[APPENDIX] SERA: SAMPLE EFFICIENT REWARD AUGMENTATION IN OFFLINE-TO-ONLINE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

A ETHICAL CLAIM

Despite the potential of offline RL to learn from the static datasets without the necessity to access the online environment, the offline method does not guarantee the optimal policy. Therefore, online fine-tuning is essential for policy improvement. In this study, we propose a novel and versatile reward augmentation framework, named Sample Efficient Reward Augmentation (SERA) which can be seamlessly plugged into various model-free algorithms. We believe our approach is constructive and will enhance the sample efficiency of offline-to-online RL. Additionally, given that SERA is an integrated algorithm, we also believe it can broadly and readily benefit existing algorithms.

B THEORETICAL ANALYSIS

In this section, we provide the supplementary mathematical analysis for SERA.

B.1 ANALYSIS OF APPROXIMATE STATE MARGINAL MATCHING.

Approximate Marginal Matching (ASMM). Given the empirical state distribution $\rho_\pi(\mathbf{s})$ under current empirical policy π and target density $p^*(\mathbf{s})$, the optimization of $\min D_{\mathrm{KL}}(\rho_\pi(\mathbf{s})||p^*(\mathbf{s}))$ is equivalent to Equation 1.

$$\min D_{\mathrm{KL}}(\rho_\pi(\mathbf{s})||p^*(\mathbf{s})) \triangleq \max \mathbb{E}_{\mathbf{s}\sim\rho_\pi(\mathbf{s})}[\log p^*(\mathbf{s}) + \mathcal{H}_\pi[\mathbf{s}]]. \tag{1}$$

ASMM. In section 3.2, due to the absence of an explicit definition for p^*, we propose the concept of implicit SMM, *i.e.*, $\min D_{\mathrm{KL}}(\rho_\pi(\mathbf{s})||p^*(\mathbf{s})) \approx \max \mathbb{E}_{\mathbf{s}\sim\rho_\pi(\mathbf{s})}[\mathcal{H}_\pi[\mathbf{s}]]$. Here we will provide a complementary analysis to assess the feasibility of this method.

Why does ASMM encourage covering the target density? We commence our analysis by expressing $\max \mathbb{E}_{\mathbf{s}\sim\rho_\pi(\mathbf{s})}[\mathcal{H}_\pi(\mathbf{s})]$ in an alternative form:

$$\begin{aligned}
\max \mathbb{E}_{\mathbf{s}\sim\rho_\pi(\mathbf{s})}[\mathcal{H}_\pi(\mathbf{s})] &= \max \mathbb{E}_{\mathbf{s}\sim\rho_\pi(\mathbf{s})}[-\log \rho_\pi(s)] \\
&= \max \int_{\mathbf{s}\sim dom(\rho_\pi)} -\rho_\pi(\mathbf{s}) \log \rho_\pi(\mathbf{s}) d\mathbf{s} \\
&= \max \int_{\mathbf{s}\sim dom(\rho_\pi)} -\rho_\pi(\mathbf{s}) \log (\frac{\rho_\pi(\mathbf{s})}{p^*(\mathbf{s})} \times p^*(\mathbf{s})) d\mathbf{s} \\
&= \max \int_{\mathbf{s}\sim dom(\rho_\pi)} -\rho_\pi(\mathbf{s}) (\log p^*(\mathbf{s}) + \log \frac{\rho_\pi(\mathbf{s})}{p^*(\mathbf{s})}) d\mathbf{s} \\
&= \min \int_{\mathbf{s}\sim dom(\rho_\pi)} \rho_\pi(\mathbf{s}) \log p^*(\mathbf{s}) + \rho_\pi(\mathbf{s}) \log \frac{\rho_\pi(\mathbf{s})}{p^*(\mathbf{s})} d\mathbf{s}
\end{aligned}$$

where $dom(\rho_\pi)$ donates the domain of state space under function ρ_π. Subsequently, we employ a logarithmic inequality, i.e. $1 - \frac{1}{x} \leq \log x \leq x - 1$, to further derive the aforementioned expression:

$$\begin{aligned}\min \int_{\mathbf{s}\sim dom(\rho_\pi)} \rho_\pi(\mathbf{s}) \log p^*(\mathbf{s}) + \rho_\pi(\mathbf{s}) \log \frac{\rho_\pi(\mathbf{s})}{p^*(\mathbf{s})} d\mathbf{s} &\geq \min \int_{\mathbf{s}\sim dom(\rho_\pi)} \rho_\pi(\mathbf{s}) - \frac{\rho_\pi(\mathbf{s})}{p^*(\mathbf{s})} + \rho_\pi(\mathbf{s}) - p^*(\mathbf{s}) d\mathbf{s} \\ &= 2 - \max \int_{\mathbf{s}\sim dom(\rho_\pi(\mathbf{s}))} \frac{\rho_\pi(\mathbf{s})}{p^*(\mathbf{s})} + p^*(\mathbf{s}) d\mathbf{s} \\ &\geq 2 - \max \int_{\mathbf{s}\sim dom(\rho_\pi(\mathbf{s}))} \frac{1}{p^*(\mathbf{s})} + p^*(\mathbf{s}) d\mathbf{s}\end{aligned}$$

It is worth noting that, given that $p^*(\mathbf{s})$ is the fixed target state density and $p^*(\mathbf{s}) \in [0, 1]$ for all $\mathbf{s} \in dom(p^*)$, we have $(\frac{1}{p^*(\mathbf{s})} + p^*(\mathbf{s})) > 0$. Therefore, the process of maximising $\mathbb{E}_{\mathbf{s}\sim\rho_\pi(\mathbf{s})}[\mathcal{H}[\mathbf{s}]]$ is equivalent to maximizing $\int_{\mathbf{s}\sim dom(\rho_\pi(\mathbf{s}))} \frac{1}{p^*(\mathbf{s})} + p^*(\mathbf{s}) d\mathbf{s}$. This leads the domain of $\rho_\pi(\mathbf{s})$(which is initially smaller than the domain of $p^*(\mathbf{s})$ due to limited state exploration at the beginning of state entropy maximization) to cover the domain of $p^*(\mathbf{s})$.

Trade off between ρ_π and p^* when maximizing entropy. We further derivate Equation B.1 and obtained :

$$\begin{aligned}\max \mathbb{E}_{\mathbf{s}\sim\rho_\pi(\mathbf{s})}[\mathcal{H}_\pi(\mathbf{s})] &= \max \mathbb{E}_{\mathbf{s}\sim\rho_\pi(\mathbf{s})}[-\log \rho_\pi(s)] \\ &= \max \int_{\mathbf{s}\sim dom(\rho_\pi)} -\rho_\pi(\mathbf{s}) \log \rho_\pi(\mathbf{s}) d\mathbf{s} \\ &= \max \int_{\mathbf{s}\sim dom(\rho_\pi)} -\rho_\pi(\mathbf{s}) \log (\frac{\rho_\pi(\mathbf{s})}{p^*(\mathbf{s})} \times p^*(\mathbf{s})) d\mathbf{s} \\ &= \max \int_{\mathbf{s}\sim dom(\rho_\pi)} -\rho_\pi(\mathbf{s}) \log p^*(\mathbf{s}) - \rho_\pi(\mathbf{s}) \log \frac{\rho_\pi(\mathbf{s})}{p^*(\mathbf{s})} d\mathbf{s} \\ &= \min \underbrace{\int_{\mathbf{s}\sim dom(\rho_\pi)} \rho_\pi(\mathbf{s}) \log p^*(\mathbf{s})}_{\text{term.1}} d\mathbf{s} + \max \underbrace{\int_{\mathbf{s}\sim dom(\rho_\pi)} -\rho_\pi(\mathbf{s}) \log \frac{\rho_\pi(\mathbf{s})}{p^*(\mathbf{s})}}_{\text{term.2}} d\mathbf{s}\end{aligned}$$

Analysis term.1: We further derive term_1:

At first,

$$\begin{aligned}\mathcal{J}_{\text{term}_1} &= \max \int_{\mathbf{s}\sim dom(\rho_\pi)} \rho_\pi(\mathbf{s}) \log \frac{1}{p^*(\mathbf{s})} d\mathbf{s} \\ &\geq \int_{\mathbf{s}\sim dom(\rho_\pi)} \rho_\pi(\mathbf{s}) \log \frac{\rho_\pi(\mathbf{s})}{p^*(\mathbf{s})} d\mathbf{s} \\ &= D_{\text{KL}}(\rho_\pi(\mathbf{s}) || p^*(\mathbf{s}))\end{aligned} \tag{2}$$

meanwhile, we study $\mathcal{J}_{\text{term}_1} - D_{\text{KL}}(\rho_\pi(\mathbf{s}) || p^*(\mathbf{s}))$.

$$\begin{aligned}&\mathcal{J}_{\text{term}_1} - D_{\text{KL}}(\rho_\pi(\mathbf{s}) || p^*(\mathbf{s})) \\ =& \int_{\mathbf{s}\sim dom(\rho_\pi)} \rho_\pi(\mathbf{s}) \log \frac{1}{p^*(\mathbf{s})} - \rho_\pi \log \frac{\rho_\pi(\mathbf{s})}{p^*(\mathbf{s})} d\mathbf{s} \\ &= \int_{\mathbf{s}\sim dom(\rho_\pi)} \rho_\pi(\mathbf{s}) \log \frac{1}{\rho_\pi(\mathbf{s})} d\mathbf{s} \\ &\leq \int_{\mathbf{s}\sim dom(\rho_\pi)} \rho_\pi(\mathbf{s}) [\frac{1}{\rho_\pi(\mathbf{s})} - 1] d\mathbf{s} \\ &= \int_{\mathbf{s}\sim dom(\rho_\pi)} (1 - \rho_\pi(\mathbf{s})) d\mathbf{s} \\ &\leq \int_{\mathbf{s}\sim dom(\rho_\pi)} d\mathbf{s}\end{aligned} \tag{3}$$

Therefore, $\mathcal{J}_{\mathrm{term}_1} \leq (D_{\mathrm{KL}}(\rho_\pi(\mathbf{s})||p^*(\mathbf{s})) + \int_{\mathbf{s}\sim dom(\rho_\pi)} d\mathbf{s})$ and minimizing term_1 is equivalent to maximizing the KL divergence between $\rho_\pi(\mathbf{s})$ and $p^*(\mathbf{s})$, then push $\rho_\pi(\mathbf{s})$ away from $p^*(\mathbf{s})$.

Analysis term.2: We can observe that term.2 is a form of KL deivergence:

$$\begin{aligned}\mathcal{J}_{\mathrm{term}_2} &= \max \int_{\mathbf{s}\sim dom(\rho_\pi)} -\rho_\pi(\mathbf{s}) \log \frac{\rho_\pi(\mathbf{s})}{p^*(\mathbf{s})} \\ &= \min D_{\mathrm{KL}}(\rho_\pi(\mathbf{s})||p^*(\mathbf{s})),\end{aligned} \tag{4}$$

Thus, optimizeing term.2 is equiv to minimize the KL divergence between $p^*(\mathbf{s})$ and $\rho_\pi(\mathbf{s})$, thereby driving ρ_π approaching $p^*(\mathbf{s})$.

Analysis Summary: In conclusion, based on the Analysis term.1 and Analysis term.2, it can be deduced that the optimization of term 1 makes $\rho_\pi(\mathbf{s})$ away from $p^*(\mathbf{s})$, whereas the optimization of Term 2 facilitates the convergence of $\rho_\pi(\mathbf{s})$ towards $p^*(\mathbf{s})$. Therefore, these two objectives represent a trade-off, offering the advantage of encouraging the agent to approach the target distribution while maintaining its capacity for exploration.

B.2 Mathematics Analysis of SERA Algorithm

In this section, we examine the mathematical viability of the SERA framework, focusing on two key aspects: **1)** Guarantee of Soft policy optimization **2)** Prevention of OOD state actions.

We first introduce the modified soft Q Bellman backup operator, denoted as Equation 5,

$$\mathcal{T}^\pi_{sera} Q\left(\mathbf{s}_t, \mathbf{a}_t\right) \triangleq r\left(\mathbf{s}_t, \mathbf{a}_t\right) + r^{aug}\left(\mathbf{s}_t, \mathbf{a}_t\right) + \gamma \mathbb{E}_{\mathbf{s}_{t+1}\sim p}\left[V\left(\mathbf{s}_{t+1}\right)\right] \tag{5}$$

In this equation, the term $V\left(\mathbf{s}_t\right) = \mathbb{E}_{\mathbf{a}_t\sim\pi}\left[Q\left(\mathbf{s}_t, \mathbf{a}_t\right) - \log \pi\left(\mathbf{a}_t \mid \mathbf{s}_t\right)\right]$ is defined.

Lemma B.1 (Soft Policy Evaluation with SERA.) *Given the modified soft bellman backup operator $\mathcal{T}^\pi_{sera}$ in Equation 5, along with a mapping $Q^0 : \mathcal{S}\times\mathcal{A}\rightarrow\mathbb{R}$ where $|\mathcal{A}| < \infty$. We define an iterative sequence as $Q^{k+1} = \mathcal{T}^\pi Q^k$. It can be shown that when index k tends towards infinity, the sequence Q^k converges to a soft Q-value of π.*

proof. Let us define the SERA reward as follows

$$r^\pi_{sera}\left(\mathbf{s}_t, \mathbf{a}_t\right) \triangleq r\left(\mathbf{s}_t, \mathbf{a}_t\right) + \lambda \operatorname{Tanh}\left(\mathcal{H}\left(\mathbf{s}_t \mid \min\left(Q_{\phi_1}(\mathbf{s}_t, \mathbf{a}_t), Q_{\phi_2}(\mathbf{s}_t, \mathbf{a}_t)\right)\right)\right) + \mathbb{E}_{\mathbf{s}_{t+1}\sim p}\left[\mathcal{H}\left(\pi\left(\cdot \mid \mathbf{s}_{t+1}\right)\right)\right] \tag{6}$$

and rewrite the update rule as

$$Q\left(\mathbf{s}_t, \mathbf{a}_t\right) \leftarrow r^\pi_{sera}\left(\mathbf{s}_t, \mathbf{a}_t\right) + \gamma \mathbb{E}_{\mathbf{s}_{t+1}\sim p, \mathbf{a}_{t+1}\sim\pi}\left[Q\left(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}\right)\right]. \tag{7}$$

Then we can apply mathematical analysis of convergence for policy evaluation as outlined in Sutton & Barto (1998) to prove the result. It is essential to note that the assumption $|\mathcal{A}| < \infty$ is necessary to ensure the boundedness of the SERA reward."

Lemma B.2 (Soft Policy Improvement with SERA) *Let $\pi_{old} \in \Pi$, and let π_{new} be the solution to the minimization problem defined as:*

$$\pi_{\mathrm{new}} = \arg\min_{\pi'\in\Pi} \mathrm{D}_{\mathrm{KL}}\left(\pi'\left(\cdot \mid \mathbf{s}_t\right) \| \frac{\exp\left(Q^{\pi_{\mathrm{old}}}\left(\mathbf{s}_t, \cdot\right)\right)}{Z^{\pi_{\mathrm{old}}}\left(\mathbf{s}_t\right)}\right). \tag{8}$$

Then, it follows that $Q^{\pi_{new}}\left(\mathbf{s}_t, \mathbf{a}_t\right) \geq Q^{\pi_{old}}\left(\mathbf{s}_t, \mathbf{a}_t\right)$ for all $\left(\mathbf{s}_t, \mathbf{a}_t\right) \in \mathcal{S}\times\mathcal{A}$ provided that $|\mathcal{A}| < \infty$.

proof. Starting from Equation 9, which has been established in the work by (Haarnoja et al., 2018), as:

$$\mathbb{E}_{\mathbf{a}_t\sim\pi_{\mathrm{new}}}\left[Q^{\pi_{\mathrm{old}}}\left(\mathbf{s}_t, \mathbf{a}_t\right) - \log \pi_{\mathrm{new}}\left(\mathbf{a}_t \mid \mathbf{s}_t\right)\right] \geq V^{\pi_{\mathrm{old}}}\left(\mathbf{s}_t\right), \tag{9}$$

we proceed to consider the soft Bellman equation, which can be expressed as:

$$\begin{aligned} Q^{\pi_{\mathrm{old}}}\left(\mathbf{s}_t, \mathbf{a}_t\right) &= r\left(\mathbf{s}_t, \mathbf{a}_t\right) + r_{aug}\left(\mathbf{s}_t, \mathbf{a}_t\right) + \gamma \mathbb{E}_{\mathbf{s}_{t+1}\sim p}\left[V^{\pi_{\mathrm{old}}}\left(\mathbf{s}_{t+1}\right)\right] \\ &\leq r\left(\mathbf{s}_t, \mathbf{a}_t\right) + r_{aug}\left(\mathbf{s}_t, \mathbf{a}_t\right) + \gamma \mathbb{E}_{\mathbf{s}_{t+1}\sim p}\left[\mathbb{E}_{\mathbf{a}_{t+1}\sim\pi_{\mathrm{new}}}\left[Q^{\pi_{\mathrm{old}}}\left(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}\right) - \log \pi_{\mathrm{new}}\left(\mathbf{a}_{t+1} \mid \mathbf{s}_{t+1}\right)\right]\right] \\ &\vdots \\ &\leq Q^{\pi_{\mathrm{new}}}\left(\mathbf{s}_t, \mathbf{a}_t\right) \end{aligned} \tag{10}$$

Here, we have iteratively expanded $Q^{\pi_{\text{old}}}$ on the right-hand side by applying both the soft Bellman equation and the inequality from Equation 9.

Theorem B.3 (Converged SERA Soft Policy is Optimal) *Repetitive using Lemma 1 and Lemma 2 to any $\pi \in \Pi$ leads to convergence towards a policy π^*. And it can be proved that $Q^{\pi^*}(\mathbf{s}_t, \mathbf{a}_t) \geq Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t)$ for all policies $\pi \in \Pi$ and all state-action pairs $(\mathbf{s}_t, \mathbf{a}_t) \in \mathcal{S} \times \mathcal{A}$, provided that $|\mathcal{A}| < \infty$.*

proof.

Let π_i represent the policy at iteration i. According to Lemma 2, the sequence Q^{π_i} exhibits a monotonic increase. Given that rewards and entropy and thus Q^{π} are bounded from above for policies within the set Π, the sequence converges to a certain policy π^*. It is essential to demonstrate that π^* is indeed an optimal policy. Utilizing a similar iterative argument as employed in the proof of Lemma 2, we can establish that $Q^{\pi^*}(\mathbf{s}_t, \mathbf{a}_t) > Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t)$ holds for all $(\mathbf{s}_t, \mathbf{a}_t) \in \mathcal{S} \times \mathcal{A}$. In other words, the soft value associated with any other policy in Π is lower than that of the converged policy. Consequently, π^* is confirmed as the optimal policy within the set Π.

Theorem B.4 (Conservative Soft Q values with SERA) *By employing a double Q network, we ensure that in each iteration, the Q-value from the single Q network, denoted as $Q^{\pi_i}_{\textit{single Q}}(\mathbf{s}_t, \mathbf{a}_t)$, is greater than or equal to the Q-value obtained from the double Q network, represented as $Q^{\pi_i}_{\textit{double Q}}(\mathbf{s}_t, \mathbf{a}_t)$, for all $(\mathbf{s}_t, \mathbf{a}_t) \in \mathcal{S} \times \mathcal{A}$, where the action space is finite.*

proof. Let's begin by defining $\hat{Q}(\mathbf{s}_t, \mathbf{a}_t) = \min\left(Q_{\phi_1}(\mathbf{s}_t, \mathbf{a}_t), Q_{\phi_2}(\mathbf{s}_t, \mathbf{a}_t)\right)$. We then proceed to examine the difference between the augmented rewards in the context of SERA for the single Q and double Q networks:

$$
\begin{aligned}
& r_{aug}(\mathbf{s}_t, \mathbf{a}_t | \hat{Q}(\mathbf{s}_t, \mathbf{a}_t)) - r_{aug}(\mathbf{s}_t, \mathbf{a}_t | Q(\mathbf{s}_t, \mathbf{a}_t)) \\
=& \sum_{i=0}^{N} \log 2 \max(||s_i - s_i^{knn}||, ||\hat{Q}(\mathbf{s}_t, \mathbf{a}_t) - \hat{Q}^{knn}(\mathbf{s}_t, \mathbf{a}_t)||) - \\
& \sum_{i=0}^{N} \log 2 \max(||s_i - s_i^{knn}||, ||Q(\mathbf{s}_t, \mathbf{a}_t) - Q^{knn}(\mathbf{s}_t, \mathbf{a}_t)||) \\
=& \log \frac{\prod_{i=0}^{N} \max(||s_i - s_i^{knn}||, ||\hat{Q}(\mathbf{s}_t, \mathbf{a}_t) - \hat{Q}^{knn}(\mathbf{s}_t, \mathbf{a}_t)||)}{\prod_{i=0}^{N} \max(||s_i - s_i^{knn}||, ||Q(\mathbf{s}_t, \mathbf{a}_t) - Q^{knn}(\mathbf{s}_t, \mathbf{a}_t))||} \\
\approx& \log \frac{\prod_{i=0}^{N} \max(||s_i - s_i^{knn}||, \mathcal{H}(\hat{Q}))}{\prod_{i=0}^{N} \max(||s_i - s_i^{knn}||, \mathcal{H}(Q)||)} \\
\leq& \log \frac{\prod_{i=0}^{N} \max(||s_i - s_i^{knn}||, \mathcal{H}(Q))}{\prod_{i=0}^{N} \max(||s_i - s_i^{knn}||, \mathcal{H}(Q))} = 0
\end{aligned}
\tag{11}
$$

Consequently, we establish that $r_{aug}(\mathbf{s}_t, \mathbf{a}_t | \hat{Q}(\mathbf{s}_t, \mathbf{a}_t)) \leq r_{aug}(\mathbf{s}_t, \mathbf{a}_t | Q(\mathbf{s}_t, \mathbf{a}_t))$. Now we consider the modified soft Bellman equation

$$
\begin{aligned}
& Q^{\pi_i}_{\text{double Q}}(\mathbf{s}_t, \mathbf{a}_t) \\
=& r(\mathbf{s}_t, \mathbf{a}_t) + r_{aug}(\mathbf{s}_t, \mathbf{a}_t | \hat{Q}(\mathbf{s}_t, \mathbf{a}_t)) + \gamma \cdot \mathbb{E}_{\mathbf{s}_{t+1} \sim p}[\hat{V}(\mathbf{s}_{t+1})] \\
=& r(\mathbf{s}_t, \mathbf{a}_t) + r_{aug}(\mathbf{s}_t, \mathbf{a}_t | \hat{Q}(\mathbf{s}_t, \mathbf{a}_t)) + \gamma \cdot \mathbb{E}_{\mathbf{s}_{t+1} \sim p, \mathbf{a}_{t+1} \sim \pi}\left[\hat{Q}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) - \log \pi(\mathbf{a}_{t+1} \mid \mathbf{s}_{t+1})\right] \\
& \vdots \\
=& r(\mathbf{s}_t, \mathbf{a}_t) + r_{aug}(\mathbf{s}_t, \mathbf{a}_t | \hat{Q}(\mathbf{s}_t, \mathbf{a}_t)) + \gamma \cdot \mathbb{E}_{\mathbf{s}_{t+1} \sim p, \mathbf{a}_{t+1} \sim \pi}[r^{mod}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1} | \hat{Q}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}))] \cdots + \\
& \gamma^n \cdot \mathbb{E}_{\mathbf{s}_{t+n} \sim p, \mathbf{a}_{t+n} \sim \pi}[r^{mod}(\mathbf{s}_{t+n}, \mathbf{a}_{t+n} | \hat{Q}(\mathbf{s}_{t+n}, \mathbf{a}_{t+n}))] + \cdots + \text{entropy terms} \\
\leq& r(\mathbf{s}_t, \mathbf{a}_t) + r_{aug}(\mathbf{s}_t, \mathbf{a}_t | Q(\mathbf{s}_t, \mathbf{a}_t)) + \gamma \cdot \mathbb{E}_{\mathbf{s}_{t+1} \sim p, \mathbf{a}_{t+1} \sim \pi}[r^{mod}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1} | Q(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}))] \cdots + \\
& \gamma^n \cdot \mathbb{E}_{\mathbf{s}_{t+n} \sim p, \mathbf{a}_{t+n} \sim \pi}[r^{mod}(\mathbf{s}_{t+n}, \mathbf{a}_{t+n} | Q(\mathbf{s}_{t+n}, \mathbf{a}_{t+n}))] + \cdots + \text{entropy terms} \\
=& Q^{\pi_i}_{\text{single Q}}(\mathbf{s}_t, \mathbf{a}_t)
\end{aligned}
\tag{12}
$$

where we have repeatedly expanded $\hat{Q}$ in terms of SERA rewards to obtain the final inequality $Q^{\pi_i}_{\text{singleQ}} \geq Q^{\pi_i}_{\text{double Q}}$.

C EXPERIMENTAL SETUP

In this section, we introduce the benchmarks and dataset we utilized, specifically, we mainly utilize gym-mujoco and antmaze to test our algorithm.

C.1 GYM MUJOCO

Our benchmars from gym-mujoco domain mainly includes `halfcheetah`, `ant`, `hopper` and `walker2d`, and concrete information of these benchmarks can be referred to table 1. In paticular, the action and observation space of these locomotion benchmarks are continuous and any decision making will receive an immediate reward.

Environment	Task Name	Samples	Observation Dim	Action Dim
`halfcheetah`	medium	10^6	6	17
`walker2d`	medium	10^6	6	17
`hopper`	medium	10^6	3	11
`ant`	medium	10^6	8	111
`halfcheetah`	medium-replay	2.02×10^5	6	17
`walker2d`	medium-replay	3.02×10^5	6	17
`hopper`	medium-replay	4.02×10^5	3	11
`ant`	medium-replay	3.02×10^5	8	111

Table 1: Introduction of D4RL tasks (Gym-Mujoco).

C.2 ANTMAZE

Our benchmars from antmaze mainly includes `antmaze-large-diverse`, `antmaze-medium-diverse`, `antmaze-large-play` and `antmaze-medium-play`, concrete information of our benchmarks can be referred to table 2.

Environment	Task Name	Samples	Observation Dim	Action Dim
`antmaze`	large-diverse	10^6	29	8
`antmaze`	large-play	10^6	29	8
`antmaze`	medium-diverse	10^6	29	8
`antmaze`	medium-play	10^6	29	8

Table 2: Introduction of D4RL tasks (Antmaze).

D IMPLANTATION DETAILS

D.1 OFFLINE-TO-ONLINE IMPLANTATION

The workflow of our method is similar to the most of offline-to-online algorithms that we firstly pre-train on offline datasets, followed by online fine-tuning (Interacting with online environment to collect online dataset and followed by fine-tuning on offline and online datasets).

D.2 EVALUATION DETAILS

Our evaluation method can be refered to Fu et al. (2021). That is for each evaluation, we freeze the parameter of trained model, and then conducting evaluation 10$\sim$50 times and then computing the normalized score via $\frac{\text{score}_{\text{evaluation}}-\text{score}_{\text{expert}}}{\text{score}_{\text{expert}}-\text{score}_{\text{random}}}$, and then averaging these normalized evaluation scores.

D.3 SERA IMPLANTATION

In SERA framework, we modify our reward as :

$$r^{\text{mod}}(\mathbf{s},\mathbf{a}) = \lambda \cdot \underbrace{\text{Tanh}(\mathcal{H}(\mathbf{s}|\min(Q_{\phi_1}(\mathbf{s},\mathbf{a}),Q_{\phi_2}(\mathbf{s},\mathbf{a}))))}_{r^{\text{aug}}} + r(\mathbf{s},\mathbf{a}), \quad (\mathbf{s},\mathbf{a}) \sim \mathcal{D}_{\text{online}} \tag{13}$$

To calculate the intrinsic reward r^{aug} for the online replay buffer D_{online}, we use the KSG estimator, as defined in Equation 14, to estimate the conditional state density of the empirical dataset D_{online}

$$r^{\text{aug}}(\mathbf{s},\mathbf{a}) = \frac{1}{d_s}\phi(n_v(i)+1)+\log 2\cdot\max(||\mathbf{s}_i-\mathbf{s}_i^{knn}||,||\hat{Q}(\mathbf{s},\mathbf{a})-\hat{Q}(\mathbf{s},\mathbf{a})^{knn}||),(\mathbf{s},\mathbf{a})\sim\mathcal{D}_{\text{online}}. \tag{14}$$

Given that the majority of our selected baselines are implemented using the double Q($\{Q_{\phi_1}, Q_{\phi_2}\}$), the offline pre-trained double Q can be readily utilized for the computation of intrinsic rewards, and we found that the performance of SERA is sutured when λ is set to 1. We also provide a (Variance Auto Encoder) VAE implantation (Equation 15) of SERA, this realization is computing efficiency, but require extraly training a VAE model, due to Equation **??** won't require training thus we mainly test Equation **??**.

$$r^{\text{aug}}(\mathbf{s},\mathbf{a}) = -\log p_{\hat{\phi}}(s|\hat{Q}(\mathbf{s},\mathbf{a})) = -\log \mathbb{E}_{z\sim q_\phi(z|\mathbf{s},\hat{Q}(\mathbf{s},\mathbf{a}))}[\frac{p_{\hat{\phi}}(\mathbf{s}|\hat{Q}(\mathbf{s},\mathbf{a}))}{q_\phi(z|\mathbf{s},\hat{Q}(\mathbf{s},\mathbf{a}))}],(\mathbf{s},\mathbf{a})\sim\mathcal{D}_{\text{online}}. \tag{15}$$

We will test and compare the performance difference and computing efficiency between Equation 15 and Equation 14 in the future.

D.4 CODEBASE

Our implementation is based on Cal-QL:`https://github.com/nakamotoo/Cal-QL`, VCSE:`https://sites.google.com/view/rl-vcse`. Additionally, we have included our source code in the supplementary material for reference. Readers can refer to our pseudocode (see Algorithm 1) for a comprehensive understanding of the implementation details. $\hat{Q}$ see [1].

Algorithm 1 Training SERA

Require: Pre-collected data $\mathcal{D}_{\text{offline}}$.
1: Initialize π_θ, and Q_{ϕ_1}, Q_{ϕ_2}.
// Offline Pre-training Stage.
2: **for** $k = 1, \cdots, K$ **do**
3: Learn Q_ϕ on $\mathcal{D}_{\text{offline}}$ by Equation 17 or 16 //We compute target Q value via Q_{target}, learning Q_{target} by Empirical Momentum Average (EMA),*i.e.*, $Q_{\text{target}} = (1-\alpha)Q_\phi + \alpha Q_{\text{target}}$.
4: Learn π_θ on $\mathcal{D}_{\text{offline}}$ with Equation 18.
5: **end for**
// Online Fine-tuning Stage.
6: **for** $k = 1, \cdots, K$ **do**
7: Interacting π_θ to obtain $\mathcal{D}_{\text{online}}$.
8: Augmenting Reward in $\mathcal{D}_{\text{online}}$ by Equation 14.
9: Sample a batch offline data $\mathcal{D}_{\text{offline}}$, and build training batch,*i.e.*, $\mathcal{D}_{\text{mix}} = \mathcal{D}_{\text{offline}} \cup \mathcal{D}_{\text{online}}$ //mixture of offline and online is not necessary required, it depends on the quality of offline dataset.
10: Learn π_θ, Q_{ϕ_1}, and Q_{ϕ_2} on $\mathcal{D}_{\text{mix}}$ with the same objective in offline stage.
11: **end for**

Training Objective. Since SERA satisfy the guarantee of soft Q optimization, we primarily validate our method on CQL and Cal-QL, regarding the training objective of Cal-QL and CQL, we update Cal-QL's Q Network using Equation 16, and we update CQL's Q Network using Equation 17:

$$L(Q) = \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}}[(Q^\pi(\mathbf{s},\mathbf{a})-\mathcal{B}^\pi_\mathcal{M}Q(\mathbf{s},\mathbf{a}))^2]+\mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\pi}[\max(Q^\pi(\mathbf{s},\mathbf{a}),V^\mu(\mathbf{s}))]-\mathrm{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}[Q^\pi(\mathbf{s},\mathbf{a})]. \tag{16}$$

$$L(Q) = \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}}[(Q^\pi(\mathbf{s},\mathbf{a}) - \mathcal{B}^\pi_\mathcal{M}Q(\mathbf{s},\mathbf{a}))^2] + \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}}[-Q^\pi(\mathbf{s},\mathbf{a}) + Q^\pi(\mathbf{s}',\pi(\mathbf{s}'))], \tag{17}$$

where $\mathcal{D}$ is the batch training data. Meanwhile, updating their policies by Equation 18:

$$\mathcal{J}(\pi_\theta) = \mathrm{E}_{\mathbf{s}\sim\mathcal{D}}[-Q^\pi(\mathbf{s},\pi_\theta(\mathbf{s})) + \alpha\log(\pi_\theta(\mathbf{s}))]. \tag{18}$$

[1] where ϕ_1 and ϕ_2 are the params of double Q Networks and $\hat{Q}(\mathbf{s},\mathbf{a}) = \min(Q_{\phi_1}(\mathbf{s},\mathbf{a}), Q_{\phi_2}(\mathbf{s},\mathbf{a}))$, and x_i^{knn} is the $n_x(i)$-th nearest neighbor of x_i.

D.5 COMPUTING RESOURCES

Our experiments were run on a computer cluster with 4×32GB RAM, AMD EPYC 7742 64-Core CPU, and NVIDIA-A100 GPU, Linux. Most of our code base (The implantation of **Cal-QL**, **CQL**, **TD3+BC**, **SAC**) are based on JAX [2], part of our implantation (**IQL**, **AWAC**) are based on Pytorch[3] (We use different deep learning frameworks mainly to preliminary validate that our algorithm can work in various of deep learning frameworks).

D.6 OUR HYPER-PARAMETER

Hyper-parameter of SERA. The K-nearest neighbors (knn) for SERA are configured as follows: [0, 10, 15, 25, 50, 85, 100, 110], and the parameter λ in Equation 13 is set to 1.

Hyper-parameter of Baselines In the context of these algorithms, we conducted tests related to AWAC and IQL using the repository available at `https://github.com/tinkoff-ai/CORL`, while tests related to Cal-QL and CQL were performed using the repository accessible at `https://github.com/nakamotoo/Cal-QL`. The following five tables present fundamental but critical hyperparameter settings for five baseline algorithms.

Table 3: Hyper-parameters of AWAC.

Hyperparameter	Value
0ffline pre-train iterations	$1e^6$
0nline fine-tuning iterations	$1e^6$
Buffer size	20000000
Batch size	256
learning rate	$3e^{-4}$
γ	0.99
awac τ	5e-3
awac λ	1.0
Actor Architecture	4× Layers MLP (hidden dim 256)
Critic Architecture	4× Layers MLP (hidden dim 256)

Table 4: Hyper-parameters of IQL.

Hyperparameter	Value
0ffline pre-train iterations	$1e^6$
0nline fine-tuning iterations	$1e^6$
Batch size	256
learning rate of π	$3e^{-4}$
learning rate of V	$3e^{-4}$
learning rate of Q	$3e^{-4}$
γ	0.99
IQL τ	0.7 # *Coefficient for asymmetric loss*
β (Inverse Temperature)	3.0# *small beta → BC, big beta → maximizing Q*
Actor Architecture	4× Layers MLP (hidden dim 256)
Critic Architecture	4× Layers MLP (hidden dim 256)

[2] `https://github.com/google/jax.git`
[3] `https://pytorch.org/`

Table 5: Hyper-parameters of TD3+BC.

Hyperparameter	Value
0ffline pre-train iterations	$1e^6$
0nline fine-tuning iterations	$1e^6$
learning rate of π	$1e^{-4}$
learning rate of Q	$3e^{-4}$
γ	0.99
Batch size	256
TD3 alpha	2.5
Actor Architecture	4$\times$ Layers MLP (hidden dim 256)
Critic Architecture	4$\times$ Layers MLP (hidden dim 256)

Table 6: Hyper-parameters of Cal-QL. We only provide the basic setting, for more detail setting, please directly refer to `https://nakamotoo.github.io/projects/Cal-QL`

Hyperparameter	Value
0ffline pre-train iterations	$1e^6$
0nline fine-tuning iterations	$1e^6$
learning rate of π	$1e^{-4}$
learning rate of Q	$3e^{-4}$
γ	0.99
Batch size	256
Actor Architecture	4$\times$ Layers MLP (hidden dim 256)
Critic Architecture	4$\times$ Layers MLP (hidden dim 256)

Table 7: Hyper-parameters of CQL. CQL uses Cal-QL's code-base, and we only need to remove Cal-QL's calibration loss when deploying CQL.

Hyperparameter	Value
0ffline pre-train iterations	$1e^6$
0nline fine-tuning iterations	$1e^6$
learning rate of π	$1e^{-4}$
learning rate of Q	$3e^{-4}$
γ	0.99
Batch size	256
Actor Architecture	4$\times$ Layers MLP (hidden dim 256)
Critic Architecture	4$\times$ Layers MLP (hidden dim 256)

E APPENDED EXPERIMENTAL RESULTS

In Table 8, we have provided completed offline-to-online results, including the ant-maze domain and the medium and medium-replay scenarios in the gym-mujoco environment, which is matched Figure **??**. From Table 8, it can be observed that CQL paired with SERA exhibits the best average performance on the selected tasks. In Table 9, we compare a series of different efficient offline-to-online methods, including APL, PEX, and BR. Specifically, we tested these methods on the ant-maze domain and the `medium` and `medium-replay` tasks in the gym-mujoco environment. We found that SERA shows the best overall performance, indicating that SERA, when paired with CQL, can achieve superior results.

Task	IQL	AWAC	TD3+BC	CQL	**CQL+SERA**	Cal-QL	**Cal-QL+SERA**
`antmaze-large-diverse`	59	00	00	89.2	89.8	86.3	**94.5**
`antmaze-large-play`	51	00	00	91.7	92.6	83.3	**95.0**
`antmaze-medium-diverse`	92	00	00	89.6	98.9	96.8	**99.6**
`antmaze-medium-play`	94	00	00	97.7	**99.4**	95.8	98.9
`halfcheetah-medium`	57	67	49	69.9	**87.9**	45.6	46.9
`walker2d-meidum`	93	91	82	123.1	**130.0**	80.3	90.0
`hopper-medium`	67	101	55	56.4	**62.4**	55.8	61.7
`ant-medium`	113	121	43	123.8	**136.9**	96.4	104.2
`halfcheetah-medium-replay`	54	44	49	39.5	**53.73**	25.7	26.7
`walker2d-medium-replay`	90	73	90	87.6	**107.65**	7.4	29.7
`hopper-medium-replay`	**91**	56	88	4	15.0	3.5	1.8
`ant-medium-replay`	123	**127**	127	30	116.6	55.1	68.0
Average Fine-tuned	82.2	56.7	48.6	75.1	90.9	61.3	68.1

Table 8: Normalized score after online fine-tuning. We report the online fine-tuned normalized return. SERA obviously improves the performance of CQL and Cal-QL. In particular, CQL-SERA (mean score of **90.9**) is the best out of the 12 selected baselines. Notably, part of Antmaze's baseline results are *quoted* from existing studies. Among them, AWAC's results are *quoted* from Kostrikov et al. (2021) and CQL's results are *quoted* from Nakamoto et al. (2023).

Task	CQL+APL	CQL+PEX	CQL+BR	CQL+SUNG	CQL+SERA
`antmaze-large-diverse`	0	0	0.1	44.1	89.8
`antmaze-large-play`	0	0	0	52.7	92.6
`antmaze-medium-diverse`	36.8	0.3	13.6	85.6	98.9
`antmaze-medium-play`	22.8	0.3	22.2	86.3	99.4
`halfcheetah-medium`	44.7	43.5	56.7	79.7	87.9
`walker2d-meidum`	75.3	34.0	81.7	86.0	130.0
`hopper-medium`	102.7	46.3	97.7	104.1	62.4
`halfcheetah-medium-replay`	78.6	45.5	64.9	75.6	53.7
`walker2d-medium-replay`	103.2	40.1	88.5	108.2	107.7
`hopper-medium-replay`	97.4	66.5	78.8	101.9	15.2
Average Fine-tuned	56.2	27.6	50.4	82.4	83.8

Table 9: Comparison of various efficient offline-to-online methods.

F EXTENDED EXPERIMENTS

F.1 TREND OF STATE ENTROPY CHANGING

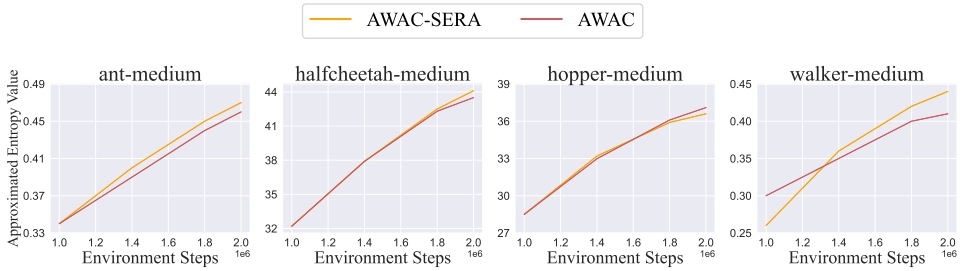
State Entropy as Intrinsic Reward. If the state density $\rho(\mathbf{s})$ is unknown, we can instead using non-parametric entropy estimator to approximate the state entropy (Seo et al., 2021). Specifically, given N i.i.d. samples $\{\mathbf{s}_i\}$, the k-nearest neighbors (knn) entropy estimator can be defined as[4]:

$$\hat{H}_N^k(S) = \frac{1}{N}\sum_{i=1}^{N} \log \frac{N \cdot ||\mathbf{s}_i - s_i^{knn}||_2^{d_\mathbf{s}} \cdot \hat{n}_{\hat{\pi}}^{\frac{d_\mathbf{s}}{2}}}{k \cdot \Gamma(\frac{d_\mathbf{s}}{2}+1)} \propto \frac{1}{N}\sum_{i=1}^{N} \log ||\mathbf{s}_i - \mathbf{s}_i^{knn}||. \tag{19}$$

[4] d_s is the dimension of state and Γ is the gamma function, $\hat{n}_{\hat{\pi}} \propto 3.14$.

Visualization of State Entropy Changing. In this experiment, for each training step, we select the buffer and randomly sample 5000 instances to approximate the entropy using Equation 2. and then plot the trend of approximated state entropy. For the majority of the tasks, the state entropy of *AWAC-SERA* was either progressively greater than or consistently exceeded that of *AWAC-base*. This indicates that SERA effectively enhances the agent's exploratory tendencies, enabling them cover much more observation region.

Figure 1: The Changing of Approximated Entropy along with increasing training steps. We found that the approximated state entropy in the buffer collected by AWAC using SERA was greater in the later stages of online finetuning.

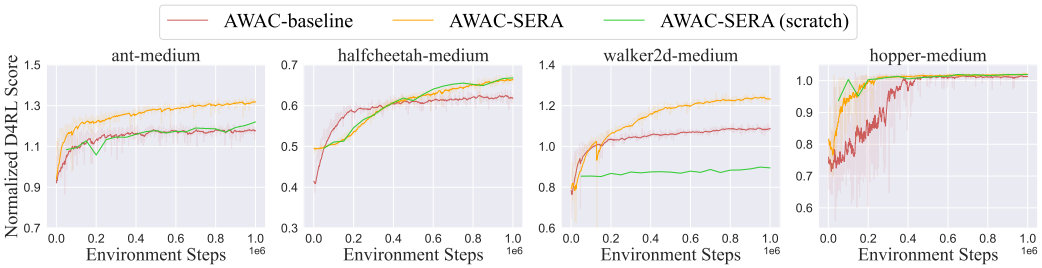


F.2 PRETRAINED Q VS. RANDOM Q

Pre-trained Q condition versus un-pre trained Q condition. To validate the statement in our main paper that intrinsic reward computation is influenced by the initialization of Q, we conducted experiments comparing the effects of pre-trained initialized Q and from-scratch[5] trained Q during intrinsic reward calculation. Our findings indicate that intrinsic rewards based on offline-initialized Q generally outperform those derived from a from-scratch trained Q across most tasks.

Figure 2: Offline Pre-trained Q condition vs. Randomly initialized Q condition. In the majority of our selected Gym-Mujoco tasks, the use of offline-initialized intrinsic reward conditions yielded better performance and higher sample efficiency. To provide clarity, *AWAC-base* means AWAC algorithm without any modification, *AWAC-SERA* signifies AWAC with SERA augmentation, and *AWAC-SERA (scratch)* denotes AWAC with SERA where the computation of reward conditions satisfying note 5

.



F.3 Q CONDITION VS. V CONDITION

Differing from Kim et al. (2023), SERA conditions its intrinsic reward on $\min(Q_{\phi_1}(\mathbf{s},\mathbf{a}), Q_{\phi_2}(\mathbf{s},\mathbf{a}))$ rather than $V(\mathbf{s})$. In comparison to VCSE, SERA's advantage lies in its consideration of transitions. For example, assuming that there exist two transitions $T_1 = (\mathbf{s}, \mathbf{a}_1, \mathbf{s}_1)$ and $T_2 = (\mathbf{s}, \mathbf{a}_2, \mathbf{s}_2)$, since T_1 and T_2 have the same current observation, they will yield the same value conditioned intrinsic reward $-\log(\mathbf{s}|V(\mathbf{s}))$. This can introduce bias in the value learning process especially when current observation corresponds to a substantial number of valuable decisions and a limited number of low-value decisions. This is because low-value decisions can still receive relatively high intrinsic rewards based on the higher value expectations

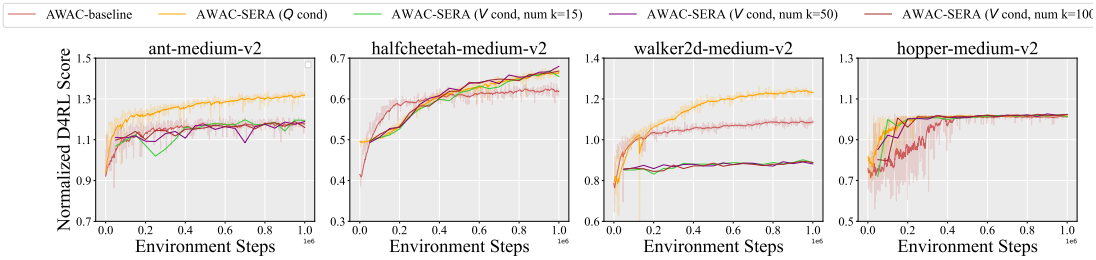
[5]We use from-scratch Q to compute intrinsic reward, while continuing to utilize the offline-initialized Q for conducting online fine-tuning.

$V(\mathbf{s})$ for the current state, subsequently influencing the agent's decision-making. However, if we condition intrinsic reward on $Q(\mathbf{s}, \mathbf{a})$, it can take into account the decision-making simultaneously.

To further validate our claims, we chose AWAC as baseline and used both Q-network and V-network to compute the intrinsic reward's condition. We conducted tests on `halfcheetah-medium`, `walekr2d-medium`, `hopper-medium` and `ant-medium`. As shown in Figure 3, using the Q-network to compute condition has better performance compared to using V-network to compute condition.

Figure 3: Q condition vs. V condition. In this experiment, we selected AWAC as the base algorithm and compared using V network and Q network to calculate the intrinsic reward's condition. The experimental results indicate that using the Q-network to compute the condition leads to overall better performance for AWAC.

.



G Extended Related Work

In this section, we systematically introduce recent developments in offline-to-online learning and summarize the corresponding methods,

The first perspective involves adopting a conservative policy optimization during online fine-tuning, typically achieved through the incorporation of policy constraints. Specifically, there are three main approaches within this category. The first approach constrains the predictions of the fine-tuning policy within the scope of offline support during online fine-tuning (Liu et al., 2023). While this method contributes to achieving stable online fine-tuning performance, it tends to lead to overly conservative policy learning, and the accuracy of the estimation of offline support also influences the effectiveness of online fine-tuning. The second approach utilizes an offline dataset to constrain policy learning (Nair et al., 2021; Kostrikov et al., 2021; Xiao et al., 2023; Mark et al., 2023). However, the effectiveness of fine-tuning cannot be guaranteed if the dataset quality is poor. This method is sensitive to the quality of the dataset. The third approach employs pre-trained policies to constrain online fine-tuning, but this paradigm is influenced by the quality of the pre-trained policy (Zhang et al., 2023; Yu & Zhang, 2023).

The second perspective involves adopting a conservative approach during offline training, specifically using pessimistic constraints to learn Q to avoid OOD (Out-of-Distribution) issues. Research in this category primarily includes: Learning a conservative Q during offline pretraining and employing an appropriate experience replay method during online learning or using Q ensemble during offline pre-training to avoid OOD issues (Lee et al., 2021; Lyu et al., 2022; Hong et al., 2023). However, as this approach introduces conservative constraints during critic updates, the value estimates between offline and online are not aligned, leading to a decrease in performance during early online fine-tuning. Therefore, Cal-QL introduces a calibrated conservative term to ensure standard online fine-tuning (Nakamoto et al., 2023).

Addtionally, there are also some other methods, such that ODT (Zheng et al., 2022) combined sequence modeling with Goal conditioned RL to conduct offline-to-online RL.

References

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2021.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

Joey Hong, Aviral Kumar, and Sergey Levine. Confidence-conditioned value functions for offline reinforcement learning, 2023.

Dongyoung Kim, Jinwoo Shin, Pieter Abbeel, and Younggyo Seo. Accelerating reinforcement learning with value-conditional state entropy exploration, 2023.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning, 2021.

Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble, 2021.

Jinxin Liu, Ziqi Zhang, Zhenyu Wei, Zifeng Zhuang, Yachen Kang, Sibo Gai, and Donglin Wang. Beyond OOD state actions: Supported cross-domain offline reinforcement learning. *CoRR*, abs/2306.12755, 2023. doi: 10.48550/arXiv.2306.12755. URL https://doi.org/10.48550/arXiv.2306.12755.

Jiafei Lyu, Xiaoteng Ma, Xiu Li, and Zongqing Lu. Mildly conservative q-learning for offline reinforcement learning, 2022.

Max Sobol Mark, Ali Ghadirzadeh, Xi Chen, and Chelsea Finn. Fine-tuning offline policies with optimistic action selection, 2023. URL https://openreview.net/forum?id=2x8EKbGU51k.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets, 2021.

Mitsuhiko Nakamoto, Yuexiang Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning, 2023.

Younggyo Seo, Lili Chen, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. State entropy maximization with random encoders for efficient exploration, 2021.

R.S. Sutton and A.G. Barto. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 9(5):1054–1054, 1998. doi: 10.1109/TNN.1998.712192.

Chenjun Xiao, Han Wang, Yangchen Pan, Adam White, and Martha White. The in-sample softmax for offline reinforcement learning, 2023.

Zishun Yu and Xinhua Zhang. Actor-critic alignment for offline-to-online reinforcement learning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 40452–40474. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/yu23k.html.

Haichao Zhang, We Xu, and Haonan Yu. Policy expansion for bridging offline-to-online reinforcement learning, 2023.

Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer, 2022.