N \sim \text{Support}(\hat{\mu})} \left| \max_{a' \sim \{a'_i\}^N} Q_1(s', a') - \max_{a' \sim \{a'_i\}^N} Q_2(s', a') \right| \\
&\leq \gamma \max_{s,a} \mathbb{E}_{s' \sim \mathcal{D}} \mathbb{E}_{\{a'_i\}^N} \|Q_1 - Q_2\|_\infty \\
&= \gamma \|Q_1 - Q_2\|_\infty,
\end{aligned}
$$

where we use a fact that $\max_{a' \in \{a'_i\}^N} Q_1(s', a') - \max_{a' \in \{a'_i\}^N} Q_2(s', a') \leq \|Q_1 - Q_2\|_\infty$. Its proof is quite straightforward. Let $\hat{a} = \arg\max_{\{a'_i\}^N} Q_1(s', a'_i)$, then

$$
\begin{aligned}
\max_{a' \in \{a'_i\}^N} Q_1(s', a') - \max_{a' \in \{a'_i\}^N} Q_2(s', a') &= Q_1(s', \hat{a}) - \max_{a' \in \{a'_i\}^N} Q_2(s', a') \\
&\leq Q_1(s', \hat{a}) - Q_2(s', \hat{a}) \leq \|Q_1 - Q_2\|_\infty.
\end{aligned}
$$

Therefore, we conclude that Proposition 6 still holds for the practical MCB operator. □

Proposition 10 (No erroneous overestimation will occur)**.** *Assuming that* $\sup_s D_{\text{TV}}(\hat{\mu}(\cdot|s) \parallel \mu(\cdot|s)) \leq \epsilon < \frac{1}{2}$, *we have*

$$
\mathbb{E}_{\{a'_i\}^N \sim \hat{\mu}(\cdot|s)} \left[\max_{a' \in \{a'_i\}^N} Q(s, a') \right] \leq \max_{a' \in \text{Support}(\mu(\cdot|s))} Q(s, a') + (1 - (1 - 2\epsilon)^N) \frac{r_{\max}}{1 - \gamma}.
$$

Proof. The total variation divergence $D_{TV}(p \| q) = \frac{1}{2} \sum_x |p(x) - q(x)|$ for some distributions $p(\cdot)$ and $q(\cdot)$, then we have

$$
\begin{aligned}
1 > 2\epsilon \geq 2 \sup_s D_{\text{TV}}(\hat{\mu}(\cdot|s) \parallel \mu(\cdot|s)) &\geq \sum_a |\hat{\mu}(a|s) - \mu(a|s)| \\
&= \sum_{a \in \text{Support}(\mu(\cdot|s))} |\hat{\mu}(a|s) - \mu(a|s)| + \sum_{a \notin \text{Support}(\mu(\cdot|s))} |\hat{\mu}(a|s) - \mu(a|s)| \\
&\geq \sum_{a \notin \text{Support}(\mu(\cdot|s))} \hat{\mu}(a|s),
\end{aligned} \tag{29}
$$

where we use the fact that $\mu(a|s) = 0$ if $a \notin \text{Support}(\mu(\cdot|s))$. Denote $Q_{\max}$ as the maximum value of the OOD actions, and $Q_{\max} \leq \frac{r_{\max}}{1-\gamma}$. Thus, we have

$$
\begin{aligned}
&\mathbb{E}_{\{a'_i\}^N \sim \hat{\mu}(\cdot|s)} \left[\max_{a' \in \{a'_i\}^N} Q(s, a') \right] \\
&\leq \mathbb{P}\left(\bigcap_i \{a'_i \in \text{Support}(\mu(\cdot|s))\} \right) \max_{a' \in \text{Support}(\mu(\cdot|s))} Q(s, a') + \mathbb{P}\left(\bigcup_i \{a'_i \notin \text{Support}(\mu(\cdot|s))\} \right) Q_{\max} \\
&\leq \max_{a' \in \text{Support}(\mu(\cdot|s))} Q(s, a') + \left(1 - \mathbb{P}(a'_1 \in \text{Support}(\mu(\cdot|s)))^N \right) \frac{r_{\max}}{1 - \gamma} \\
&\leq \max_{a' \in \text{Support}(\mu(\cdot|s))} Q(s, a') + \left(1 - (1 - 2\epsilon)^N \right) \frac{r_{\max}}{1 - \gamma}.
\end{aligned}
$$

□

Remark: Note that we empirically fit a behavior policy $\hat{\mu}(\cdot|s)$ as we usually do not have prior knowledge about the true behavior policy μ. The conclusion says that for the practical MCB operator and empirical behavior

policy $\hat{\mu}(\cdot|s)$, even if OOD actions are sampled from the $\hat{\mu}(\cdot|s)$, the extrapolation error is under the scale of $\left(1-(1-2\epsilon)^N\right)\frac{r_{\max}}{1-\gamma}$, indicating the fact that no severe overestimation will occur. If the empirical behavior policy well fits the true behavior policy, then the error term will be arbitrarily small and the value estimate will well approximate the maximum Q value in the batch. We assume that the total variation divergence between $\hat{\mu}(\cdot|s)$ and μ is bounded by $\frac{1}{2}$, which is easy to satisfy in practice, e.g., fit the empirical behavior policy via supervised learning.

B Deterministic MCQ

As mentioned in the main text, the practical MCB operator is pluggable and can be combined with wide range of off-policy online RL methods. In this section, we present a deterministic version of MCQ by incorporating the practical MCB operator with TD3 [15].

Similarly, we train a CVAE G_ω which is made up of an encoder $E_\xi(s,a)$ and a decoder $D_\psi(s,z)$ as the generative model, which is optimized by:

$$\mathcal{L}_{\text{CVAE}} = \mathbb{E}_{(s,a)\sim\mathcal{D},z\sim E_\xi(s,a)}\left[(a-D_\psi(s,z))^2 + \text{KL}\left(E_\xi(s,a),\mathcal{N}(0,\mathbf{I})\right)\right]. \tag{30}$$

The objective function for the critics gives:

$$\mathcal{L}_{\text{critic}} = \lambda\mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[(Q_{\theta_i}(s,a)-y)^2\right] + (1-\lambda)\mathbb{E}_{s^{\text{in}}\sim\mathcal{D}}\left[(Q_{\theta_i}(s^{\text{in}},\pi(s^{\text{in}}))-y')^2\right], \tag{31}$$

where $\theta_i, i=1,2$, is the parameter of the critic network, and the target value for the in-distribution action gives:

$$y = r(s,a) + \gamma\left[\min_{i=1,2} Q_{\theta_i'}(s',\pi_{\phi'}(s'))\right], \tag{32}$$

where $\theta_i', i=1,2$ is the parameter of the lagging target network, and ϕ' is the parameter of the actor network. For OOD actions, the target value gives:

$$y' = \min_{j=1,2}\mathbb{E}_{\{a_i'\}^N\sim\hat{\mu}}\left[\max_{a'\sim\{a_i'\}^N} Q_{\theta_j}(s^{\text{in}},a')\right]. \tag{33}$$

The actor is then optimized by $\max_\phi \mathbb{E}_{s\sim\mathcal{D}}\left[Q_{\theta_1}(s,\pi_\phi(s))\right]$. The full procedure of the deterministic MCQ is presented in Algorithm 2.

Algorithm 2 Deterministic version of Mildly Conservative Q-learning (MCQ)

Initialize CVAE G_ω, critic networks $Q_{\theta_1}, Q_{\theta_2}$ and actor network π_ϕ with random parameters
Initialize target networks $\theta_1' \leftarrow \theta_1, \theta_2' \leftarrow \theta_2, \phi' \leftarrow \phi$ and offline replay buffer $\mathcal{D}$.
for t = 1 to T **do**
  Sample a mini-batch $B = \{s,a,r,s',d\}$ from $\mathcal{D}$, where d is the done flag
  Train CVAE via minimizing Eq. (30)
  Get target value: $y = r(s,a) + \gamma\left[\min_{i=1,2} Q_{\theta_i'}(s',\pi_{\phi'}(s')\right]$
  Set $s^{\text{in}} = \{s,s'\}$ and compute the target value for the OOD actions via Eq. (33)
  Update critic θ_i with gradient descent via minimizing Eq. (31)
  Update ϕ by policy gradient: $\nabla_\phi J(\phi) = |B|^{-1}\sum \nabla_a Q_{\theta_1}(s,a)|_{a=\pi_\phi(s)}\nabla_\phi\pi_\phi(s)$
  Update target networks: $\phi' \leftarrow \tau\phi + (1-\tau)\phi', \theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i', i=1,2$
end for

Next, we empirically examine the performance of the deterministic MCQ by conducting experiments on D4RL [12] MuJoCo locomotion tasks. As we want to focus on the benefits of MCQ on non-expert datasets, and considering the fact that all baselines can achieve good performance on expert datasets, we only report the performance comparison on non-expert datasets (random, medium, medium-replay, and medium-expert). The results are summarized in Table 2. For the deterministic MCQ, we adopt identical λ and N as vanilla MCQ (see Table 5) except that we use $\lambda = 0.4$ on hopper-medium-expert-v2. We observe that the deterministic MCQ has a good performance on many non-expert datasets, which is consistent with vanilla MCQ (which is built upon SAC). The deterministic MCQ has an average score of 68.4 on non-expert datasets and the vanilla MCQ has an average score of 72.8. Note that it is not suitable to compare BC against the deterministic MCQ (though deterministic MCQ still outperforms BC on all of the datasets), since the actor in our BC is not deterministic.

All these validate our claim that the practical MCB operator is pluggable and can be combined with any off-policy online RL algorithm to make it work in the offline setting. Note that it is also very interesting to report all the numerical results based on [2], which we omit here.

Table 2: Normalized average score comparison of MCQ against baseline methods on D4RL benchmarks over the final 10 evaluations. 0 corresponds to a random policy and 100 corresponds to an expert policy. The experiments are run on MuJoCo "-v2" datasets over 4 random seeds. r = random, m = medium, m-r = medium-replay, m-e = medium-expert, e = expert. MCQ (TD3) denotes the deterministic version of MCQ that built upon TD3.

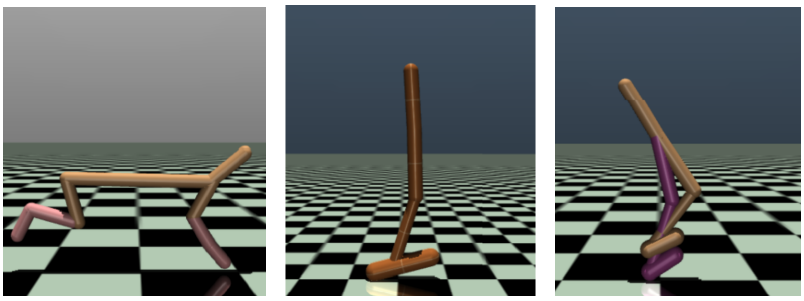
Task Name	BC	SAC	CQL	UWAC	TD3+BC	IQL	MCQ (TD3)
halfcheetah-r	2.2±0.0	29.7 ±1.4	17.5±1.5	2.3±0.0	11.0±1.1	13.1±1.3	23.6 ±0.8
hopper-r	3.7±0.6	9.9±1.5	7.9±0.4	2.7±0.3	8.5±0.6	7.9±0.2	31.0 ±1.7
walker2d-r	1.3±0.1	0.9±0.8	5.1±1.3	2.0±0.4	1.6±1.7	5.4±1.2	10.3 ±6.8
halfcheetah-m	43.2±0.6	55.2 ±27.8	47.0±0.5	42.2±0.4	48.3±0.3	47.4±0.2	58.3 ±1.3
hopper-m	54.1±3.8	0.8±0.0	53.0±28.5	50.9±4.4	59.3±4.2	66.2 ±5.7	73.6 ±10.3
walker2d-m	70.9±11.0	-0.3±0.2	73.3±17.7	75.4±3.0	83.7 ±2.1	78.3±8.7	88.4 ±1.3
halfcheetah-m-r	37.6±2.1	0.8±1.0	45.5 ±0.7	35.9±3.7	44.6±0.5	44.2±1.2	51.5 ±0.2
hopper-m-r	16.6±4.8	7.4±0.5	88.7±12.9	25.3±1.7	60.9±18.8	94.7 ±8.6	99.5 ±1.7
walker2d-m-r	20.3±9.8	-0.4±0.3	81.8 ±2.7	23.6±6.9	81.8 ±5.5	73.8±7.1	83.3 ±1.9
halfcheetah-m-e	44.0±1.6	28.4±19.4	75.6±25.7	42.7±0.3	90.7 ±4.3	86.7 ±5.3	85.4 ±3.4
hopper-m-e	53.9±4.7	0.7±0.0	105.6 ±12.9	44.9±8.1	98.0±9.4	91.5±14.3	106.1 ±2.3
walker2d-m-e	90.1±13.2	1.9±3.9	107.9 ±1.6	96.5±9.1	110.1 ±0.5	109.6 ±1.0	110.3 ±0.1
Average	36.5	11.3	59.1	37.0	58.2	59.9	68.4

C Experimental Details and Parameter Setup

In this section, we first give a brief introduction to the D4RL benchmarks [12]. Then we present our implementation details and experimental details.

C.1 D4RL Benchmarks

In the paper, we evaluate MCQ on the D4RL MuJoCo-Gym domain, which contains three environments (halfcheetah, hopper, walker2d), and five types of datasets (random, medium, medium-replay, medium-expert, expert). **Random** datasets are gathered by a random policy. **Medium** datasets contain experience from an early-stopped SAC policy. **Medium-replay** datasets are collected during the training process of the "medium" SAC policy. **Medium-expert** datasets are formed by combining the suboptimal samples and the expert samples. **Expert** datasets are made up of expert trajectories. The illustration of MuJoCo tasks is shown in Figure 4.



(a) HalfCheetah (b) Hopper (c) Walker2d

Figure 4: MuJoCo datasets. We conduct experiments on halfcheetah, hopper, and walker2d tasks.

D4RL offers a metric, normalized score, to evaluate the performance of the offline RL algorithm, which is calculated by:

$$\text{score} = \frac{\text{average return} - \text{return of the random policy}}{\text{return of the expert policy} - \text{return of the random policy}} \times 100.$$

Note that if the normalized score equals to 0, that indicates that the learned policy has a similar performance as the random policy, while 100 corresponds to an expert policy. In D4RL, different types of datasets (e.g., random,

medium, expert) share the identical reference minimum score and reference maximum score. We summarize the reference score for each environment in Table 3.

Table 3: The referenced min score and max score for the MuJoCo dataset in D4RL.

Domain	Task Name	Reference Min Score	Reference Max Score
MuJoCo	halfcheetah	-280.18	12135.0
MuJoCo	hopper	-20.27	3234.3
MuJoCo	Walker2d	1.63	4592.3

C.2 Implementation Details

Baselines. We conduct experiments on the recently released MuJoCo "-v2" datasets. We adopt behavior cloning (BC), Soft Actor-Critic (SAC) [20], and recently proposed offline RL methods, CQL [35], UWAC [59], TD3+BC [13], and IQL [32] as baseline methods. The results of CQL are obtained by running the official codebase (https://github.com/aviralkumar2907/CQL). For UWAC, we rerun it on MuJoCo "-v2" datasets using the official codebase (https://github.com/apple/ml-uwac) since the results reported in its original paper are not obtained on "-v2" datasets. For TD3+BC, we actually can reproduce its results, and thus we take the results reported in its original paper (Table 7 in the appendix) directly. For IQL, we take its results on medium, medium-replay, and medium-expert from its original paper directly. We run IQL on random and expert datasets with its official codebase (https://github.com/ikostrikov/implicit_q_learning). All methods are run over 4 different random seeds and the normalized average scores are reported in the main text.

MCQ implementation details. We build the practical MCB operator upon Soft Actor-Critic (SAC), and modify its critic objective function by adding an auxiliary loss such that OOD actions can be trained. For some datasets, we subtract the pseudo target value a small positive number (as in Definition 1) since we find it behaves better. We also stop overly estimated value from backpropagating for some of the datasets. We additionally train a generative model, CVAE. For each training epoch, we sample a mini-batch from the logged dataset, and train the CVAE. Next, we leverage the critic networks for estimating the Q-values and their corresponding target values upon in-distribution state-action pairs. Furthermore, we sample N actions from the CVAE based on the current state and the next state, and sample N actions from the learned policy. We then construct the pseudo target values for OOD actions. Finally, we optimize the learned policy with the standard way SAC does. We detail the default parameter setup for SAC and CVAE in Table 4.

We set the number of sampled actions $N = 10$ by default as we experimentally find that MCQ is insensitive to N in a wide range. Across all of our experiments, we tune the weighting coefficient λ across $\{0.3, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95\}$ with grid search. We summarize the hyperparameters we use for running the MuJoCo datasets in Table 5, where we find that generally $0.7 \leq \lambda < 1$ is suitable for most of the datasets. For some datasets with narrow distributions, e.g., expert datasets and medium-expert datasets, a comparatively small λ is better because actions sampled from the learned policy are more likely to lie in the OOD region in these tasks, and larger weights are expected to be assigned to OOD actions training loss. Again, as we have explained in the main text, one ought not to use too small λ (and absolutely cannot set $\lambda = 0$) as our first priority is always training the in-distribution data in a good manner.

Offline-to-Online details. For offline-to-online experiments, we directly put the online samples into the offline replay buffer after offline initialization. Both offline transitions and online transitions are sampled equally during online interaction and training. The hyperparameters of all methods are kept unchanged on both offline stage and online stage. The results of baseline methods are acquired by running their official codebases, i.e., CQL (https://github.com/aviralkumar2907/CQL), TD3+BC (https://github.com/sfujim/TD3_BC), IQL (https://github.com/ikostrikov/implicit_q_learning), AWAC (https://github.com/vitchyr/rlkit). For those involve normalization over states, we normalize the online samples with the mean and standard deviation calculated based on the offline samples. All methods are run over 4 different random seeds. We choose a subset of tasks for offline-to-online fine-tuning different from IQL and AWAC to ensure that our empirical experiments on offline-to-online fine-tuning are consistent to the offline experiments on MuJoCo datasets.

D Comparison of MCQ against More Baselines

In this section, we extensively compare MCQ against severe recent strong offline RL methods on D4RL MuJoCo datasets, including BCQ [14], BEAR [34], MOPO [64], Decision Transformer (DT) [7], CDC [11], and PBRL [4]. BCQ aims at learning a batch-constraint policy and it also involves a generative model, i.e., CVAE. BEAR lies in the category of explicit policy constraint offline RL methods. MOPO is a well-known model-based offline RL algorithm that leverages the learned dynamics models for uncertainty quantification. Decision Transformer (DT) is also a recent model-based offline RL method, and it relies on the transformer for sequential modeling.

Table 4: Hyperparameters setup for MCQ.

Hyperparameter	**Value**
SAC	
Actor network	$(400, 400)$
Critic network	$(400, 400)$
Batch size	256
Critic learning rate	3×10^{-4}
Actor learning rate	3×10^{-4}
Optimizer	Adam [28]
Discount factor	0.99
Reward scale	1
Maximum log std	2
Minimum log std	-20
Use automatic entropy tuning	Yes
Target update rate	5×10^{-3}
CVAE	
Encoder hidden dimension	750
Decoder hidden dimension	750
Hidden layers	2
CVAE learning rate	1×10^{-3}
Batch size	256
Latent dimension	$2\times$ action dimension

Table 5: Detailed hyperparameters used in MCQ, where we conduct experiments on D4RL MuJoCo-Gym "-v2" datasets.

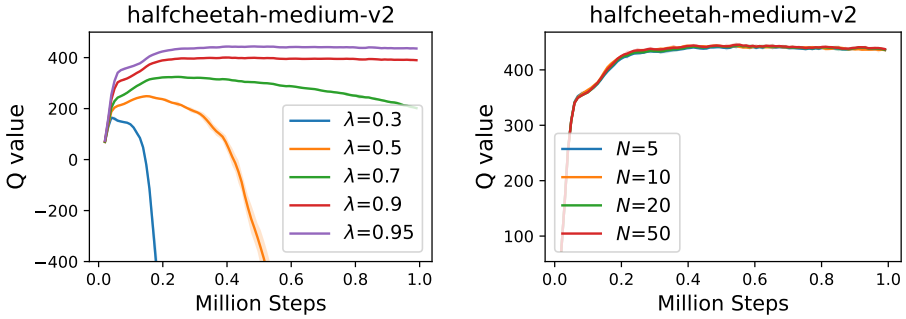
Task Name	weighting coefficient λ	number of sampled actions N
halfcheetah-random-v2	0.95	10
hopper-random-v2	0.6	10
walker2d-random-v2	0.8	10
halfcheetah-medium-v2	0.95	10
hopper-medium-v2	0.7	10
walker2d-medium-v2	0.9	10
halfcheetah-medium-replay-v2	0.95	10
hopper-medium-replay-v2	0.9	10
walker2d-medium-replay-v2	0.9	10
halfcheetah-medium-expert-v2	0.7	10
hopper-medium-expert-v2	0.5	10
walker2d-medium-expert-v2	0.8	10
halfcheetah-expert-v2	0.5	10
hopper-expert-v2	0.5	10
walker2d-expert-v2	0.3	10

CDC adds critic regularization for the critics and the explicit policy constraint for the actor. PBRL is an ensemble-based offline RL method, which also quantifies the estimation uncertainty. We conduct experiments on D4RL [12] MuJoCo locomotion datasets. Note that the results of BCQ is obtained by running its official codebase (https://github.com/sfujim/BCQ), and the results of BEAR, MOPO, and PBRL are taken directly from [4]. The results of Decision Transformer are taken from [13] (Table 7 in the appendix). Since CDC does not report standard deviation in its original paper, we omit them and directly report its mean performance. We summarize the normalized average score of each method in Table 6.

As shown, MCQ outperforms these baseline methods on most of the non-expert datasets, often by a large margin, and is competitive with them on expert datasets. Note that RBRL also leverages OOD sampling in its formulation, The main difference between MCQ and PBRL lies in the fact that PBRL adopts OOD sampling for penalizing the estimation uncertainty while we construct pseudo target values for OOD actions. Moreover, PBRL utilizes critic ensemble for uncertainty quantification while MCQ only adopts double critics. MCQ is much more computationally efficient than PBRL. One can also find that MCQ surpasses PBRL on many datasets, especially on non-expert datasets.

Table 6: Normalized average score comparison of MCQ against baseline methods on D4RL benchmarks. 0 corresponds to a random policy and 100 corresponds to an expert policy. The experiments are run on MuJoCo "-v2" datasets over 4 random seeds. r = random, m = medium, m-r = medium-replay, m-e = medium-expert, e = expert.

Task Name	BCQ	BEAR	MOPO	DT	CDC	PBRL	MCQ (ours)
halfcheetah-r	2.2±0.0	2.3±0.0	35.9 ±2.9	-	27.4	11.0±5.8	28.5 ±0.6
hopper-r	7.8±0.6	3.9±2.3	16.7±12.2	-	14.8	26.8±9.3	31.8 ±0.5
walker2d-r	4.9±0.1	12.8±10.2	4.2±5.7	-	7.2	8.1±4.4	17.0 ±3.0
halfcheetah-m	46.6±0.4	43.0±0.2	73.1 ±2.4	42.6±0.1	46.1	57.9±1.5	64.3 ±0.2
hopper-m	59.4±8.3	51.8±4.0	38.3±34.9	67.6±1.0	60.4	75.3 ±31.2	78.4 ±4.3
walker2d-m	71.8±7.2	-0.2±0.1	41.2±30.8	74.0±1.4	82.1	89.6 ±0.7	91.0 ±0.4
halfcheetah-m-r	42.2±0.9	36.3±3.1	69.2 ±1.1	36.6±0.8	44.7	45.1±8.0	56.8 ±0.6
hopper-m-r	60.9±14.7	52.2±19.3	32.7±9.4	82.7±7.0	55.4	100.6 ±1.0	101.6 ±0.8
walker2d-m-r	57.0±9.6	7.0±7.8	73.7±9.4	66.6±3.0	23.0	77.7±14.5	91.3 ±5.7
halfcheetah-m-e	95.4 ±2.0	46.0±4.7	70.3±21.9	86.8±1.3	59.6	92.3 ±1.1	87.5±1.3
hopper-m-e	106.9±5.0	50.6±25.3	60.6±32.5	107.6±1.8	86.9	110.8 ±0.8	111.2 ±0.1
walker2d-m-e	107.7 ±3.8	22.1±44.9	77.4±27.9	108.1 ±0.2	70.9	110.1 ±0.3	114.2 ±0.7
Average Above	55.2	27.3	49.4	-	48.2	67.1	72.8
halfcheetah-e	89.9±9.6	92.7 ±0.6	81.3±21.8	-	82.1	92.4 ±1.7	96.2 ±0.4
hopper-e	109.0 ±4.0	54.6±21.0	62.5±29.0	-	102.8	110.5±0.4	111.4 ±0.4
walker2d-e	106.3 ±5.0	106.6 ±6.8	62.4±3.2	-	87.5	108.3 ±0.3	107.2 ±1.1
Total Average	64.5	38.8	53.3	-	56.7	74.4	79.2



(a) Q value w.r.t. λ

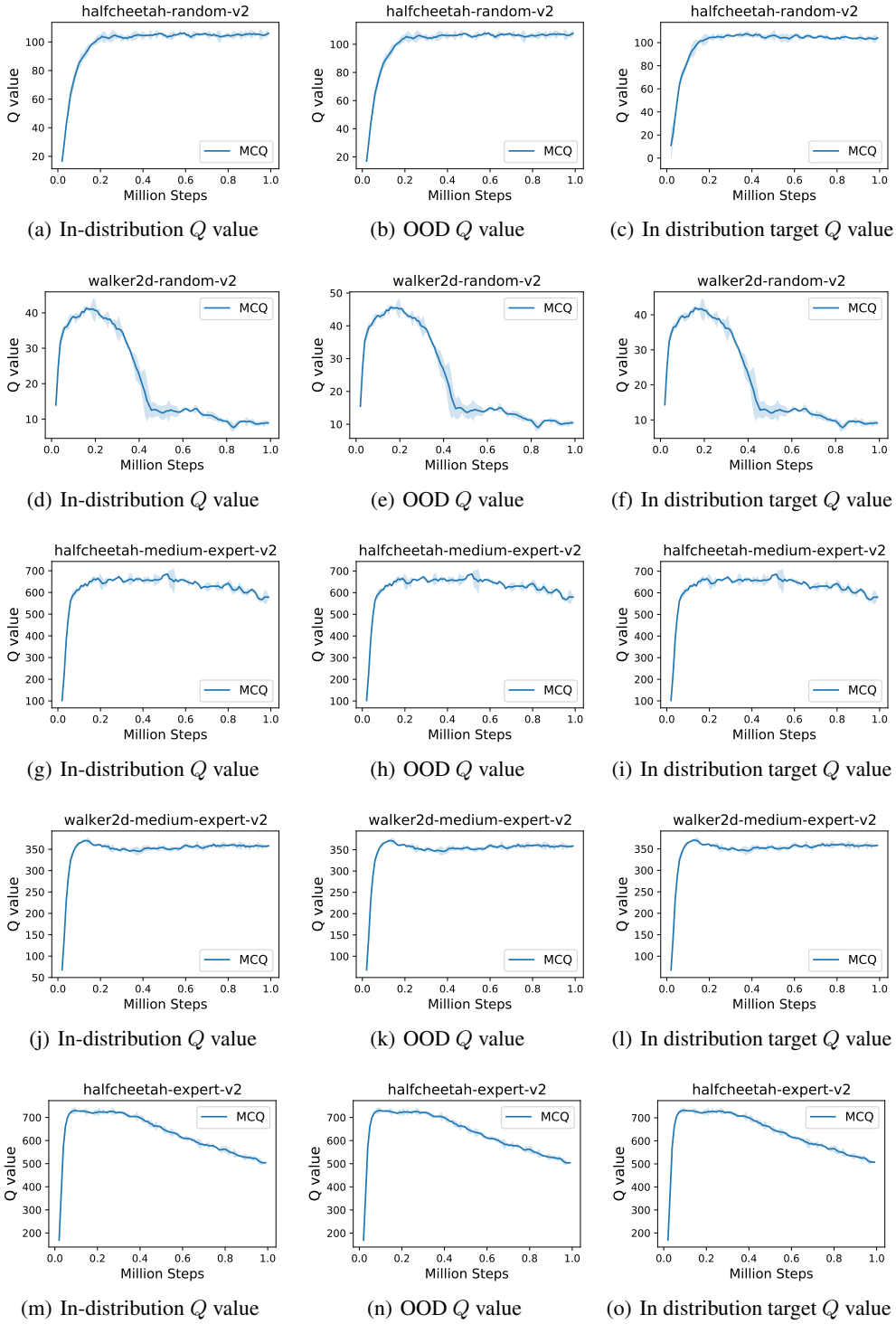
(b) Q value w.r.t. N

Figure 5: Missing value estimation on halfcheetah-medium-v2.

E Wider Evidence on Value Estimation of MCQ

We first want to include the Q-value estimation with respect to the λ and N on halfcheetah-medium-v2, which we omit in the main text due to the space limit. The results are shown in Figure 5, where we can observe that the results resemble those on hopper-medium-replay-v2, e.g., the Q value estimate collapses with smaller λ and is comparatively insensitive to N in a wide range.

Furthermore, we include more evidence that MCQ induces a good and stable value estimation in Figure 6 across more datasets with the parameters reported in Section C.2. We estimate the in-distribution Q-value by $\mathbb{E}_{s,a\sim\mathcal{D}}\mathbb{E}_{i=1,2}\left[Q_{\theta_i}(s,a)\right]$, and estimate the Q-value for the OOD actions via $\mathbb{E}_{s\sim\mathcal{D},a\sim\pi(\cdot|s)}\mathbb{E}_{i=1,2}\left[Q_{\theta_i}(s,a)\right]$, and the target value estimates for the in-distribution samples are give by the standard way of calculating target value in SAC. We find that on all of the datasets, MCQ has a good value estimation upon both in-distribution samples and OOD samples. The Q estimation curve is quite smooth and stable on most of them. Based on these observations, we can conclude that MCQ guarantees a good value estimation and will not incur severe overestimation.



(a) In-distribution Q value (b) OOD Q value (c) In distribution target Q value

(d) In-distribution Q value (e) OOD Q value (f) In distribution target Q value

(g) In-distribution Q value (h) OOD Q value (i) In distribution target Q value

(j) In-distribution Q value (k) OOD Q value (l) In distribution target Q value

(m) In-distribution Q value (n) OOD Q value (o) In distribution target Q value

Figure 6: Q value estimation of MCQ on different types of datasets, where Q estimates upon in-dataset samples and OOD samples, target Q value for the in-dataset samples are depicted. The shaded region captures the standard deviation.

F Computation Cost and Compute Infrastructure

As for the computational cost, MCQ generally consumes 8 to 12 hours to train on MuJoCo tasks due to the cost of OOD sampling. In contrast, CQL takes around 12 to 15 hours. We compare MCQ against CQL on halfcheetah-medium-v2 task in terms of runtime per epoch (1K gradient steps) and GPU memory during training. We conduct the experiments on a single RTX3090 GPU. We detail the computation cost comparison of MCQ against CQL in Table 7. As shown, MCQ is computationally more efficient than CQL and can achieve better performance with lower computational costs.

Table 7: Computation cost comparison of MCQ and CQL.

Algorithm	**Runtime (s/epoch)**	**GPU Memory (GB)**
CQL	38.2	1.4
MCQ	30.3	1.2

In Table 8, we list the compute infrastructure that we use to run all of the baseline algorithms and MCQ experiments.

Table 8: Compute infrastructure.

CPU	**GPU**	**Memory**
AMD EPYC 7452	RTX3090×8	288GB

G Omitted Background for VAE

In this section, we give a brief introduction to the VAE, and CVAE [29, 51]. Given a dataset $\{x_i\}_{i=1}^N$, the VAE learns a generative model $p_\theta(X) = \prod_{i=1}^N p_\theta(x_i)$, where $X = \{x_i\}_{i=1}^N$ and θ is the parameter of the generative model. The goal of the VAE is to generate samples that come from the identical distribution as the samples in the dataset, which is equivalent to maximize $p_\theta(x_i)$ for all x_i. VAE achieves this by introducing a latent variable z with a prior distribution $p(z)$ and modeling a decoder $p_\theta(X|z)$. The true posterior distribution $p_\theta(z|X)$ is approximated by training an encoder $q_\phi(z|X)$ parameterized by ϕ. VAE then optimizes the following variational lower bound:

$$\log p_\theta(X) \geq \mathbb{E}_{z\sim q_\phi(\cdot|X)}\left[\log p_\theta(X|z)\right] - \mathrm{KL}(q_\phi(z|X), p(z)). \tag{34}$$

Note that $\log p_\theta(X|z)$ denotes the reconstruction loss, and $\mathrm{KL}(p(\cdot), q(\cdot))$ represents the KL-divergence between two probability distribution $p(\cdot)$ and $q(\cdot)$. The prior distribution $p(z)$ is usually set to be a multivariate Gaussian distribution.

Conditional variational auto-encoder (CVAE) is a variant of the vanilla VAE, which aims to model $p_\theta(X|Y)$. The corresponding variational lower bound is given below.

$$\log p_\theta(X|Y) \geq \mathbb{E}_{z\sim q_\phi(\cdot|X,Y)}\left[\log p_\theta(X|z,Y)\right] - \mathrm{KL}(q_\phi(z|X,Y), p(z|Y)). \tag{35}$$

When sampling from the CVAE, we first sample from the prior distribution $p(z|Y)$ and pass it to the decoder to get the desired sample.

H Experimental Results on Other Datasets

To further illustrate the effectiveness of the MCQ algorithm, we conduct more experiments on 4 maze2d "-v1" datasets and 8 Adroit "-v0" datasets from D4RL benchmarks. We compare MCQ against behavior cloning (BC), BEAR [34], CQL [35], BCQ [14], TD3+BC [13], and IQL [32]. We take the results of BC, BEAR, CQL, BCQ on these datasets from [55] directly. For the results of TD3+BC and IQL, we run them on these datasets with their official codebases. All methods are run over 4 different random seeds. The hyperparameters we adopt for these tasks are shown in Table 10. We observe that MCQ behaves fairly well on maze2d datasets, and is competitive to prior methods on Adroit tasks. BC and CQL behaves well on Adroit datasets. Note that we find that BC beats MCQ on hammer-human-v0 and pen-cloned-v0. The reason may be attributed to the high complexity and narrow distribution of the Adroit tasks which makes it hard for Q networks to learn well, and the sampled action can easily lie outside of the dataset's support. Nevertheless, MCQ achieves the highest average score on these datasets.

Table 9: Normalized score comparison of different baseline methods on D4RL benchmarks. 0 corresponds to a random policy and 100 corresponds to an expert policy.

Task Name	BC	BEAR	CQL	BCQ	TD3+BC	IQL	MCQ (ours)
maze2d-umaze	-3.2	65.7	18.9	49.1	25.7±6.1	65.3±13.4	81.5 ±23.7
maze2d-umaze-dense	-6.9	32.6	14.4	48.4	39.7±3.8	57.8±12.5	107.8 ±3.2
maze2d-medium	-0.5	25.0	14.6	17.1	19.5±4.2	23.5±11.1	106.8 ±38.4
maze2d-medium-dense	2.7	19.1	30.5	41.1	54.9±6.4	28.1±16.8	112.7 ±5.5
Average Above	-2.0	35.6	19.6	38.9	35.0	37.2	102.2
pen-human	34.4	-1.0	37.5	68.9	0.0±0.0	68.7 ±8.6	68.5 ±6.5
door-human	0.5	-0.3	9.9	0.0	0.0±0.0	3.3±1.3	2.3±2.2
relocate-human	0.0	-0.3	0.2	-0.1	0.0±0.0	0.0±0.0	0.1 ±0.1
hammer-human	1.5	0.3	4.4	0.5	0.0±0.0	1.4±0.6	0.3±0.1
pen-cloned	56.9	26.5	39.2	44.0	0.0±0.0	35.3±7.3	49.4 ±4.3
door-cloned	-0.1	-0.1	0.4	0.0	0.0±0.0	0.5±0.6	1.3 ±0.4
relocate-cloned	-0.1	-0.3	-0.1	-0.3	0.0 ±0.0	-0.2±0.0	0.0 ±0.0
hammer-cloned	0.8	0.3	2.1	0.4	0.0±0.0	1.7 ±1.0	1.4 ±0.5
Average Total	7.2	13.9	14.3	22.4	11.7	23.8	44.3

Table 10: Detailed hyperparameters used in MCQ, where we conduct experiments on D4RL maze2d "-v1" datasets and Adroit "-v0" datasets.

Task Name	weighting coefficient λ	number of sampled actions N
maze2d-umaze	0.7	10
maze2d-umaze-dense	0.95	10
maze2d-medium	0.9	10
maze2d-medium-dense	0.9	10
pen-human	0.3	10
door-human	0.7	10
relocate-human	0.3	10
hammer-human	0.7	10
pen-cloned	0.9	10
door-cloned	0.3	10
relocate-cloned	0.3	10
hammer-cloned	0.5	10