Actor-Critic Alignment for Offline-to-Online Reinforcement Learning

Zishun Yu [1] **Xinhua Zhang** [1]

Abstract

Deep offline reinforcement learning has recently demonstrated considerable promises in leveraging offline datasets, providing high-quality models that significantly reduce the online interactions required for fine-tuning. However, such a benefit is often diminished due to the marked state-action distribution shift, which causes significant bootstrap error and wipes out the good initial policy. Existing solutions resort to constraining the policy shift or balancing the sample replay based on their online-ness. However, they require online estimation of distribution divergence or density ratio. To avoid such complications, we propose deviating from existing actor-critic approaches that directly transfer the state-action value functions. Instead, we post-process them by aligning with the offline learned policy, so that the Q-values for actions *outside* the offline policy are also tamed. As a result, the online fine-tuning can be simply performed as in the standard actor-critic algorithms. We show empirically that the proposed method improves the performance of the fine-tuned robotic agents on various simulated tasks.

1. Introduction

Offline reinforcement learning (RL) provides a novel tool that allows offline batch data to be leveraged by RL algorithms without having to interact with the environment (Levine et al., 2020). This opens up new opportunities for important scenarios such as health care decision making, and goal-directed dialog learning. Due to the limitation of offline data, it generally remains beneficial and necessary to fine-tune the learned model through online interactions, and ideally the latter will enjoy a faster learning curve thanks to the favorable initialization.

[1]Department of Computer Science, University of Illinois Chicago, Chicago, IL 60607, USA. Correspondence to: Zishun Yu <zyu32@uic.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

Unfortunately, it has been long observed that a direct offline-to-online (O2O) transfer often leads to catastrophic degradation of performance in the online stage, which is unacceptable in critical applications including medical treatment and autonomous driving. A key cause lies in the significant shift of state distribution at online phase compared with the offline data (Fujimoto et al., 2019; Kumar et al., 2019; Fu et al., 2019; Kumar et al., 2020a). As a result, the Bellman backup suffers a compounded error (Farahmand et al., 2010; Munos, 2005), because the Q-value has not been well estimated for the state-actions lying outside the offline distribution.

A number of solutions have been developed to address this issue. The most straightforward approach is importance sampling (Laroche et al., 2019; Gelada & Bellemare, 2019; Zhang et al., 2020; Huang & Jiang, 2020), which requires an additional effort of estimating the behavior policy, and suffers from high variance, especially when it differs markedly from the learned policy (a more realistic issue for the offline setting than the conventional off-policy setting). The model-based approach, on the other hand, also suffers from the distribution shift in state marginals and actions (Mao et al., 2022; Kidambi et al., 2020; Yu et al., 2020; Janner et al., 2019). It may exploit the model to pursue out-of-distribution (OOD) states and actions where the model mis-believes to yield a high return. So they also require detecting and quantifying the shift. In addition, they suffer from standard challenges plaguing model-based RL algorithms such as long horizon and high dimensionality.

Dynamic programming proffers lower variance and directly learns the value functions and policy. Several approaches have been proposed to combat distribution shift. A natural idea is to constrain the policy to the proximity of the behavior policy, and this has been implemented with probability divergences in AWAC (Nair et al., 2020) and Siegel et al. (2020); Peng et al. (2019); Wu et al. (2019); Kumar et al. (2019), or by behavior cloning regularization (Zhao et al., 2021; Fujimoto & Gu, 2021). A second class of approaches resort to pessimistic under-estimate of the state-action values (Kumar et al., 2020b; Kostrikov et al., 2021), especially for OOD actions that could have an unjustified high value. Conservative Q-learning (CQL, Kumar et al., 2020b) has been shown to produce a relatively safer O2O transfer in balanced replay (Lee et al., 2022), which further prioritizes the

experience transitions that are closer to the current policy.

Unfortunately, all these methods require online estimation of distribution divergence or density ratio (for priority score or regularization weight). Excess conservatism can also slow down the online fine-tuning. A third category of methods avoid these complications by estimating the epistemic uncertainty of the Q-function, so that OOD actions carry a larger uncertainty which in turn yields conservative target values for Bellman backup (Jaksch et al., 2010; O'Donoghue et al., 2018; Osband et al., 2016; Kumar et al., 2019). However, it is generally hard to find calibrated uncertainty estimates, especially for deep neural nets (Fujimoto et al., 2019).

To resolve the aforementioned issues, we propose a novel alignment step for actor-critic RL that can be flexibly inserted between offline and online training, dispensing with any estimation of Q-function uncertainty, distribution divergence, or density ratio. Our key insight is drawn from soft actor-critic (SAC, Haarnoja et al., 2018), where the optimal entropy-regularized policy is simply the softmax of the Q-function. Now that the Q-function is generally problematic for OOD actions while the policy learned offline is assumed trustworthy (though still needs fine-tuning), it is natural to *align* the critic to the actor upon the completion of offline learning, so that the Q function is tamed to be consistent with the policy under the softmax function, especially for those actions that lie outside the behavior policy. As a result, the online fine-tuning will only need to take the simple form of the standard SAC, and empirically the proposed method outperforms state-of-the-art fine-tuned robotic agents on various simulated tasks.

Our contribution and novelty can be summarized as follows:

- We propose a novel O2O RL approach that outperforms or matches the current SOTAs.
- Our approach does not rely on offline conservatism, allowing it to transfer to a broader range of offline models.
- We propose, for the first time, discarding Q-values learned offline as a means of combating distribution shift in O2O RL. We also design a novel reconstruction of Q-functions for online fine-tuning.
- When offline data is not available at online fine-tuning – a very realistic scenario due to data privacy concerns, our method remains applicable and stable, while strong competitors such as balanced replay cease being applicable.

2. Related Work

Decision transformer (Chen et al., 2021) and trajectory transformer (Janner et al., 2021) have recently been shown effective for offline reinforcement learning, where the batch trajectories' likelihood is maximized auto-regressively to model action sequences conditioned on a task. Zheng et al. (2022) extended them to *online* decision transformers (ODTs) by populating the replay buffer with online ODT rollouts labeled with hindsight experience replay. As a result, sequence modeling becomes effective for online fine-tuning. Our method remains in the actor-critic framework, and we demonstrate similar or superior empirical performance to ODT.

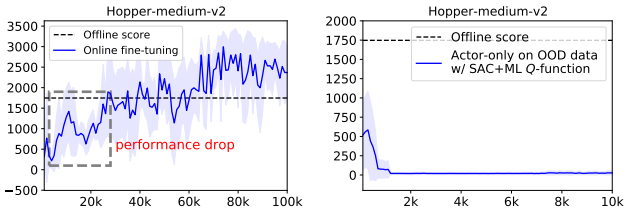
Behavior cloning often plays an important role in effective O2O RL. It can take the form of constraining the policy around the behavior policy under certain probability discrepancy measure, or simply imposing least square or cross-entropy regularization to drive the policy to imitate transitions (Zhao et al., 2021; Fujimoto & Gu, 2021). Such a regularizer often requires delicate annealing, and to this end, Zhao et al. (2021) designed heuristic rules based on reward feedback. Recently, Kostrikov et al. (2022a) employ behavior cloning to guide the extraction of policy from an expectile-based implicit Q-learning (IQL).

It is noteworthy that behavior cloning is also commonly used in imitation learning, where the goal is to imitate instead of outperforming the demonstrator, differing from the O2O setting. A number of efforts have been made to fuse it with RL for improvement (Lu et al., 2021). A similar line of research is to boost online learning from demonstration, (e.g., Hester et al., 2018; Reddy et al., 2019). However, they focus on accelerating online learning by utilizing offline data, and are not concerned about the safety or performance drop in porting the pre-trained policy to online.

In the area of robot learning, it is also of interest to initialize RL agent from hand-engineered policies or traditional controller such as model-predictive controllers (Silver et al., 2018; Johannink et al., 2019; Bouton et al., 2019). In particular, Silver et al. (2018) discussed the "misalignment" between an pre-specified actor and a critic that is potentially initialized poorly. They propose to update the critic alone for a "burn in" period during online fine-tuning until the critic loss falls below certain threshold. This setting is quite relevant to O2O RL although their actor and critic are not trained via offline RL. We include further discussions on the "burn in" solution in Appendix C.5.

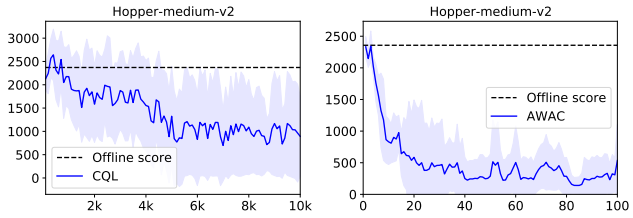
3. Preliminary

We follow the standard protocol that formulates a RL environment as a Markov decision process (MDP). An MDP $\mathcal{M}$ is often described as a 5-tuple $(\mathcal{S}, \mathcal{A}, \mathbb{P}, r, \gamma)$, where $\mathcal{S}$ is the state-space, $\mathcal{A}$ is the action space, $\mathbb{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition function, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is a discount factor. A policy is a distribution $\pi(a|s) \in \Delta(\mathcal{A})$, and the agent aims to find a policy that maximizes the expected return $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t]$.



(a) Score vs. # online updates (b) Score vs. # actor updates

Figure 1: Online performance vs. number of updates initialized from pre-trained SAC+ML policy and Q-function. (a): updating all learnable parameters and interacting with environment to collect data online; (b): controlled actor-only experiment with SAC actor and temperature updates.



(a) Score vs. # updates (CQL) (b) Score vs. # updates (AWAC)

Figure 2: Actor-only experiments: (a) SAC-style actor-only updates initialized from pre-trained CQL policy and Q-function; (b) AWAC-style actor-only updates initialized from pre-trained AWAC policy and Q-function.

Soft actor-critic To learn from offline data generated by a behavior policy, we will focus on off-policy RL methods. In particular, the soft actor-critic method (SAC, Haarnoja et al., 2017; 2018) learns a Q-function $Q_\mu(s,a)$ with parameter μ, and a Gaussian policy $\pi_\theta(a|s)$ whose sufficient statistics are determined by a neural network with parameter θ. Let $\mathbf{d}$ be the empirical distribution corresponding to the replay buffer, and we intentionally left it flexible on state, state-action, or transition. Then SAC alternates between updating the critic and actor by minimizing the following respective objectives:

$$\mathcal{L}_\pi^{\text{SAC}}(\theta,\mathbf{d}) := \mathop{\mathbb{E}}_{s\sim\mathbf{d}} \mathop{\mathbb{E}}_{a\sim\pi_\theta(\cdot|s)} \left[\alpha \log \pi_\theta(a|s) - Q_\mu(s,a)\right],$$

$$\mathcal{L}_Q^{\text{SAC}}(\mu,\mathbf{d}) := \mathop{\mathbb{E}}_{(s,a,r,s')\sim\mathbf{d}} \left[\Big(Q_\mu(s,a) - y(r,s')\Big)^2\right], \quad (1)$$

where $y(r,s') := r + \gamma \mathop{\mathbb{E}}_{a'\sim\pi_\theta(\cdot|s')} \left[Q_{\bar{\mu}}(s',a') - \alpha \log \pi_\theta(a'|s')\right].$

Here, $\alpha > 0$ and $\bar{\mu}$ is the delayed Q-function parameter. If π_θ is based on a universal neural network, its optimal value that minimizes $\mathcal{L}_\pi^{\text{SAC}}(\theta,\mathbf{d})$ admits a closed form:

$$\pi_\theta(a|s) = \exp\left(\tfrac{1}{\alpha} Q_\mu(s,a)\right) \Big/ \sum_{a\in\mathcal{A}} \exp\left(\tfrac{1}{\alpha} Q_\mu(s,a)\right). \quad (2)$$

In practice, one simply performs gradient descent steps on $\mathcal{L}_\pi^{\text{SAC}}$ because even if the network is universal, the value of θ that corresponds to the optimal solution (2) is hard to find.

It is important to note that adding a baseline function $Z(s)$ to $Q_\mu(s,a)$ does not change the optimal π_θ in (2). Therefore, given π_θ, $Q_\mu(s,a)$ can be recovered as

$$Q_\mu(s,a) = Z(s) + \alpha \log \pi_\theta(a|s), \quad (3)$$

where $Z(s)$ provides additional freedom to fit other aspects of the problem; see Section 5.2.

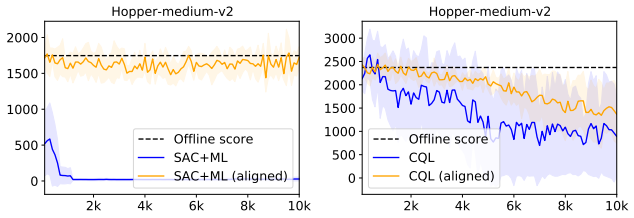
Our inspiration is drawn from (3), where the critic Q is *aligned* with the actor π_θ. Although such a connection is derived from the opposite direction in (2), it allows us to initialize the *online* critic in line with the *offline* learned actor. This means, the actor that minimizes the entropy-regularized loss $\mathcal{L}_\pi^{\text{SAC}}$ based on such an aligned Q-function will be exactly the offline learned actor. As a result, compared with just encoding the policy within the actor, additional protection is now provided against drifting rapidly from offline trained policy. We will first overview this motivation in Section 4, and then detail the approach in Section 5.

4. Challenges in O2O RL and Motivation Behind Actor-Critic Alignment

We first show the challenges in O2O RL on the hopper-medium-v2 dataset, and then *preview* the effectiveness of aligning online critic with offline actor. Figure 1a shows significant performance drop if online fine-tuning is directly applied to an offline trained model, a phenomenon observed by many prior works. Here the offline method is SAC+ML which will be introduced in Section 5.1, but similar performance was observed with other offline methods.

To further demonstrate such an issue in a controlled fashion, we designed an *actor-only* experiment in order to illustrate how offline trained Q-functions impact online fine-tuning on OOD data. We first trained offline on hopper-medium-v2, and then used hopper-random-v2 as OOD data , to ablate the factor of online data collection. Subsequent updates were applied on actor only. The distribution shift is presented in Appendix B.1. As confirmed by Figure 1b, because of applying offline trained Q-function to OOD data, running actor-only updates suffers a drop from offline score.

To investigate whether this problem is unique to the SAC+ML method chosen for offline learning, we next tested with CQL and AWAC. Figure 2a used the Q-function from CQL offline training followed by SAC fine-tuning for actor only on OOD data, and it again failed to keep up the offline score. Figure 2b shows even more severe drop for AWAC. Combined with the SAC+ML result in Figure 1, one



(a) SAC+ML for offline training (b) CQL for offline training

Figure 3: Comparison of *actor-only* updates between *aligned* Q-function (orange) and offline Q-function (blue).

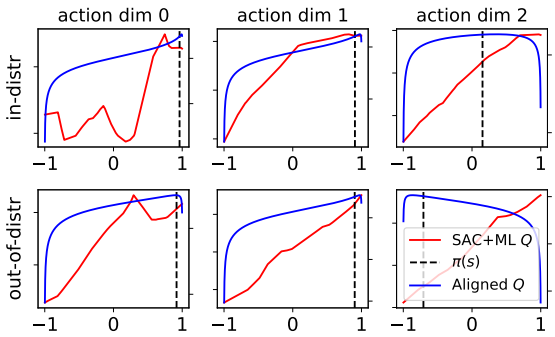


Figure 4: SAC+ML Q-values (left y-axis) v.s. aligned Q-values (right y-axis) for in-distribution sample (top row) and OOD sample (bottom row). Since only the trend of each curve matters, we omit the y-axis tick values.

may postulate that this issue plagues offline RL in general, although it could be alleviated by conservative estimation.

4.1. Preview of actor-critic alignment performance

Interestingly, our alignment step (to be introduced in Section 5.2) manages to resolve this issue under SAC+ML offline training as shown in Figure 3a. Since we only update the actor here, it is expected that no improvement can be achieved upon the offline score. Similarly, Figure 3b shows that the decay of performance is much slowed down thanks to the alignment. In Appendix B.2, we show actor-only experiment results on more tasks where our alignment step allows offline scores to be retained for most of tasks while SAC+ML and CQL failed.

As a first step towards understanding the impact of alignment, we visualize in Figure 4 how the offline Q-functions deviate from the actor over OOD states, and how the alignment operation restores the consistency. Here the plot is based on two randomly sampled states, and more comprehensive results on a larger population are relegated to Appendix F.1. We trained SAC+ML on hopper-medium, and sampled in-distribution states from it. To sample OOD states, we resorted to the hopper-random dataset.

The top row of Figure 4 compares the offline learned Q-values and our aligned/reconstructed Q-values for an in-distribution sample. The bottom row shows a similar comparison, but on an OOD sample. The actions have 3 dimensions, and for the i-th column, we perturbed the i-th dimension in $[-1, 1]$, with all the other dimensions fixed. Clearly, the offline learned Q-values are often inconsistent with the policy's choice, even for in-distribution samples. But our alignment much improves the consistency, which encourages the policy to stay close to the offline policy, safeguarding the process of transfer.

5. Aligning Critics with Actors for O2O RL

We now detail our method that consists of three phases: offline, actor-critic (AC) alignment, and online. The whole procedure is summarized in Table 5 in Appendix A.

5.1. Offline Training

Since our contribution lies in O2O transfer, the offline training method can be general indeed. To be self-contained, we now describe an offline actor-critic approach using SAC with maximum likelihood (ML) regularization, noting that other methods are also applicable and will be used in our experiment (e.g., CQL in Section 6.3), provided that it produces a stochastic policy. However, TD3+BC (Fujimoto & Gu, 2021) runs TD3 (Fujimoto et al., 2018) offline with a behaviour cloning regularization (BC, Bain & Sammut, 1995). It is not applicable in our setting because its policy is deterministic. So we replaced TD3 with SAC whose policy is stochastic, akin to SAC+BC (Nair et al., 2020). Moreover, since the online phase uses an ML regularizer (see Section 5.3), we also adopted ML for offline, hence the name SAC+ML. We will compare SAC+ML with TD3+BC in Appendix C.1, where their performances appear comparable.

Actor update. Let $\mathbf{d}$ be the empirical distribution of a mini-batch sampled from the offline dataset $\mathcal{D}$. The actor update of TD3+BC and SAC+ML aims to minimize the following respective objectives:

$$\mathcal{L}_\pi^{\text{TD3+BC}}(\theta, \mathbf{d}) = \mathop{\mathbb{E}}_{(s,a)\sim\mathbf{d}}\Big[-\lambda Q_\mu(s, \pi_\theta(s)) + (\pi_\theta(s) - a)^2\Big],$$

$$\mathcal{L}_\pi^{\text{SAC+ML}}(\theta, \mathbf{d}) = \mathop{\mathbb{E}}_{(s,a)\sim\mathbf{d}} \mathop{\mathbb{E}}_{b\sim\pi_\theta(\cdot|s)}\Big[-\log \pi_\theta(a|s) - \lambda\Big(Q_\mu(s, b) - \alpha \log \pi_\theta(b|s)\Big)\Big], \tag{4}$$

where the hyperparameter λ balances Q values with the BC or ML regularization. In practice, we employed the clipped double Q-learning technique (Hasselt, 2010) to train two Q-networks Q_{μ_1} and Q_{μ_2}. It is beneficial for both offline

and online training (Fujimoto et al., 2018). λ is then set to

$$\lambda := \omega \,/\, \mathop{\mathbb{E}}_{(s,a)\sim\mathbf{d}} |Q_\mu(s,a)|, \quad \text{where } Q_\mu := \min\{Q_{\mu_1}, Q_{\mu_2}\},$$

and ω is a hyper-parameter. So λ is recomputed after every critic update, requiring almost no additional computation.

Critic update. SAC+ML follows the same critic update as SAC in (1), except for the double Q part:

$$\mathcal{L}_Q^{\text{SAC+ML}}(\mu_i, \mathbf{d}) := \mathop{\mathbb{E}}_{(s,a,r,s')\sim\mathbf{d}} \Big[\big(Q_{\mu_i}(s,a) - y(r,s')\big)^2\Big] \quad (5)$$

$$\text{with } y(r,s') := r + \gamma \mathop{\mathbb{E}}_{a'\sim\pi_\theta(\cdot|s')} [Q_{\bar{\mu}}(s',a') - \alpha \log \pi_\theta(a'|s')],$$

and $\bar{\mu}$ is a delayed version of μ, a.k.a. target network, with $Q_{\bar{\mu}}(s,a) = \min_{i\in\{1,2\}} Q_{\bar{\mu}_i}(s,a)$, akin to Q_μ. The temperature update tries to reduce the following term over $\alpha > 0$:

$$\mathcal{L}_{\text{temp}}^{\text{SAC+ML}}(\alpha, \mathbf{d}) := -\alpha \mathop{\mathbb{E}}_{s\sim\mathbf{d}} \mathop{\mathbb{E}}_{a\sim\pi_\theta(\cdot|s)} \big[\log \pi_\theta(a|s) - \bar{\mathcal{H}}\big]. \quad (6)$$

Here, $\bar{\mathcal{H}} > 0$ is the target entropy value, a hyper-parameter specified a priori. The Lagrange multiplier α is automatically tuned in (6), lower bounding the entropy of π_θ by $\bar{\mathcal{H}}$. The pseudo-code of SAC+ML is relegated to Appendix A.

5.2. Actor-critic Alignment

At the end of offline learning, the learned policy π_{θ_0} often performs reasonably well, and is ready for online fine-tuning. So we denote the policy with index 0. In conventional actor-critic, the critic is supposed to be updated frequently enough to accurately pursue the state-action values for the current policy. However, even if such updates are conducted proactively, the distribution shift problem in O2O still plagues the critic under deep net approximation, because the Q-values are not trustworthy beyond what has been visited under the behavior policy. So the over-estimated Q-values can rapidly destroy the learned actor and critic through Bellman backup.

In order to avoid this issue, we propose taming the out-of-distribution Q-values by directly aligning the critics with the actors, as a post-processing step for offline learning, or an initialization step for online learning. In particular, inspired by (3), we choose to discard the Q_{μ_i} learned from the offline phase, and reset them into[1]

$$Q_i(s,a) = \log \pi_{\theta_0}(a|s) + Z_{\psi_i}(s). \quad (7)$$

The baseline $Z_{\psi_i}(s)$ can be naturally calibrated by minimizing the Bellman residual on offline data:

$$\mathcal{L}_Z^{\text{SAC+ML}}(\psi_i, \mathbf{d}) := \mathop{\mathbb{E}}_{(s,a,r,s')\sim\mathbf{d}} \big[(\log \pi_{\theta_0}(a|s) + Z_{\psi_i}(s) - y(r,s'))^2\big], \quad (8)$$

$$\text{where } y(r,s') := r + \gamma \mathop{\mathbb{E}}_{a'\sim\pi_{\theta_0}(\cdot|s')} [\log \pi_{\theta_0}(a'|s') + Z_\psi(s')],$$

$$Z_\psi := \min\{Z_{\psi_1}, Z_{\psi_2}\}. \quad (9)$$

The standard semi-gradient is employed on Z_ψ in (9). This optimization is simply a regression problem and can be solved by Adam. The details are deferred to Appendix A.1, where the pseudo-code is also given in Algorithm 1.

Flexibility in offline training. Thanks to this alignment step that disregards the Q-function learned offline, the offline learning algorithm is not limited to SAC+ML. In Section 6.3, we will show that our alignment approach can be well applied to the offline policy learned from CQL.

5.3. Online Training

During the online fine-tuning, we restore the full flexibility of Q-functions by using the following parameterization:

$$Q_{\phi_i}(s,a) \coloneqq \log \pi_{\theta_0}(a|s) + R_{\phi_i}(s,a), \quad \text{where} \quad R_{\phi_i}(s,a) \text{ is } \textbf{initialized} \text{ with } Z_{\psi_i}(s). \quad (10)$$

Such an initialization can be simply implemented by loading the weights of Z_{ψ_i} and setting the weights corresponding to action to zeros. It is noteworthy that one should refrain from constraining Q to closed-form manifold induced by the latest π_θ throughout the online phase, i.e., setting $Q_{\phi_i}(s,a)$ to $\log \pi_\theta(a|s) + Z_{\phi_i}(s)$ for some trainable baseline Z_{ϕ_i}. This is because it would lead to no improvement of the policy. As such, we only leverage the closed-form for initialization.

The update on temperature is exactly the same as (6), and the update on critic resembles that of the offline phase in (5), except that the training variable is now only $R_{\phi_i}(s,a)$. In particular, we adapt SAC critic update to our Q_ϕ, along with standard tricks of target network and double Q-clipping:

$$\mathcal{L}_Q(\phi_i, \mathbf{d}) \coloneqq \mathop{\mathbb{E}}_{\mathbf{d}} \left[\Big(\log \pi_{\theta_0}(a|s) + R_{\phi_i}(s,a) - y(r,s')\Big)^2\right],$$

$$\text{where } y(r,s') \coloneqq r + \gamma \mathop{\mathbb{E}}_{a'\sim\pi_\theta(\cdot|s')} \big[\log \pi_{\theta_0}(a'|s') + R_{\bar{\phi}}(s',a') - \alpha \log \pi_\theta(a'|s')\big].$$

The first expectation is over $(s,a,r,s') \sim \mathbf{d}$. The actor's objective follows from the **vanilla SAC**, and can be written as follows with $R_\phi := \min_{i\in\{1,2\}} R_{\phi_i}$, $Q_\phi := \log \pi_{\theta_0} + R_\phi$,

[1] Compared with (3), it appears that we have set α there to 1, while its value at the end of offline learning is generally much smaller than 1. This creates no contradiction, however, because $\log \pi_{\theta_0}$ will be used to parameterize the online Q-function as in (10), and the α for online phase SAC is initialized to 1. So their product, passed through the softmax, will recover π_{θ_0}.

and $\mathbf{d}$ being a mini-batch sampled from the replay buffer:

$$\begin{aligned}\mathcal{L}_\pi(\theta, \mathbf{d}) \coloneqq & - \mathop{\mathbb{E}}_{s\sim\mathbf{d}} \mathop{\mathbb{E}}_{a\sim\pi_\theta(\cdot|s)} \left[Q_\phi(s,a) - \alpha \log \pi_\theta(a|s)\right], \\ = & - \mathop{\mathbb{E}}_{s\sim\mathbf{d}} \mathop{\mathbb{E}}_{a\sim\pi_\theta(\cdot|s)} \left[R_\phi(s,a) - \alpha \log \pi_\theta(a|s)\right] \\ & \underbrace{- \mathop{\mathbb{E}}_{s\sim\mathbf{d}} \mathop{\mathbb{E}}_{a\sim\pi_\theta(\cdot|s)} \left[\log \pi_{\theta_0}(a|s)\right]}_{\text{penalizing deviation of } \pi_\theta \text{ from } \pi_{\theta_0}}. \end{aligned} \tag{11}$$

Behavior cloning in (11). Naturally unfolding from SAC using the parameterization (10) is the expectation of log-likelihood of π_{θ_0} under π_θ in (11), a maximum-likelihood term that enforces the actions favored by the new policy π_θ to also enjoy a high log-likelihood under the offline policy π_{θ_0}. Different from AdaBC (Zhao et al., 2021), we sidestepped an ad-hoc introduction of behavior cloning regularization and tweaking of its weight. This regularization is *not* applied on the offline data, but on the policy π_{θ_0} achieved by offline learning. To be consistent, we also used the maximum-likelihood regularization in offline training.

One might argue that this interpretation is artificial because, after all, the $\log \pi_{\theta_0}$ term can be subsumed into the free variable R_{ϕ_i} in (10), obliterating this BC regularizer in (11). This in fact makes sense if the entire optimization is convex and the range of R_{ϕ_i} as a function set is closed under addition with $\log \pi_{\theta_0}$. However, since R_{ϕ_i} is a neural net , such conditions do not hold true. As a result, the composite form in (10) does play a crucial role in the empirical performance, which is manifested in our ablation study in Section 6.5.

In practice, we also introduced two techniques to stabilize online learning. The first β-clipping trick addresses the excessively large magnitude of $\log \pi_{\theta_0}$ by capping its absolute values. The second critic interpolation gives the flexibility to balance between safety transfer and policy improvement. For the sake of space, they are deferred to Appendix A.2.

6. Experiments

We next compared our actor-critic alignment method (ACA) with a number of state-of-the-art methods as summarized in Table 1. Although CQL was not developed for O2O transfer, we still included it due to its strong performance. The implementation of our ACA algorithm can be found at https://github.com/ZishunYu/ACA.

Our experiments aim to demonstrate:

- SAC→ACA matches or outperforms SOTAs such as balanced replay (BR, Lee et al., 2022), advantage weighted actor critic (AWAC, Nair et al., 2020), and online decision transformer (ODT, Zheng et al., 2022);
- Direct transfer such as SAC→SAC and CQL→SAC suffers significant performance drop;
- Transfer from offline method significantly outperforms training SAC online from scratch. We will present ablation studies to examine various components of ACA.

6.1. Comparison with baseline methods

We used the HalfCheetah, Hopper, and Walker2d environments from the D4RL-v2 datasets (Fu et al., 2020). Each of them has five levels. All offline/online experiments ran 5 random seeds. We ran all offline methods for 500 episodes with 1000 mini-batches each, except for halfcheetah-medium-expert, halfcheetah-expert, and hopper-expert where we ran CQL for 1500 episodes to converge. All online experiments were run for 100 episodes with 1000 environment interactions each. This protocol is quite commonly used, and more implementation details are deferred to Appendix E.

Figure 5 shows the average return as a function of training episodes, achieved at each offline model (left half of the subplots) and online model (right half). Since SAC→ACA and SAC→SAC share the same offline method, their curves coincide on the left of the subplots, with the green curve shown only (no blue) on the left. A similar situation occurs to CQL→BR and CQL→SAC, and only the purple curve is shown on the left half (no pink).

In Figure 5, CQL→SAC (purple→pink) drops significantly on the expert level (fifth row) and medium-expert level (fourth row). SAC→SAC (green→blue) drops in almost all cases, except random (first row) and medium-replay (third row). It is clear that our SAC→ACA (green) barely suffers performance drop. The only exception is Hopper-medium-expert, but all other methods (except AWAC) also suffer a drop there, while ours recover most rapidly. Besides, ours offers comparable policy improvement to the strongest baseline, which is CQL→BR.

Since different baselines in Table 1 employ different offline methods, **we emphasize** that it is insufficient to base the comparison of fine-tuning methods only on their final online performance. Therefore, we provided in Table 2—via numbers in round brackets—the increase (δ) of return achieved by fine-tuning on top of the final offline policy. CQL→BR appears the best among all baseline methods, scoring an increase of **190.94** in total. However, it is significantly outperformed by our SAC→ACA, which achieves a total improvement of **278.31**, while also being the highest in the final score. As Nair et al. (2020) also reports fine-tuning results at 500k online steps, we will provide further comparisons with their results in Appendix C.2.

Besides the numerical advantage, ACA also offers additional methodological benefits by accommodating different offline methods for initialization (Section 6.3), and learning online without accessing offline data (Section 6.4).

Table 1: Baseline algorithms for O2O RL. See acronyms below.

Flag	Name	Offline	Online	Description
	SAC→ACA (Ours)	SAC+ML	Algorithm 3	Our method init from SAC+ML
	SAC→SAC	SAC+ML	SAC	SAC init from SAC+ML
	CQL→BR	CQL	SAC w/ BR	Balanced replay init from CQL
	CQL→SAC	CQL	SAC	SAC init from CQL
	AWAC	AWAC	AWAC	AWAC init from AWAC
	SAC	-	SAC	SAC from scratch

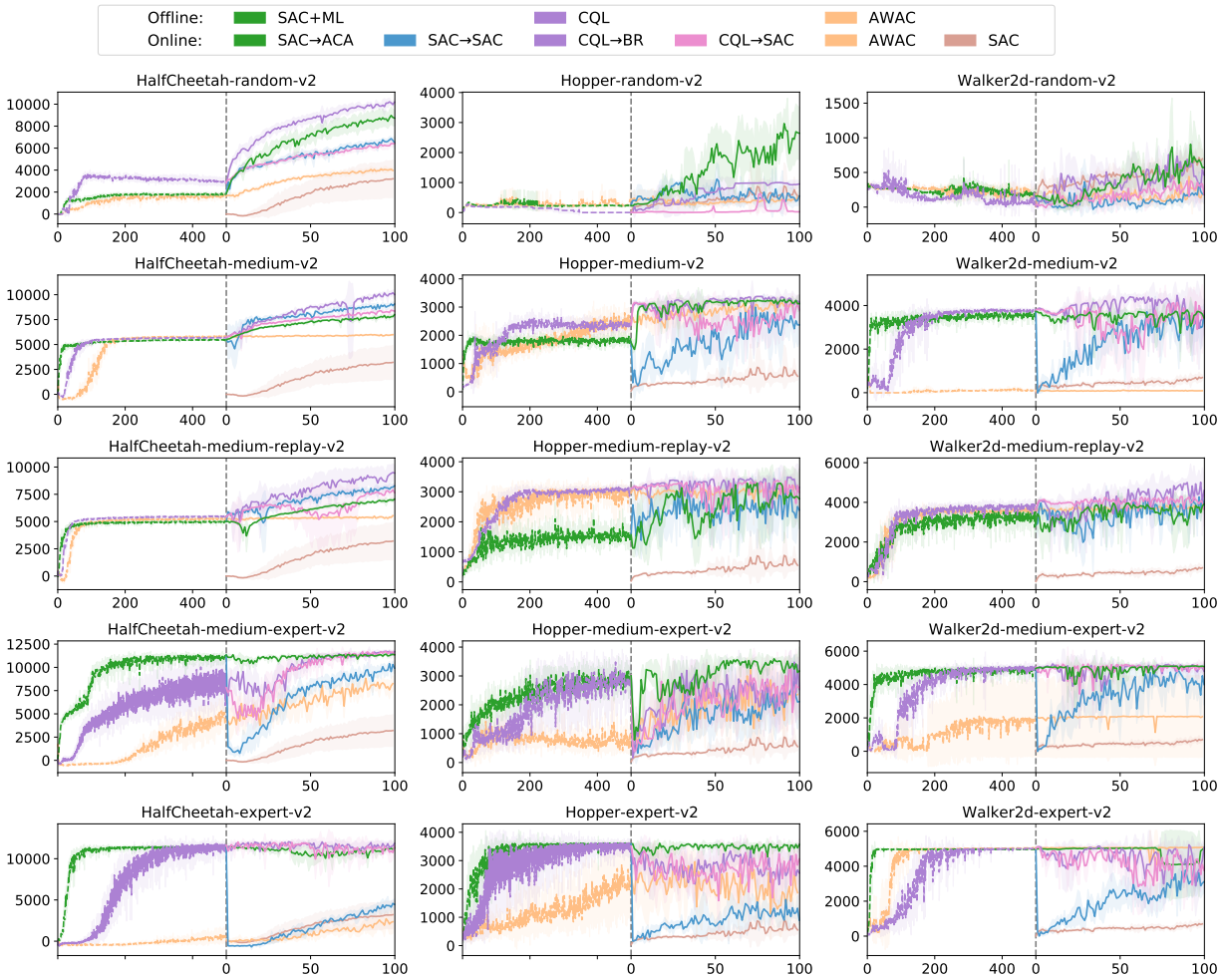


Figure 5: Comparing SAC→ACA (ours) with other baselines for offline-to-online RL. The shaded areas stand for the standard deviation. Refer to Table 1 for legend meanings.

6.2. Comparison with online decision transformer

Since ODT is not based on dynamic programming, we compared it with SAC→ACA in this separate section. As Zheng et al. (2022) experimented using 200k online samples and averaged over 10 seeds, we ran SAC+ML with 5 additional seeds and ran 200k online steps for all 10 SAC+ML runs, to make the comparison fair. In addition, we included IQL for comparison, with both ODT and IQL results duplicated from Zheng et al. (2022). As shown in Table 3, for almost all medium and medium-replay tasks, our SAC→ACA outperforms ODT and IQL in both final performance and performance increase (δ) by a large margin.

In addition, we conducted comparisons on the AntMaze-v2 environments, and observed that the AntMaze environment could benefit from i) standard Gaussian policy parameterization rather than squashed Gaussian, and ii) less stochastic policy by using a lower value of $\widetilde{\mathcal{H}}$. Further details can be found in Appendix E. We also report offline scores, stan-

Table 2: Average normalized D4RL scores of various O2O methods. Outside parenthesis: scores at the end of 100k online steps. Inside parenthesis: the increase of that score upon the end of offline training.

Dataset	Env	Score(δ) SAC→SAC	CQL→SAC	AWAC	CQL→BR	SAC→ACA (ours)
Random	HalfCheetah	54.60(37.31)	54.00(28.70)	34.26(18.94)	84.36(59.06)	72.60(55.31)
	Hopper	17.62(9.69)	1.37(0.72)	16.85(3.03)	29.80(29.15)	81.85(73.92)
	Walker2d	3.86(0.50)	3.91(3.25)	4.15(-0.58)	10.05(9.39)	12.42(9.06)
Medium	HalfCheetah	75.20(28.86)	69.52(21.10)	50.48(1.54)	82.95(34.52)	66.58(20.25)
	Hopper	73.39(19.08)	89.77(16.30)	97.53(24.48)	98.14(24.67)	96.54(42.24)
	Walker2d	79.63(-1.53)	81.78(-0.70)	1.93(-0.54)	76.36(-6.11)	74.66(-6.50)
Med.-Replay	HalfCheetah	68.90(26.37)	63.91(18.01)	46.84(2.42)	78.36(32.46)	59.03(16.50)
	Hopper	74.04(25.22)	92.01(-3.95)	95.98(0.00)	97.25(1.28)	85.54(36.72)
	Walker2d	85.40(23.21)	79.28(0.89)	80.81(2.97)	100.06(21.68)	85.17(22.98)
Med.-Expert	HalfCheetah	82.15(-11.38)	93.94(28.75)	68.75(32.30)	96.35(31.17)	93.74(0.21)
	Hopper	65.44(-27.64)	80.46(-13.26)	73.13(47.50)	78.51(-15.22)	98.02(4.94)
	Walker2d	87.18(-20.95)	107.03(-2.63)	45.21(4.45)	104.43(-5.22)	110.54(2.42)
Expert	HalfCheetah	38.17(-55.42)	94.88(-0.98)	21.23(14.83)	97.67(1.82)	93.14(-0.46)
	Hopper	28.20(-82.68)	94.56(-13.95)	57.97(-12.85)	79.32(-29.19)	110.21(-0.67)
	Walker2d	67.76(-40.45)	81.92(-27.24)	110.68(0.80)	110.65(1.50)	109.59(1.38)
Total		901.54(-69.80)	1088.33(55.03)	805.80(139.29)	1224.26(190.94)	1249.65(278.31)

Table 3: Comparing SAC→ACA with ODT and IQL. HC = HalfCheetah, H = Hopper, W = Walker2d, AM=AntMaze.

Dataset	Env	IQL(200k)	δ_{IQL}	ODT(200k)	δ_{ODT}	ACA(200k)	δ_{ACA}
Medium	HC	47.41	0.04	42.16	-0.56	72.67	26.28
	H	66.79	2.98	97.54	30.59	99.32	42.39
	W	80.33	0.44	76.79	4.60	76.05	-3.30
Med.-Replay	HC	44.14	0.04	40.42	0.43	64.29	22.11
	H	96.23	4.10	88.89	2.25	103.17	53.92
	W	70.55	-3.12	76.86	7.94	82.09	18.89
Locomotion total		405.45	4.48	422.66	45.25	497.58	160.27
AM-umaze		89.50	2.40	88.50	35.40	93.0	35.0
AM-umaze-diverse		56.80	-7.60	56.00	7.99	67.0	27.0
AntMaze total		146.30	-5.20	144.50	43.39	160.0	62.0

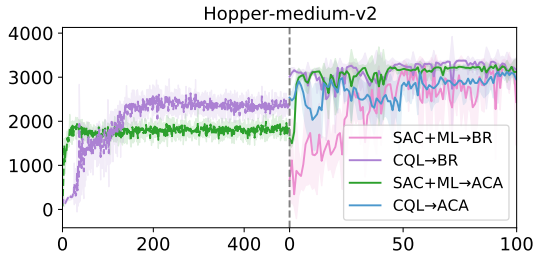


Figure 6: ACA could achieve similar performance while initializing from both SAC+ML/CQL. BR requires CQL.

dard deviations, in addition to Table 3, in Table 8 along with AntMaze training curves in Figure 12 in Appendix C.3.

6.3. Initialization from different offline methods

A key advantage of our alignment method lies in the flexibility of utilizing any offline RL method, as long as it outputs a stochastic policy because the Q-function is reset anyway. In contrast, SOTA methods sometimes require certain properties in the offline method such as pessimism. For example, BR's performance depends critically on the use of CQL.

To demonstrate our flexibility, we adopted CQL for offline learning and made a simple change to the alignment step which, in (8), clips $\log \pi_{\theta_0}$ to 0 when it is negative. In comparison, we also tested BR by using SAC+ML as the offline learner. Figure 6 shows the results of ACA/BR initialized from SAC+ML/CQL. While the performance of our approach does not change much when initializing from different offline models, BR shows significant performance drops when it is initialized from SAC+ML, i.e. non-pessimistic offline training. Further comparison is deferred to Appendix D.3.

6.4. Online training without offline data

When the application precludes the accessibility of offline data during online fine-tuning, we re-ran the benchmarks for medium-replay, medium-expert, expert. There is obviously no reason to replay offline data at random level, and empirically, we observed that online fine-tuning already performed well on medium when no offline data was replayed.

Figure 13 in Appendix C.4 shows the online performance of our method without using offline data, compared with other baselines which also do not access offline data during online fine-tuning. The BR algorithm requires offline data. So compared with Figure 5, we no longer have the purple line that corresponds to CQL→BR. It turns out that all the other baselines retain similar online performance as in Figure 5,

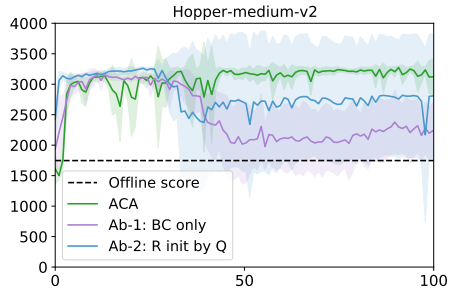


Figure 7: Ablation study: we changed the critic update to show that the re-parameterization of Q-function is crucial.

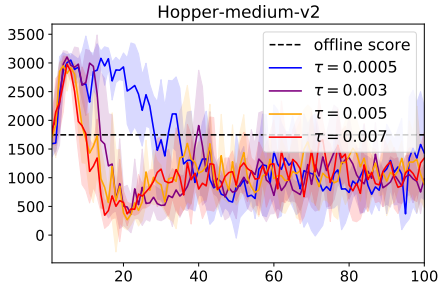


Figure 8: Online fine-tuning with delayed π_{θ^-} instead of π_{θ_0}. The delayed parameter θ^- is updated by $\theta^- \leftarrow \tau\theta^- + (1-\tau)\theta$ with different values of τ.

which had been shown inferior to our SAC→ACA. Table 4 further highlights that ACA does not exhibit significant change in online performance in the absence of offline data.

Table 4: Scores for SAC→ACA at 100k online steps, and its increase from offline result (in parenthesis). Comparison is made between with or without offline data.

Dataset	Env	Score(δ) w/ offline data	Score(δ) w/o offline data
Med.-Replay	HalfCheetah	59.03(16.50)	59.48(16.95)
	Hopper	85.54(36.72)	77.19(28.37)
	Walker2d	85.17(22.98)	84.27(22.08)
Med.-Expert	HalfCheetah	93.74(0.21)	93.81(0.28)
	Hopper	98.02(4.94)	105.67(12.59)
	Walker2d	110.54(2.42)	110.93(2.81)
Expert	HalfCheetah	93.14(-0.46)	90.76(-2.83)
	Hopper	110.21(-0.67)	109.22(-1.66)
	Walker2d	109.59(1.38)	110.52(2.31)
Total		844.98(84.02)	841.84(80.88)

6.5. Ablation Study

Choice of R_ϕ initialization As mentioned in Section 5.3, $\log \pi_{\theta_0}$ in (11) can be considered as a "behaviour cloning" regularization. One may wonder whether this, instead of actor-critic alignment, is the primary contributor to the empirical effectiveness. We therefore conducted the following ablation study, which answers this question in the negative.

In contrast to the parameterization of online Q-function in (10), we designed two alternatives where R_ϕ is initialized through offline critic Q_μ instead of Z_ψ, and the actor update was kept intact as in (11). Ablation 1 (**BC only**): we used regular SAC critic updates without our parameterization. The term $\log \pi_{\theta_0}$ now only appears in the actor update, which could be seen a SAC framework with behaviour cloning regularized actor update. Ablation 2 ($\mathbf{R}_\phi$ **init by** $\mathbf{Q}_\mu$): we adopt the same decomposed parameterization as in (10), but initialize R_{ϕ_i} with the offline learned Q_{μ_i}, instead of Z_{ψ_i}, to show the importance of Z_{ψ_i}. The objectives of the two ablations are relegated to Appendix D.1.

Figure 7 shows that the two ablation alternatives suffer clear performance drop due to the attributed error in the offline trained Q-functions. See Appendix D.1 for more comparison.

Choice of $\log \pi_{\theta_0}$ Reparameterzing Q_ϕ with $Q_\phi = \log \pi_\theta + R_\phi$ and $Q_\phi = \log \pi_{\theta_0} + R_\phi$ are technically equivalent at *initialization*. However, online updates of Q_ϕ might benefit from using a static $\log \pi_{\theta_0}$, as $Q_\phi = \log \pi_{\theta_0} + R_\phi$ has only one moving part R_ϕ, leading to potential better training stability.

We now verify our hypothesis and our choice of π_{θ_0} by testing $Q_\phi = \log \pi_{\theta^-} + R_\phi$, where θ^- is the delayed version of θ. Figure 8 shows that it fails in the hopper-medium task regardless the choice of τ. It is noteworthy that the smallest $\tau = 0.0005$ performs the best among all values of τ, consistent with the good performance achieved by our choice π_{θ_0} (which corresponds to $\tau = 0$). Intuitively, when the policy experiences a performance drop, hard-wiring Q towards the delayed π_{θ^-} will amplify this negative effect.

7. Conclusion and Future Work

We proposed an actor-critic alignment method that allows safe offline-to-online RL and achieves strong empirical performance. To combat distribution shift, we designed a novel approach that disregards offline learned Q-functions, and reconstructs it based on the learned policy using a closed-form that is motivated from the entropy-regularized actor update. Since it does not need an offline critic, online actor-critic fine-tuning is made possible for offline supervised methods, e.g. decision transformer and RvS (Emmons et al., 2022).

Acknowledgements

We thank the reviewers for their constructive comments. This work is supported by NSF grant RI:1910146 and NIH grant R01CA258827.

References

Bain, M. and Sammut, C. A framework for behavioural cloning. In *Machine Intelligence 15*, pp. 103–129, 1995.

Bouton, M., Julian, K. D., Nakhaei, A., Fujimura, K., and Kochenderfer, M. J. Decomposition methods with deep corrections for reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 33:330–352, 2019.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Emmons, S., Eysenbach, B., Kostrikov, I., and Levine, S. RvS: What is essential for offline RL via supervised learning? In *International Conference on Learning Representations (ICLR)*, 2022.

Farahmand, A.-m., Szepesvári, C., and Munos, R. Error propagation for approximate policy and value iteration. In *Advances in Neural Information Processing Systems*, 2010.

Fu, J., Kumar, A., Soh, M., and Levine, S. Diagnosing bottlenecks in deep q-learning algorithms. In *International Conference on Machine Learning (ICML)*, pp. 2021–2030, 2019.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Fujimoto, S. and Gu, S. S. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*, pp. 1587–1596. PMLR, 2018.

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

Gelada, C. and Bellemare, M. G. Off-policy deep reinforcement learning by bootstrapping the covariate shift. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning (ICML)*, pp. 1352–1361. PMLR, 2017.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, pp. 1861–1870. PMLR, 2018.

Hasselt, H. Double Q-learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2010.

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Osband, I., Dulac-Arnold, G., Agapiou, J., Leibo, J., and Gruslys, A. Deep q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

Huang, J. and Jiang, N. From importance sampling to doubly robust policy gradient. In *International Conference on Machine Learning (ICML)*, 2020.

Jaksch, T., Ortner, R., and Auer, P. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(51):1563–1600, 2010. URL http://jmlr.org/papers/v11/jaksch10a.html.

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

Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Johannink, T., Bahl, S., Nair, A., Luo, J., Kumar, A., Loskyll, M., Ojea, J. A., Solowjow, E., and Levine, S. Residual reinforcement learning for robot control. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6023–6029. IEEE, 2019.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. MOReL: Model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Kostrikov, I., Fergus, R., Tompson, J., and Nachum, O. Offline reinforcement learning with fisher divergence critic regularization. In *International Conference on Machine Learning (ICML)*, 2021.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations (ICLR)*, 2022a.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=68n2s9ZJWF8.

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

Kumar, A., Gupta, A., and Levine, S. Discor: Corrective feedback in reinforcement learning via distribution correction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020a.

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

Laroche, R., Trichelair, P., and Combes, R. T. D. Safe policy improvement with baseline bootstrapping. In *International Conference on Machine Learning (ICML)*, 2019.

Lee, S., Seo, Y., Lee, K., Abbeel, P., and Shin, J. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, pp. 1702–1712. PMLR, 2022.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Lu, Y., Hausman, K., Chebotar, Y., Yan, M., Jang, E., Herzog, A., Xiao, T., Irpan, A., Khansari, M., Kalashnikov, D., and Levine, S. AW-opt: Learning robotic skills with imitation and reinforcement at scale. In *5th Annual Conference on Robot Learning*, 2021.

Mao, Y., Wang, C., Wang, B., and Zhang, C. MOORe: Model-based offline-to-online reinforcement learning. *arXiv preprint arXiv:2201.10070*, 2022.

Munos, R. Error bounds for approximate value iteration. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2005.

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

O'Donoghue, B., Osband, I., Munos, R., and Mnih, V. The uncertainty Bellman equation and exploration. In *International Conference on Machine Learning (ICML)*, 2018.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. Deep exploration via bootstrapped DQN. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Peng, X. B., Kumar, A., Zhang, G., and Levine, S. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Reddy, S., Dragan, A. D., and Levine, S. SQIL: Imitation learning via reinforcement learning with sparse rewards. *arXiv preprint arXiv:1905.11108*, 2019.

Siegel, N., Springenberg, J. T., Berkenkamp, F., Abdolmaleki, A., Neunert, M., Lampe, T., Hafner, R., Heess, N., and Riedmiller, M. Keep doing what worked: Behavior modelling priors for offline reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2020.

Silver, T., Allen, K., Tenenbaum, J., and Kaelbling, L. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

Takuma Seno, M. I. d3rlpy: An offline deep reinforcement library. In *NeurIPS 2021 Offline Reinforcement Learning Workshop*, December 2021.

Wu, Y., Tucker, G., and Nachum, O. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J. Y., Levine, S., Finn, C., and Ma, T. MOPO: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Zhang, R., Dai, B., Li, L., and Schuurmans, D. Gendice: Generalized offline estimation of stationary values. In *International Conference on Learning Representations (ICLR)*, 2020.

Zhao, Y., Boney, R., Ilin, A., Kannala, J., and Pajarinen, J. Adaptive behavior cloning regularization for stable offline-to-online reinforcement learning. In *Offline Reinforcement Learning Workshop at Neural Information Processing Systems*, 2021.

Zheng, Q., Zhang, A., and Grover, A. Online decision transformer. In *International Conference on Machine Learning (ICML)*, 2022. https://proceedings.mlr.press/v162/zheng22c/zheng22c.pdf.

A. Algorithm Details

The pseudo-code of the offline, alignment, and online phases is provided in Algorithm 1, 2, and 3, respectively.

Table 5: Learning in three phases for offline-to-online RL with actor-critic alignment

Phase	Learnable Component	Description
Offline training	$\pi_\theta(a\|s), Q_\mu(s,a)$ $Z_\psi(s)$ (practically), α	i) Run offline RL to train $\pi_\theta(a\|s)$ and $Q_\mu(s,a)$. ii) Fit the baseline function $Z_\psi(s)$ with Eq. (8).
Actor-critic alignment	$Z_\psi(s)$ (conceptually)	Initialize $R_\phi(s,a)$ with $Z_\psi(s)$, online critic is parameterized as $Q_\phi(s,a) := \log \pi_{\theta_0}(a\|s) + R_\phi(s,a)$.
Online training	$\pi_\theta(a\|s), R_\phi(s,a), \alpha$	Run ACA (Algorithm 3) with offline trained policy $\pi_\theta(a\|s)$ and aligned critic $Q_\phi(s,a) \coloneqq \log \pi_{\theta_0}(a\|s) + R_\phi(s,a)$.

Although the baseline function $Z_\psi(s)$ is introduced in the step of actor-critic alignment that follows offline training, in practice, we interlace its update with the offline training updates for improved optimization and efficiency.

Algorithm 1 Offline SAC+ML

Initialize parameters $\theta, \alpha, \psi_i, \mu_i, \bar{\mu}_i$ for $i \in \{1,2\}$
for each iteration **do**
  sample mini-batch from dataset $\mathcal{D}$
  update α with Eq. (6)
  update μ_i with Eq. (5) for $i \in \{1,2\}$
  update θ with Eq. (4)
  update ψ_i with Eq. (8) for $i \in \{1,2\}$
  $\bar{\mu}_i \leftarrow \tau\mu_i + (1-\tau)\bar{\mu}_i$ for $i \in \{1,2\}$
end for

Algorithm 2 Actor-critic Alignment

Require: $\theta, \alpha, \psi_i, \mu_i, \bar{\mu}_i$, for $i \in \{1,2\}$
Initialize parameters $\phi_i, \bar{\phi}_i$, for $i \in \{1,2\}$
Set $R_{\phi_i}(s,a) \leftarrow Z_{\psi_i}(s)$, for $i \in \{1,2\}$
Copy $\bar{\phi}_i \leftarrow \phi_i$
Copy $\theta_0 \leftarrow \theta$
Reset $\alpha \leftarrow 1$
Delete μ_i and $\bar{\mu}_i$, for $i \in \{1,2\}$

A.1. Optimization of Z_{ψ_i}

Since the alignment objective (8) needs to access offline data, we blended it into the offline training as shown in the second last step of Algorithm 1. It is noteworthy that this is only for the convenience of implementation, and the ψ_i values do *not* have any influence on SAC+ML training itself. Conversely, the optimized value of ψ_i provides a good initialization for a standalone optimization of objective (8). In practice, we observed that the ψ_i found from offline training is good enough, and we just directly used them to initialize the online critic R_{ϕ_i}.

A.2. Techniques to Stabilize Online Learning

We propose β-clipping trick and critic interpolation to achieve better empirical performance. As $\log \pi_{\theta_0}$ is unbounded below, the training can be numerically unstable, β-clipping trick bounds the term to stabilize training. And critic interpolation gives the flexibility to balance between safety transfer and policy improvement.

A.2.1. β-CLIPPING TRICK

As in the course of online learning, the magnitude of $|\log \pi_{\theta_0}|$ can sometimes be an/several order larger than R_ϕ, which leads to very instable critic training. Given $\pi_{\theta_0}(a|s)$ is parameterized by squashed Gaussian distribution $N(\mu_s, \sigma_s^2)$, we clip the $\log \pi_{\theta_0}$ term, as follows

$$\textsc{Clip}_\beta(\log \pi_{\theta_0}(a|s)) \coloneqq \text{SoftPlus}\Big(\log \pi_{\theta_0}(a|s) - C_\beta(s)\Big) + C_\beta(s) \tag{12}$$

$$\text{where} \quad C_\beta(s) = \min\left\{\log \pi_{\theta_0}(\mu_s - \beta\sigma_s|s), \log \pi_{\theta_0}(\mu_s + \beta\sigma_s|s)\right\}. \tag{13}$$

Algorithm 3 Online training

Require: $\theta, \theta_0, \alpha, \phi_i, \bar{\phi}_i$ from Algo. 2
  if $\mathcal{D}$ is accessible **then**
    Initialize replay buffer $\mathcal{B}$ with top N trajectories
  else
    Initialize replay buffer $\mathcal{B} \leftarrow \varnothing$
  end if
  for each iteration **do**
    sample a mini-batch from buffer $\mathcal{B}$
    update α with Eq. (6)
    update ϕ_i with Eq. (21) for $i \in \{1, 2\}$
    update θ with Eq. (20)
    $\bar{\phi}_i \leftarrow \tau\phi_i + (1-\tau)\bar{\phi}_i$
  end for

Here, β is a hyper-parameter, and $\text{SoftPlus}(x) = \log(1+\exp(x))$. Essentially it clips $\log \pi_{\theta_0}(a|s)$ at $C_\beta(s)$. This $\textsc{Clip}_\beta(\cdot)$ operator bounds the $\log \pi_{\theta_0}$ term in a reasonable range, and also requires minimal tuning of hyper-parameter, see Section D.4 for details. Using $\textsc{Clip}_\beta$, we define Q_ϕ^β as

$$Q_\phi^\beta(s,a) \coloneqq \textsc{Clip}_\beta(\log \pi_{\theta_0}(a|s)) + R_\phi(s,a). \tag{14}$$

Now, the clipped online actor/critic updates can be summarized by

$$\mathcal{L}_\pi^\beta(\theta, \mathbf{d}) = \mathbb{E}_{s\sim\mathbf{d}}\mathbb{E}_{a\sim\pi_\theta}\left[\alpha \log \pi_\theta(a|s) - Q_\phi^\beta(s,a)\right], \tag{15}$$

$$\mathcal{L}_Q^\beta(\phi_i, \mathbf{d}) = \mathbb{E}_{(s,a,r,s',d)\sim\mathbf{d}}\left[\left(Q_{\phi_i}^\beta(s,a) - y(r,s',d)\right)^2\right], \tag{16}$$

$$y(r,s',d) = r + \gamma(1-d)\mathbb{E}_{a'\sim\pi_\theta(\cdot|s')}\left[Q_\phi^\beta(s',a') - \alpha\log\pi_\theta(a'|s')\right]. \tag{17}$$

A.2.2. Critic interpolation

At the initial phase of online training, $\textsc{Clip}_\beta(\log \pi_{\theta_0}(a|s))$ dominates the actor update, safeguarding the policy. As training proceeds, R_ϕ grows to overcome the barrier and starts to improve the policy. Ideally, we wish to finely control such a junction so that the safety of O2O transition does not excessively slow down the policy improvement. To this end, we introduce an interpolation between closed-form initialized critic and restriction-free critic. We call it critic interpolation, which can be written as

$$Q_\phi^{k,\beta}(s,a) \coloneqq k\Big(\underbrace{\textsc{Clip}_\beta\big(\log\pi_{\theta_0}(a|s)\big) + R_\phi(s,a)}_{\text{closed-from initialized critic}}\Big) + (1-k)\quad \underbrace{R_\phi(s,a)}_{\text{restriction-free critic}} \tag{18}$$

$$= k \times \textsc{Clip}_\beta\big(\log\pi_{\theta_0}(a|s)\big) + R_\phi(s,a). \tag{19}$$

We set $k=1$ at $t=0$ to assert closed-form initialization. Then we linearly decay k during the course of online training, allowing a transition from closed-form initialization to free SAC update. The detailed decaying rate can be found in Appendix E.5.

A.3. Concluded Online Training

Our final online update rules are summarized as follows:

$$\mathcal{L}_\pi^{\text{online}}(\theta, \mathbf{d}) = \mathbb{E}_{s\sim\mathbf{d}}\mathbb{E}_{a\sim\pi_\theta}\left[\alpha \log \pi_\theta(a|s) - Q_\phi^{k,\beta}(s,a)\right] \tag{20}$$

$$\mathcal{L}_Q^{\text{online}}(\phi_i, \mathbf{d}) = \mathbb{E}_{(s,a,r,s',d)\sim\mathbf{d}}\left[\left(Q_\phi^{k,\beta}(s,a) - y(r,s',d)\right)^2\right] \tag{21}$$

$$y(r,s',d) = r + \gamma(1-d)\mathbb{E}_{a'\sim\pi_\theta(\cdot|s')}\left[Q_\phi^{k,\beta}(s',a') - \alpha\log\pi_\theta(a'|s')\right]. \tag{22}$$

B. Demonstration of Motivation

B.1. Illustration Distribution Shift

Figure 9 shows the histogram of the ℓ_1 norm of state vectors from the hopper-random-v2, hopper-medium-v2 and hopper-expert-v2 datasets. Clearly, there is a distribution shift. So we can obtain out-of-distribution samples (with respect to the medium/expert dataset trained models) by sampling from random dataset.

The distributional shift justifies the design of our actor-only experiments where we used another offline dataset to simulate distributional shift so that one could ablate the factor of data collection.

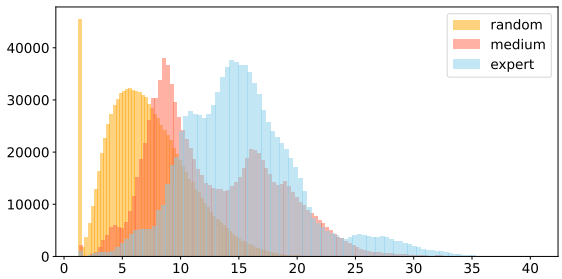


Figure 9: Histogram of ℓ_1 norm of state vectors in hopper random, medium and expert datasets.

B.2. More Actor-Only Experiments

We show the same actor-only experiments, that were presented in Sec 4, with more tasks. As a reminder, the actor-only experiments details are

- Load SAC+ML/CQL models (actor π and critic Q) pre-trained on medium/expert level
- **Disable** critic update, data collecting, etc. (In other words, only keep actor and temperature update)
- Use SAC style actor/temperature updates for actor-only updates
- Run actor-only updates on random dataset to simulate OOD data
- Run total 10k steps and evaluate its performance every 100 steps

to test how different offline/aligned Q-functions affects the actor update.

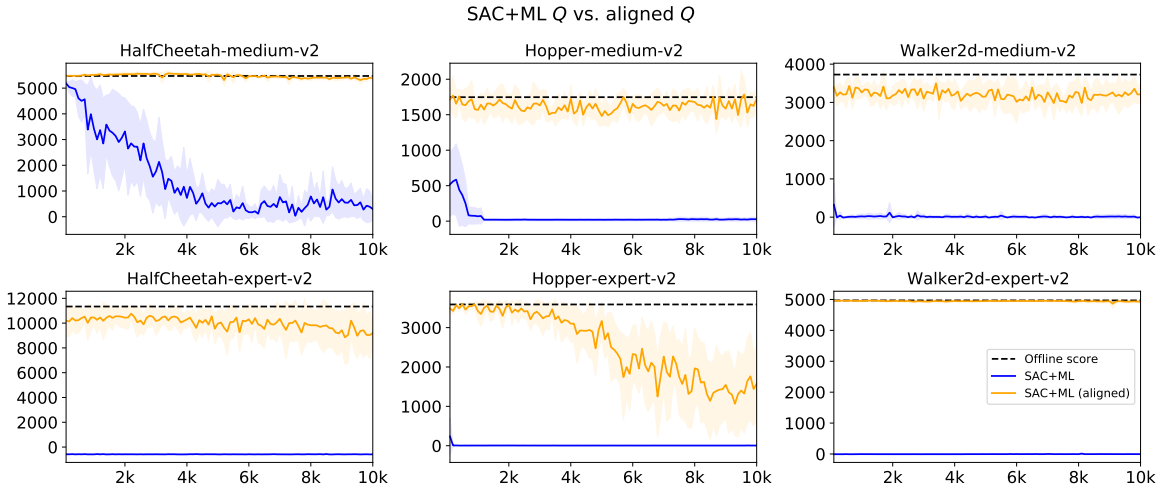


Figure 10: Actor only experiments with aligned Q-function vs. SAC+ML Q-function. Sub-title indicates where π and Q are loaded from, i.e. Hopper-expert-v2 means π and Q are loaded from model pre-trained from Hopper-expert-v2 dataset, while all actor-only online updates are made on random level dataset to simulate OOD data.

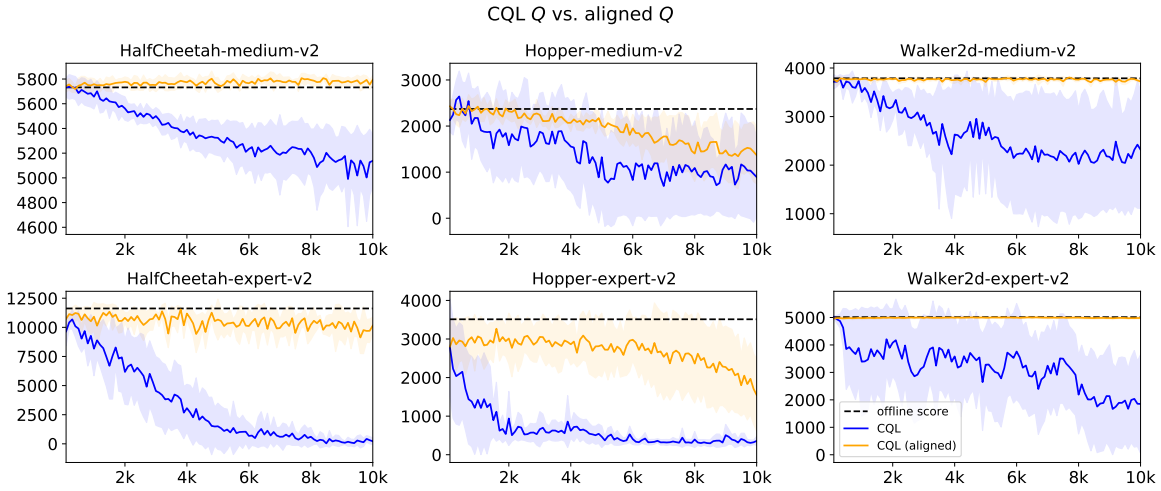


Figure 11: Actor only experiments with aligned Q-function vs. CQL Q-function. Sub-title indicates where π and Q are loaded from, i.e. Hopper-expert-v2 means π and Q are loaded from model pre-trained from Hopper-expert-v2 dataset, while all actor-only online updates are made on random level dataset to simulate OOD data.

It can be observed that, in our controlled actor-only experiments, our aligned Q-functions are able to retain offline scores for most of tasks while CQL/SAC+ML Q-function failed.

C. Additional Comparisons

C.1. SAC+ML vs. TD3+BC

We would like to emphasize that our goal is not to propose a stronger offline RL method. Table 6 is presented to show that our SAC+ML modification performs comparably to the original offline method, TD3+BC.

TD3+BC results in Table 6 were copied from Appendix C.3 of their paper (Fujimoto & Gu, 2021). The evaluation protocol is identical to theirs: (1) all experiments were done in D4RL-v2 datasets; (2) and the results reported were from the last evaluation step, averaged over 5 random seeds.

Table 6: SAC+ML vs. TD3+BC

Dataset	Environment	TD3+BC	SAC+ML
Random	HalfCheetah	11.0±1.1	17.3±2.7
	Hopper	8.5±0.6	7.9±0.3
	Walker2d	1.6±1.7	3.4±2.1
Medium	HalfCheetah	48.3±0.3	46.3±0.2
	Hopper	59.3±4.2	54.3±3.4
	Walker2d	83.7±2.1	81.2±1.6
Medium-Replay	HalfCheetah	44.6±0.5	42.5±1.7
	Hopper	60.9±18.8	48.8±20.4
	Walker2d	81.8±5.5	62.2±4.9
Medium-Expert	HalfCheetah	90.7±4.3	93.5±4.0
	Hopper	98.0±9.4	93.1±7.8
	Walker2d	110.1±0.5	108.1±1.6
Expert	HalfCheetah	96.7±1.1	93.6±0.8
	Hopper	107.8±7	110.9±1.6
	Walker2d	110.2±0.3	108.2±0.3
	Total	1013.2	971.3

C.2. Additional Comparison with AWAC

Nair et al. (2020) reported their online fine-tuning results with 500k online steps. Due to time constraint, we ran all experiments with 100k online steps. We however trained SAC→ACA for 500k steps to enable further comparison with the results reported by Nair et al. (2020).

As shown in Table 7, ACA achieves higher final scores at 500k online episodes and has similar improvement δ compared to AWAC, while AWAC has more room for improvement as AWAC (offline) generally under-performs to SAC+ML, especially on medium-expert and expert level datasets.

Table 7: Additional comparison with AWAC. * indicates numbers duplicated from Nair et al. (2020).

Env	Dataset	AWAC* (offline)	AWAC* (online)	δ_{awac}	SAC→ACA (offline)	SAC→ACA (online)	δ_{ours}
HalfCheetah	Random	2.2	52.9	50.7	17.29	96.42±3.11	79.13
	Medium	37.4	41.1	3.7	46.33	87.55±2.33	41.22
	Med.-Expert	36.8	41.0	4.2	93.53	94.83±0.51	1.30
	Expert	78.5	105.6	27.1	93.59	94.46±0.87	0.87
Hopper	Random	9.6	62.8	53.2	7.93	94.69±23.90	86.76
	Medium	72.0	91.0	19.0	54.31	103.35±0.23	49.05
	Med.-Expert	80.9	111.9	31.0	93.08	109.09±2.25	16.01
	Expert	85.2	111.8	26.6	110.88	111.02±0.58	0.14
Walker2d	Random	5.1	11.7	6.6	3.36	76.34±8.18	72.98
	Medium	30.1	79.1	49.0	81.16	88.65±3.22	7.49
	Med.-Expert	42.7	78.3	35.6	108.12	112.10±0.71	3.98
	Expert	57.0	103.0	46.0	108.21	105.82±7.92	-2.39
	Total	537.5	890.2	352.7	817.79	1174.33	356.53

C.3. Results versus ODT and IQL

Table 8: Comparing SAC→ACA with online decision transformer (ODT) and implicit Q-learning (IQL).

Dataset	Environment	IQL(offline)	IQL(200k)	δ_{IQL}	ODT(offline)	ODT(200k)	δ_{ODT}	SAC+ML	ACA(200k)	δ_{ACA}
Medium	HalfCheetah	47.37±0.29	47.41±0.15	0.04	42.72±0.46	42.16±1.48	-0.56	46.40±0.30	72.67±3.01	26.28
	Hopper	63.81±9.15	66.79±4.07	2.98	66.95±3.26	97.54±2.10	30.59	56.93±4.12	99.32±7.82	42.39
	Walker2d	79.89±3.06	80.33±2.33	0.44	72.19±6.49	76.79±2.30	4.60	79.36±2.25	76.05±20.57	-3.30
Med.-Replay	HalfCheetah	44.10±1.14	44.14±0.3	0.04	39.99±0.68	40.42±1.61	0.43	42.18±0.53	64.29±2.97	22.11
	Hopper	92.13±10.43	96.23±4.35	4.10	86.64±5.41	88.89±6.33	2.25	49.25±6.08	103.17±3.08	53.92
	Walker2d	73.67±6.37	70.55±5.81	-3.12	68.92±4.79	76.86±4.04	7.94	63.20±10.12	82.09±27.66	18.89
Total (w/o hopper-mr)		308.84	309.22	0.38	290.77	333.77	43.00	288.06	394.41	106.35
Locomotion Total (all)		400.97	405.45	4.48	377.41	422.66	45.25	337.31	497.58	160.27
antmaze-umaze		87.1±2.81	89.5±5.43	2.4	53.1±4.21	88.5±5.88	35.4	58.0±14.70	93.0±11.87	35.0
antmaze-umaze-diverse		64.4±8.95	56.8±6.42	-7.6	50.2±5.69	56.0±5.69	7.99	40.0±11.83	67.0±30.68	27.0
AntMaze Total		151.5	146.3	-5.2	103.3	144.5	43.39	98.0	160.0	62.0

Locomotion. As shown in Table 8, for almost all medium and medium-replay tasks, our SAC→ACA outperforms ODT and IQL in both final performance and performance increase (δ). We also note that ODT(offline) and IQL(offline) outperform SAC+ML in the hopper-medium-replay task by a large margin, which leaves our approach more room to improve. Therefore, we made the same comparison by excluding the hopper-medium-replay task. In this case, ODT, IQL and ours were initialized from roughly the same performance, and ours still outperforms ODT/IQL in both total final performance and total performance increase.

AntMaze. Overall, SAC→ACA outperforms ODT and IQL on AntMaze although ACA shows higher standard deviation, especially in umaze-diverse. Figure 12 shows the training curves of our SAC→ACA on AntMaze.

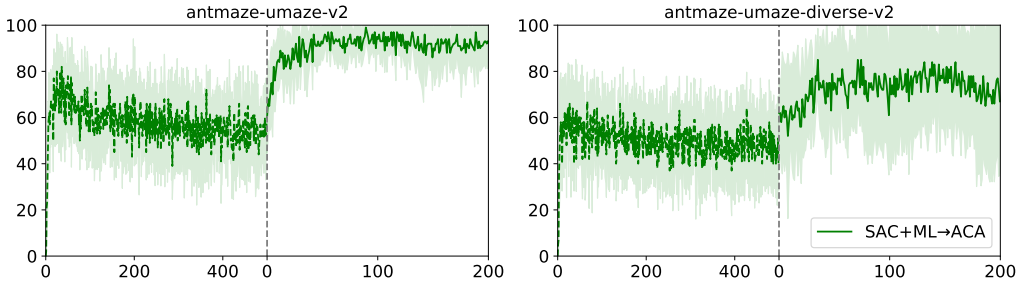


Figure 12: D4RL scores of AntMaze tasks.

C.4. Online Training without Offline Data

The distributional shift issue would clearly be severer when offline data are not accessible during the online phase. To be more conservative, we therefore set $\beta_{w/o} = 1.5\beta_{w/}$ for experiments without offline data, excluding random and medium levels as both used no offline data for our main results already. ($\beta_{w/}$ denotes the hyper-parameter we used for our main results, see Table 10 for details.) All other hyper-parameters remained unchanged.

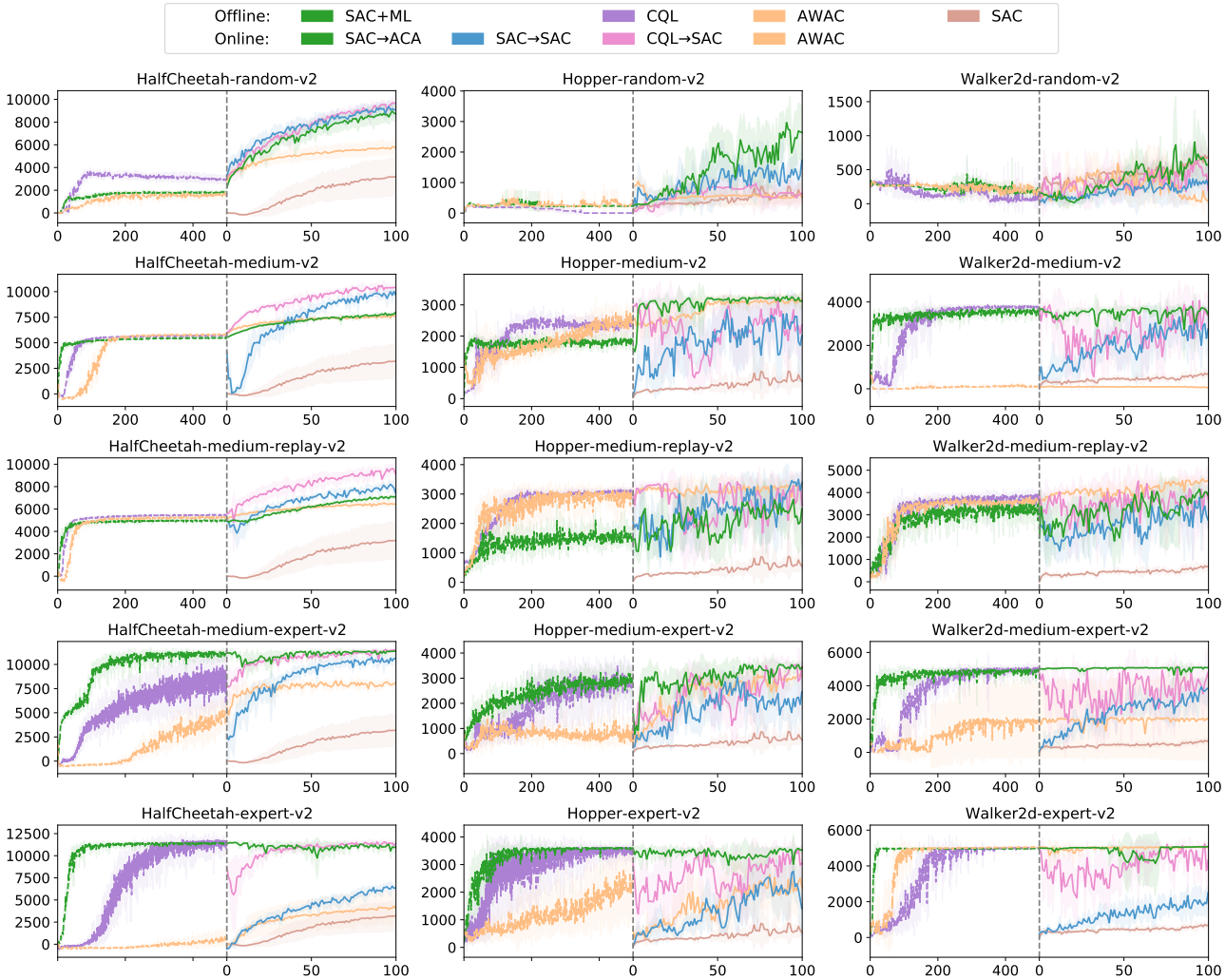


Figure 13: When offline data are not accessible, vs. other baselines

C.5. Fine-tuning with Burn in Period

In addition, we run SAC→SAC and CQL→SAC with 50k "burn in" episodes (Silver et al., 2018), where the agent interacts with the environment but only updates the critic, following in our experiment settings in Section 6. We observed that the initial performance drop upon finetuning is still noticeable in Figure 14 and Figure 15.

Hypothesis: Note that Silver et al. (2018) considers initialization from a hand-engineered policy or model-predictive controller. So the initial Q-function is not learned from offline data. Therefore it could admit high Bellman error. A "burn-in" phase will help improving the initial Q-function in terms of Bellman error with additional data collected online, as they terminate the burn-in phase when the Bellman error falls below a certain threshold. However, in the offline RL setting, there are already plenty offline data to allow the critic to reach convergence with considerably low Bellman error. And low Bellman error on offline data does not guarantee smooth O2O transfer as the key obstacle in O2O setting is the distributional shift. Additional online samples collected during "burn-in" are likely to be in-distribution, which might not contribute to tackling the distributional shift problem.

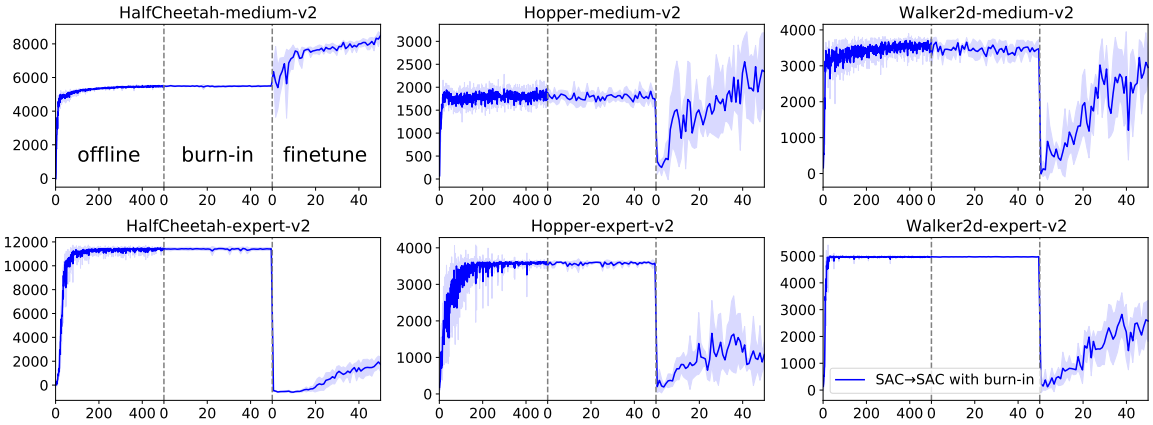


Figure 14: SAC→SAC with 50k "burn in" episodes.

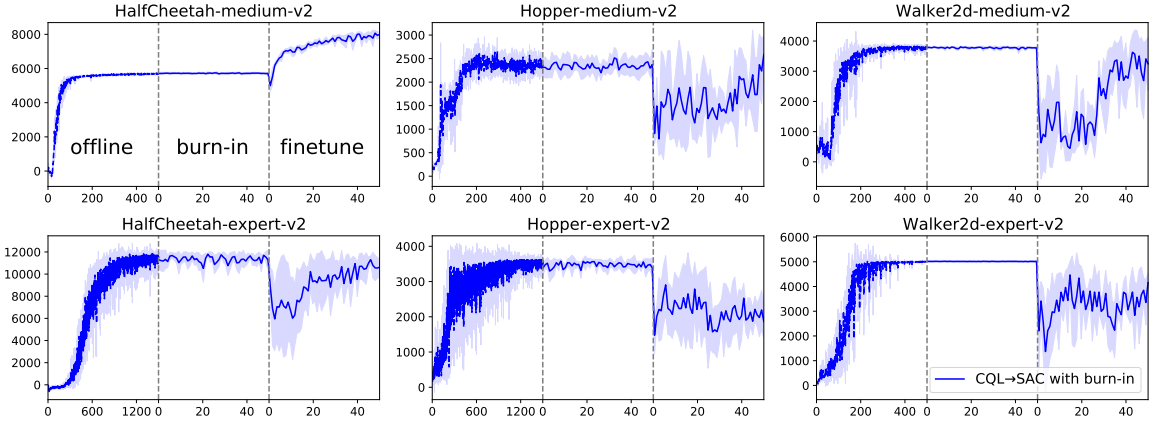


Figure 15: CQL→SAC with 50k "burn in" episodes.

D. Ablation Studies

D.1. Alternative Q Reparameterizations

Here we write out the detailed formula of the critic objective in the two ablation studies in Section 6.5.

Ablation 1: BC only

$$\mathcal{L}_Q^{\text{ablation1}}(\phi_i, \mathbf{d}) = \mathbb{E}_{(s,a,r,s',d)\sim\mathbf{d}}\left[\Big(R_\phi(s,a) - y(r,s',d)\Big)^2\right] \tag{23}$$

$$y(r,s',d) = r + \gamma(1-d)\mathbb{E}_{a'\sim\pi_\theta(\cdot|s')}\Big(R_\phi(s',a') - \alpha\log\pi_\theta(a'|s')\Big) \tag{24}$$

where $\log\pi_{\theta_0}$ is no longer part of the critic update but still exists in actor update as Eq. (11) and R_ϕ is initialized by offline critic Q_μ, therefore, one could consider it is regular SAC style actor-critic with a behaviour cloning regularizer in actor update.

Ablation 2: $\mathbf{R}_\phi$ init by $\mathbf{Q}_\mu$

$$\mathcal{L}_Q^{\text{ablation2}}(\phi_i, \mathbf{d}) = \mathbb{E}_{(s,a,r,s',d)\sim\mathbf{d}}\left[\Big(Q_\phi^{k,\beta}(s,a) - y(r,s',d)\Big)^2\right] \tag{25}$$

$$y(r,s',d) = r + \gamma(1-d)\mathbb{E}_{a'\sim\pi_\theta(\cdot|s')}\Big(Q_\phi^{k,\beta}(s',a') - \alpha\log\pi_\theta(a'|s')\Big) \tag{26}$$

where actor-critic updates are identical to our proposal, however R_ϕ is initialized by offline critic Q_μ instead of baseline Z_ψ to show the importance of the baseline Z_ψ.

In addition, we show our ablation study with all medium-level tasks

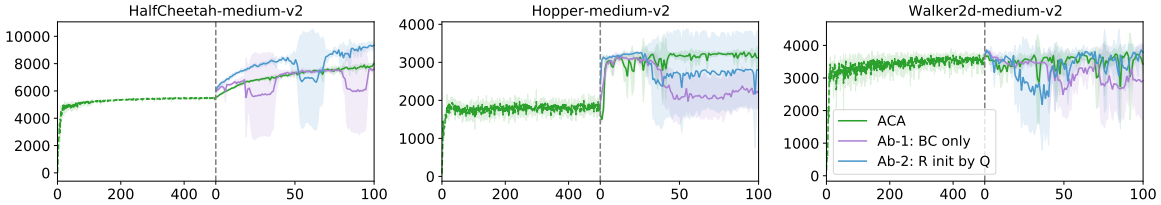


Figure 16: Ablation study: We keep the actor update same as Eq (11), we however change the critic update to show that the re-parameterized Q-function is critical. **BC only** (ablation 1): R_ϕ is seen as the online critic and is initialized by Q_μ. Critic updates are made as regular SAC critic update without our re-parameterization. It therefore can be seen as SAC with behavior cloning regularized actor update. $\mathbf{R}_\phi$ **init by** $\mathbf{Q}_\mu$ (ablation 2): We keep the ACA framework but initialize R_ϕ by Q_μ instead of Z_ψ to show the importance of the baseline.

Figure 16 shows that on tasks vulnerable to transfer risk such as hopper-medium and walker-medium (second and third subplots), the two ablation alternatives suffer clear performance drop due to the attributed error in the offline trained Q-functions. However, some tasks can be less vulnerable. For example, on halfcheetah-medium, Figure 5 shows that SAC→SAC (green→blue) only suffers a small amount of drop, although it employs no mechanism to combat distribution shift. In such a task, the two ablation alternatives remain competitive to no surprise (first subplot).

D.2. Reparameterization with θ^- (delayed parameters)

We in addition tested the alternative choice of $Q_\phi = \log \pi_{\theta^-} + R_\phi$ on all medium-level tasks. It still fails in all medium-level task regardless the choice of τ and the smallest $\tau = 0.0005$ still performs the best, supporting our choice π_{θ_0} over the alternative.

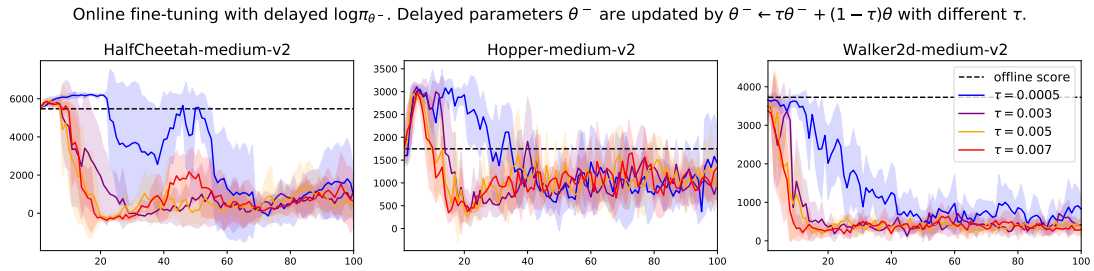


Figure 17: Online fine-tuning with delayed $\log \pi_{\theta^-}$ instead of $\log \pi_{\theta_0}$. Delayed parameters θ^- are updated by $\theta^- \leftarrow \tau\theta^- + (1-\tau)\theta$ with different values of τ.

D.3. Different Initialization

Similarly to our experiments in Section 6.3, we show additional results on halfcheetah-medium and walker2d-medium. It can be observed that ACA's performance does not affected much by different offline training approaches. While BR admits huge performance drop when initializing from SAC+ML in hopper and walker2d.

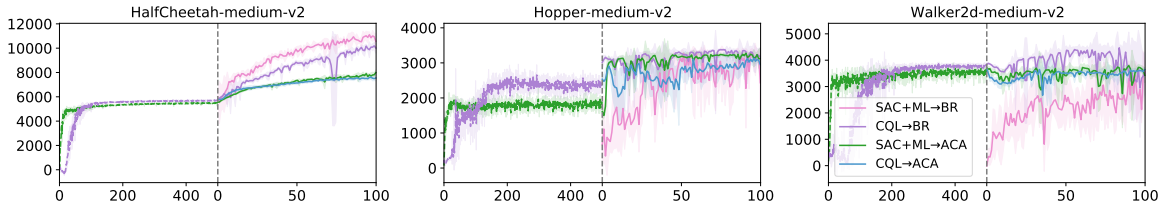


Figure 18: ACA and BR initialized from different offline methods. ACA could achieve similar performance while initializing from both SAC+ML/CQL. BR requires CQL initialization.

D.4. Sensitivity on β

Figure 19 shows that the performance of SAC→ACA is not very sensitive to the choice of β.

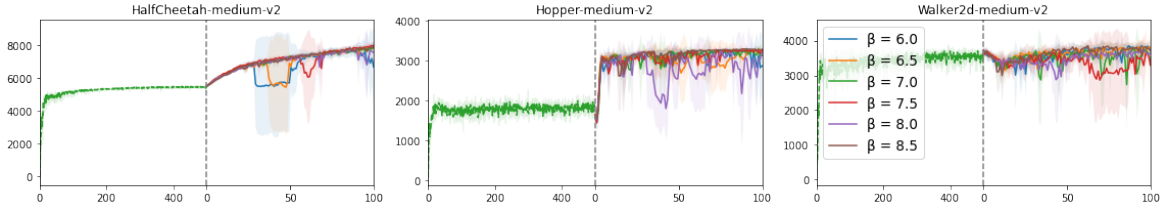


Figure 19: Results for different β

E. Implementations

Overall, all our implementations are from or based on d3rlpy (Takuma Seno, 2021), a popular RL library that specialized for offline RL. Using the same lib helps us to minimize the impact of implementation difference. Many of our baselines (see Table 11) are implemented upon SAC, with changes proposed in their original papers, respectively.

E.1. General implementation details

Evaluation protocol: All offline/online experiments ran 5 random seeds. We ran all offline algorithms for 500 episodes with 1000 mini-batches each, and all online experiments for 100 episodes with 1000 environment interactions each. After each episode, we conducted 10 evaluations and computed the average return. Results reported are mean and std of average returns, over 5 random seeds.

Choice of offline checkpoints: Evaluating in the offline phase, in fact, requires online interactions. Therefore we do **not** pick the best-performed checkpoints. Instead, we use the last checkpoints as our initialization models, for online.

Squashed Gaussian: For all methods with stochastic policies, we parameterized their policies by unimodal Gaussian, and applied the squashed Gaussian trick (Haarnoja et al., 2018) to bound the range of action to $[-1, 1]$.

One exception is ACA on AntMaze, where we used Gaussian parameterization rather than Squashed Gaussian with a state-independent standard deviation. This choice is same as Kostrikov et al. (2022b) and AntMaze tasks might benefit from this parameterization as discussed in this GitHub issue: `https://github.com/takuseno/d3rlpy/issues/171`.

Buffer initialization: We followed the instructions in AWAC and BR papers on initializing online replay buffers. For AWAC, we added all transitions in $\mathcal{D}$ to the buffer $\mathcal{B}$. And for BR, we refer to their original implementation at this URL for details. For SAC→SAC and CQL→SAC, we added all transitions in $\mathcal{D}$ to the buffer $\mathcal{B}$ as well, as there is no explicit instructions or common protocols. All replay buffer sizes were set to be 1e6, unless specified in the Appendix E.5.

E.2. Offline

AWAC and CQL: We used d3rlpy implementations for AWAC and CQL.

SAC+ML: Our SAC+ML implementation was adapted from d3rlpy's TD3+BC implementation, with changing the actor update rule to Eq. (4), and adding the learning of baseline Z_ψ.

E.3. Online

Training details for online: For all methods, we made a temperature (if applicable), a critic, and an actor update after every environmental interaction, if there were enough transitions (i.e. more than batch size) in the replay buffer. Target networks were all updated in a Polyak averaging fashion, where the step size $\tau = 0.005$ for all experiments. See Section E.5 for more hyper-parameter details. And online results, reported in tables, were also using the last checkpoints instead of best-performed ones.

SAC: We used d3rlpy implementation for SAC.

SAC→SAC and CQL→SAC: We simply loaded offline-trained SAC+ML and CQL, respectively, and then ran SAC online.

BR: We adapted all parts that related to the prioritized replay from the official BR implementation, to a d3rlpy SAC implementation base, as the original BR paper also run SAC online.

ACA (ours:) Implementation of our approach can be found at https://github.com/ZishunYu/ACA. In addition to Algorithm 3, we also did gradient norm clipping to actor updates, which is commonly used in RL implementations.

E.4. AntMaze

Following the experimental details of Kostrikov et al. (2022b) we subtract 1 from rewards for all AntMaze tasks.

E.5. Hyper-parameters

Table 9: Specific hyper-parameters for different baselines. Please refer to the original paper for the meaning of hyper-parameter names.

Algo.	Hyper-param name	Value
SAC+ML	ω	30
AWAC	λ	1.0
CQL	conservative weight	10
	# of actions sampled	10
BR	offline buffer size	2.5e6
	online buffer size	2.5e5
	density ratio estimation network arch.	$[\lvert\mathcal{S}\rvert + \lvert\mathcal{A}\rvert, 256, 256, 1]$
	density ratio estimation network temp	5
	ρ	0.75
ACA	π grad norm clip	0.25

Table 10: Hyper-params used for our main results reported in section 6.1. $x \xrightarrow{y} z$ represents that k decays from x to z using y episodes.

	hyper-param	HalfCheetah	Hopper	Walker2d
Random	N (# of init trajs)	0		
	β (β-clipping)	7		
	k (interpolation)	$1 \xrightarrow{10} 0$		
Medium	N	0		
	β	7		
	k	$1 \xrightarrow{20} 0.5$		
Medium-replay	N	50		
	β	7		
	k	$1 \xrightarrow{20} 0.5$		
Medium-expert	N	50		
	β	15		
	k	$1 \xrightarrow{\text{N/A}} 1$		
Expert	N	50		
	β	15		
	k	$1 \xrightarrow{\text{N/A}} 1$		
		umaze and umaze-diverse		
AntMaze	N	0		
	β	3		
	k	$1 \xrightarrow{100} 0.5$		

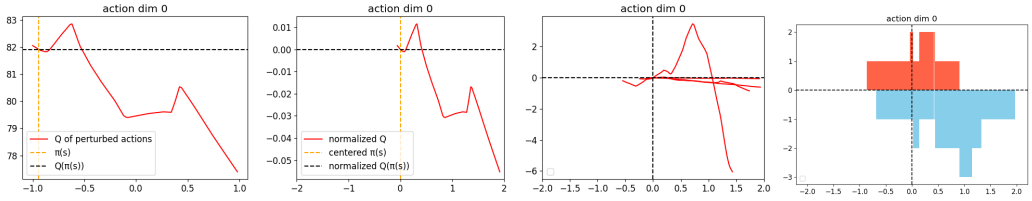
Table 11: General hyper-parameters. ACA and BR stand for SAC→ACA and CQL→BR, respectively.

	CQL	SAC+ML	ACA	SAC→SAC	BR	CQL→SAC	SAC (scratch)	AWAC (off)	AWAC (on)
Phase	offline		online					offline	online
Based on SAC?	Yes							No	
General hyper-params									
π Arch.	$[\lvert\mathcal{S}\rvert + \lvert\mathcal{A}\rvert, 256, 256, 1]$								
Q Arch.	$[\lvert\mathcal{S}\rvert + \lvert\mathcal{A}\rvert, 256, 256, 1]$								
Z Arch.	$[\lvert\mathcal{S}\rvert, 256, 256, 1]$		N/A for online						
# Q nets	2								
# Z nets	2		N/A						
τ (Polyak avg.)	0.005								
Activation	ReLU								
Optimizer	Adam for all								
Adam params	betas = (0.9, 0.999), eps = 1e-8, weight decay = 0								
π lr	1e-4	3e-4	3e-4						
Q lr	3e-4	3e-4	3e-4						
α lr	1e-4	3e-4	3e-4					N/A	
Z lr	3e-4	3e-4	N/A for online						
# episodes	500		100						
# it/ep	1000								
# batch/it	1								
Batch size	256								
Hyper-params for the base SAC impl. (Locomotion)									
Entropy target $\bar{\mathcal{H}}$	$\lvert\mathcal{A}\rvert$							N/A	
Squashed Gaussian	Yes							N/A	
Hyper-params for the base SAC impl. (AntMaze)									
$\bar{\mathcal{H}}$	N/A		$0.5\lvert\mathcal{A}\rvert$	N/A					
Squashed Gaussian	N/A		No	N/A					

F. Further Discussions

F.1. Averaged Version of Figure 4

Figures 21a and 21b provide averaged versions of Figure 4 for in-distribution and out-of-distribution samples, respectively. The approach used to generate these two figures is illustrated in Figure 20.



(a) Perturb action (b) Centralization (c) Multiple samples (d) Aggregation

Figure 20: Demonstration of how Figure 21a and Figure 21b are created. (a) Given a sample (s, a), we perturb a along a dimension to plot $Q(s, \tilde{a})$ and compare $Q(s, \tilde{a})$ to $Q(s, \pi(s))$; (b) We plot $Q(s, \tilde{a})$ with its deviation from $Q(s, \pi(s))$, so that $Q(s, \pi(s))$ is centered at $y = 0$ and $\pi(s)$ is centered at $x = 0$; (c) Such a centralization allows us to place multiple samples (different s) in the same plot, where points above the x-axis correspond to "over-estimated" perturbations; (d) We aggregate multiple samples by counting how many points are above 0. This way, the height of the red part in the bar plot quantifies the fraction of points that are "over-estimated". For an "over-estimated" point (s, a), its x-coordinate stands for the distance between a and the policy favored action $\pi(s)$.

TL;DR: Area of red region represents the fraction of Q-value "over-estimation" compared to $Q(\pi(s))$.

Note: By "over-estimation", we mean for some $a \neq \pi(s)$ such that $Q(s, a) > Q(s, \pi(s))$, which in a certain degree means that the critic Q is "disagreeing" with the policy π.

(a) in-distribution states (b) out-of-distribution states

Figure 21: Quantifying fraction of over-estimated perturbations for in-distribution and out-of-distribution states.

Details: All agents are trained on hopper-medium-v2 dataset. By in-distribution samples, we refer to states drawn from the hopper-medium-v2 dataset. By out-of-distribution samples, we use samples from the hopper-random-v2 dataset. We randomly drew 200 samples per seed, which results in a total of 1000 samples to make each plot.