COMPOSITIONAL CONSERVATISM: A TRANSDUCTIVE APPROACH IN OFFLINE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Offline Reinforcement learning (RL) is a compelling framework for learning optimal policies without additional environmental interaction. Nevertheless, offline RL inevitably faces the problem of distributional shifts, where the states and actions encountered during policy execution are not in the training dataset. A common solution involves incorporating conservatism into either the policy or value function, which serves as a safeguard against uncertainties and unknowns. In this paper, we also focus on achieving the same objectives of conservatism but from a different perspective. We propose COmpositional COnservatism with Anchor-seeking (*COCOA*) for offline RL, an approach that pursues conservatism in a compositional manner on top of the transductive reparameterization (Netanyahu et al., 2023). In this reparameterization, the input variable (the state in our case) is viewed as the combination of an anchor and its difference from the original input. Independently of and agnostically to the prevalent *behavioral* conservatism in offline RL, COCOA learns to seek both in-distribution anchors and differences with the learned dynamics model, encouraging conservatism in the *compositional input space* for the function approximators of the Q-function or the policy. Our experimental results show that our method generally improves the performance of four state-of-the-art offline RL algorithms on the D4RL benchmark.

1 INTRODUCTION

Reinforcement learning (RL) (Sutton & Barto, 2018) has achieved notable successes across various domains, from guiding robotic movements (Dasari et al., 2020) and optimizing game strategies (Mnih et al., 2015) to recently promising training of language models (Rajpurkar et al., 2016). Despite these achievements, the challenges posed by real-time interaction in complex and sensitive environments have prompted the development of offline RL as a viable direction. Offline RL (Wiering & Van Otterlo, 2012; Levine et al., 2020), or batch RL (Lange et al., 2012), involves learning policies solely from pre-existing data, without any direct interaction with the environment. Offline RL is becoming increasingly popular in real-world applications such as autonomous driving (Yu et al., 2020a) or healthcare (Gottesman et al., 2019) where prior data are abundant.

By its nature of learning from prior datasets, offline RL is often susceptible to distributional shifts. This issue arises when the distribution of states and actions encountered during policy execution is different from the training dataset, a situation particularly challenging in machine learning (Levine et al., 2020). Numerous existing offline RL algorithms tackle this challenge by reducing distributional shifts through conservative approaches, including constraining the policy or estimating uncertainty to measure distributional deviations (Kim & Oh, 2023; Ran et al., 2023; Kostrikov et al., 2022; Kumar et al., 2020; Wu et al., 2019; Kumar et al., 2019; Fujimoto et al., 2019; Sun et al., 2023; Rigter et al., 2022; Wang et al., 2021; Yu et al., 2021; 2020b; Kidambi et al., 2020). These strategies aim to keep the agent within known distributions, mitigating risks of unexpected behaviors. In this paper, we also pursue the same goals of conservatism, focusing on aligning the test data distribution with the seen distribution, but from a different perspective.

We begin by recognizing that the state distributional shift problem is closely related to addressing how to deal with the out-of-support input points of the function approximators. We explore the possibility of transforming the out-of-support learning problem into an out-of-combination problem by injecting inductive biases into function approximators of the policy or the Q-value function. Such

a transformation has been previously proposed by Netanyahu et al. (2023), where a transductive approach named *bilinear transduction* makes predictions through a bilinear architecture after reparameterizing the target function. This reparameterization decomposes the input variable into two components, namely an *anchor* and a *delta*, where the anchor is another variable in the input space and the delta is the difference between the input variable and the anchor. If the reparameterized training and test data distribution satisfy certain assumptions, and if the target function has certain properties, the bilinear transduction can address the out-of-combination problem, which potentially resolves the out-of-support problem with the original target function.

In this work, we propose COmpositional COnservatism with Anchor-seeking (*COCOA*) for offline RL, a framework that adopts a compositional approach to conservatism, building upon the transductive reparameterization (Netanyahu et al., 2023). Our approach, following (Netanyahu et al., 2023), transforms the distributional shift problem into an out-of-combination problem. This shifts the key factors for generalizability from the data to the decomposed components and the interrelations between them, demanding that the anchor and delta should be selected close to the training dataset distribution. Otherwise, if the anchor is selected arbitrarily, it can lead to unintended and potentially detrimental effects in generalization. [Yeda: Review: include this?]

We suggest a new anchor-seeking approach with an additional policy, named *anchor-seeking policy*, which constrains the agent to find anchors within the seen area of the state space. With its anchor-seeking policy, COCOA encourages anchors to be close to the offline dataset while also confining the deltas within a narrow range by identifying anchors among neighboring states. This approach reduces the input space and guides it toward space that was predominantly explored during the training phase. In summary, by learning a policy to seek in-distribution anchors and differences from the learned dynamics, we can encourage conservatism in the compositional input space of the function approximator for the Q-function and policy. This approach is independent of and agnostic to the prevalent behavioral conservatism in offline RL.

We empirically find that our method generally improves the performance of four representative offline RL methods, including CQL (Kumar et al., 2020), IQL (Kostrikov et al., 2022), MOPO (Yu et al., 2020b), and MOBILE (Sun et al., 2023) on the D4RL benchmark (Fu et al., 2020). We also show that learning anchor-seeking positively impacts the performance of our method through an ablation study. Our main contributions can be summarized as follows:

- We pursue conservatism in the *compositional input space* for the function approximators of the Q-function and policy, independently and agnostically to the prevalent *behavioral* conservatism in offline RL.
- We introduce COmpositional COnservatism with Anchor-seeking (*COCOA*) that finds in-distribution anchors and deltas with the learned dynamics model, which is crucial for compositional generalization.
- We empirically show that the performance of four state-of-the-art offline RL algorithms on the D4RL benchmark is generally improved when equipped with COCOA. Additionally, our ablation study shows the efficacy of the anchor-seeking policy compared to a heuristic anchor selection.

2 PRELIMINARIES

2.1 OFFLINE RL

We assume an MDP problem $(\mathcal{S}, \mathcal{A}, T, R)$ with a continuous state space $\mathcal{S}$, a continuous action space $\mathcal{A}$, a transition function $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, and a reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The goal is to find a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the expected return $J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R\left(s_t, a_t\right)\right]$, where $\gamma \in [0, 1)$ is a discount factor.

In offline RL, also known as batch RL, we are given a dataset $\mathcal{D}_{\text{env}} = \{(s_i, a_i, s_{i+1}, r_i)\}_{i=1}^{N}$ generated with a behavior policy. The goal in offline RL is to find a policy π that maximizes the expected return $J(\pi)$ using only the fixed dataset $\mathcal{D}_{\text{env}}$. Like most model-baed offline RL algorithms, we learn a dynamics model $\widehat{T}(s_{i+1}|s_i, a_i)$ which predicts the next state s_{i+1} given the current state s_i and

action a_i. In addition to the forward dynamics model, we also learned a reverse dynamics model $\widehat{T}(s_i|s_{i+1}, a_i)$ which predicts the current state s_i given the next state s_{i+1} and action a_i.

2.2 BILINEAR TRANSDUCTION

In this section, we follow the formulation of Netanyahu et al. (2023) about generalization problems. Without assumptions on the train and test distribution, the generalization performance of a function approximator is limited. This problem occurs especially when the test distribution is not contained in the train distribution, which is also known as an out-of-support (OOS) learning problem. As a special case of OOS, an out-of-combination (OOC) problem occurs when the input space is decomposed into two components, and the marginal of the train distribution of each component includes that of the test distribution while the joint train distribution does not necessarily contain the joint test distribution. Under certain assumptions, Netanyahu et al. (2023) propose a transductive reparameterization method called *bilinear transduction* to convert an OOS problem into an OOC problem and address it.

Bilinear transduction. Bilinear transduction (Netanyahu et al., 2023) is a method for solving extrapolation under certain assumptions. It first reparameterizes the target function $f(x)$ as follows:

$$f(x) := \bar{f}(x - \tilde{x}, \tilde{x}) \tag{1}$$

Here, $\tilde{x}$ is termed as an *anchor* which is selected from the training dataset and the difference $(x - \tilde{x})$ between the input variable x and the anchor $\tilde{x}$ is termed as a *delta*. The reparameterized target function $\bar{f}$ is approximated as a bilinear function of the embeddings $\boldsymbol{\varphi_1}$ and $\boldsymbol{\varphi_2}$:

$$\bar{f}_{\boldsymbol{\theta}}(x) = \boldsymbol{\varphi_1}(x - \tilde{x}) \cdot \boldsymbol{\varphi_2}(\tilde{x}) \tag{2}$$

With this bilinear architecture, intuitively, it facilitates the low-rank property of the embeddings $\boldsymbol{\varphi_1}$ and $\boldsymbol{\varphi_2}$ which enables the function approximator to generalize to OOC points.

Sufficient conditions for bilinear transduction. Netanyahu et al. (2023) introduce sufficient conditions for bilinear transduction to be applicable. Those assumptions are about both the dataset and the target function f. The first assumption is about a *combinatorial coverage* of the dataset. The test dataset has to have a bounded combinatorial density ratio with respect to the training dataset. It implies that the support of the joint distribution of the training distributions of the components should include the support of the joint distribution of the test distributions of the components. Second, the target function f should be *bilinearly transducible* , i.e., there exists a deterministic function $\bar{f}$ such that $f(x) = \bar{f}(x - \tilde{x}, \tilde{x})$ for all $x, \tilde{x} \in \mathcal{X}$. Lastly, the training distribution of anchors should not degenerate (Shah et al., 2020). Under these three conditions, it is possible to generalize the target function to OOC points with a theoretically guaranteed risk bound.

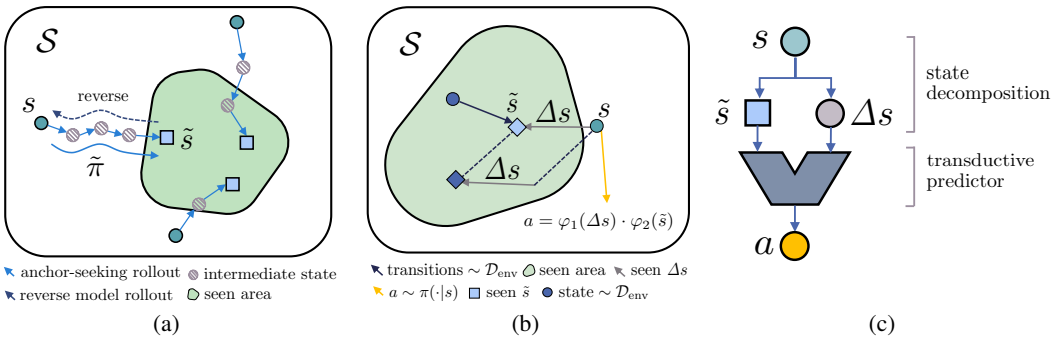
Connection to compositional generalization. In light of the literature on *compositional generalization* (Wiedemer et al., 2023), we interpret bilinear transduction as a special case of compositional generalization, where the $\boldsymbol{\varphi_1}, \boldsymbol{\varphi_2}$ models serves as *component functions*, extracting low-rank features of the input, and the inner product serves as a *composition function*.

3 COMPOSITIONAL CONSERVATISM WITH ANCHOR-SEEKING (COCOA)

3.1 OFFLINE RL WITH BILINEAR TRANSDUCTION

The base algorithms for common methods in offline RL, such as Deep Q-Networks (DQN) (Mnih et al., 2015) and Actor-Critic methods (Mnih et al., 2016; Haarnoja et al., 2018), frequently use deep neural networks as function approximators. Therefore, we employ bilinear transduction (Section 2.2) to the function approximators of policy and Q-function. In both train and test phases, we decompose the current state s into an anchor $\tilde{s}$ and a delta $\Delta s = s - \tilde{s}$, where $\tilde{s} \sim \mathcal{D}_{\text{env}}$. Then, the policy and the Q-function will be described as follows:

$$\begin{aligned} \bar{\pi}_{\boldsymbol{\theta}}(s) &= \boldsymbol{\varphi_{\theta,1}}(\Delta s) \cdot \boldsymbol{\varphi_{\theta,2}}(\tilde{s}) \\ \bar{Q}_{\boldsymbol{\phi}}(s, a) &= \boldsymbol{\varphi_{\phi,1}}(\Delta s, a) \cdot \boldsymbol{\varphi_{\phi,2}}(\tilde{s}, a) \end{aligned} \tag{3}$$



(a) (b) (c)

Figure 1: **(a)** An illustration of anchor-seeking rollouts that find anchors close to the seen area of the state space $\mathcal{S}$. Given the current state s, the anchor-seeking policy $\tilde{\pi}$ gives actions to reach the anchor $\tilde{s}$. Its behavior is derived by utilizing reverse model rollouts, which diverge from the offline dataset. **(b)** An illustration of the current state s and an anchor $\tilde{s}$. Ideally, this anchor $\tilde{s}$ has been observed during the training phase when it served as an anchor for another state. Similarly, the difference, delta, had also been encountered previously in the ideal case, but in combination with a different anchor. **(c)** The architecture of our policy $\pi(a|s)$ that aims to generalize to an unfamiliar state, by decomposing the state s into familiar components (seen anchor $\tilde{s}$ and seen delta Δs) and applying a transductive predictor. The architecture of the Q-function is similar to that of the policy.

The architecture of the policy and Q-function is illustrated in Figure 1c. The policy $\pi(a|s)$ is trained to maximize the expected return $J(\pi)$, and the Q-function $Q(s, a)$ is trained to minimize its loss function $\mathcal{L}_{\text{Q}}$ defined in the base offline RL algorithm.

Different state decompositions can result in different compositional input spaces, resulting in different generalization capabilities. In order to satisfy the assumptions of bilinear transduction 2.2, the ideal approach is to find the decomposition that fulfills these two criteria: in-distribution anchor and in-distribution delta. This ideal case is illustrated in Figure 1c. However, in contrast to the prior works (Netanyahu et al., 2023; Pinneri et al., 2023) that only focus on the transducible property of the goal state, we try to handle each state in every step, and it is computationally infeasible to enforce these constraints using brute-force methods like comparing the current states to all other points. Hence, we introduce a new anchor-seeking policy that learns to find in-distribution anchors and deltas and prevent arbitrary decomposition using the learned dynamics model, to exploit the power of bilinear transduction. We also enforce each delta to be within a distance of a few steps of the dynamics model to confine the distribution of delta in both the train and test phase to a similar range. This approach reduces the input space and guides it toward space that was predominantly explored during the training phase, thereby further enhancing generalizability.

3.2 Learning to Seek In-Distribution Decomposition

In this section, we list and describe the additional model, policy, and augmented dataset required before training the anchor-seeking policy. These include the reverse dynamics model, the anchor-seeking trajectory, and the random divergent reverse policy, with detailed explanations provided on how they are utilized.

3.2.1 Anchor-Seeking Trajectory

Training a reverse dynamics model. Given a transition (s, a, s') sampled from the dataset $\mathcal{D}_{\text{env}}$, we train a reverse transition dynamics model $\widehat{T}_r(s|s', a)$ (Wang et al., 2021; Lai et al., 2020; Goyal et al., 2018; Edwards et al., 2018; Holyoak & Simon, 1999) to predict the state s given the next state s' and action a. q This is done by minimizing the loss with the dataset's state s. The loss function is defined as:

$$\mathcal{L}_{\text{r}} = \mathbb{E}_{(s,a,s')\sim\mathcal{D}_{\text{env}}}\left[\left\|\widehat{T}_r(s', a) - s\right\|_2^2\right] \tag{4}$$

In other words, the reverse dynamics model $T(s', a)$ predicts "From which state s did we come if we arrived at s' by taking action a?".

Random divergent reverse policy. We do not use a trained reverse policy, but instead, we use a heuristic reverse policy that randomly selects an action from the dataset $\mathcal{D}_{\text{env}}$. The subsequent actions in reverse rollouts, after the initial action, follow the same direction as the initial action but are slightly scaled down and have a small Gaussian noise added. This ensures that the reverse rollout diverges away from the dataset. Since we use random actions and maintain a consistent direction throughout the reverse rollout, it is more likely to venture into unexplored regions beyond the offline dataset.

In sum, the reverse policy gives the action a_j at each rollout step j as follows:

$$\begin{aligned} a_j &= \phi a + \epsilon_j, \quad j = 1, 2, \ldots, h \\ a &\sim \mathcal{D}_{\text{env}}, \epsilon_j \sim \mathcal{N}(0, \sigma^2) \end{aligned} \tag{5}$$

where h is the horizon length, ϕ is the scale coefficient, and σ is the noise coefficient. We use 0.8 for ϕ and 0.1 for σ when the maximum action value is 1.0.

Anchor-seeking trajectory. We use rollouts of the reverse model to make anchor-seeking trajectories, for training the anchor-seeking policy. First, we sample the anchor state from the dataset and generate a reverse transition $\mathcal{D}_{\text{reverse}} = \{(s_{i+1}, a_i, s_i, r_i)\}_{i=1}^{j}$ from the anchor state using the reverse dynamics model and the random divergent reverse policy. Note that, the direction of anchor-seeking trajectory is reverse to that of reverse transition, $\mathcal{D}_{\text{reverse}}$.

Utilizing reverse model rollouts to address the OOD problem was first proposed by Wang et al. (2021). They augmented the offline dataset with reverse transition, trained a policy using this augmented dataset, and demonstrated the efficacy of such an approach in the offline RL setting. By doing so, we can effectively generate anchor-seeking trajectories for anchor-seeking training. The detail of generating anchor-seeking trajector is described in Algorithm 1.

Algorithm 1 Generation of Anchor-Seeking Trajectory

Require: Offline dataset $\mathcal{D}_{\text{env}}$, reverse dynamics $\widehat{T}_r$, anchor-seeking horizon h, rollout epoch e
1: **for** k in $1 \ldots e$ **do**
2: Sample anchor state $s_t \sim \mathcal{D}_{\text{env}}$
3: Generate reverse model rollout $\widehat{\tau} = \{(s_{t-i}, a_{t-i}, r_{t-i}, s_{t+1-i})\}_{i=1}^{h}$ from s_t by using the reverse dynamics $\widehat{T}_r$ and random divergent actions $\{a_{t-i}\}_{i=1}^{h}$
4: Add model rollouts to replay buffer, $\mathcal{D}_{\text{reverse}} \leftarrow \mathcal{D}_{\text{reverse}} \cup \{(s_{t-i}, a_{t-i}, r_{t-i}, s_{t+1-i})\}_{i=1}^{h}$
5: **end for**
6: **return** $\mathcal{D}_{\text{reverse}}$

3.2.2 Dynamics-Aware Anchor-Seeking

Training the Anchor-Seeking Policy. We train the anchor-seeking policy $\tilde{\pi}(a|s)$ before training the main policy. During the training phase, we utilized anchor-seeking trajectory which is the reverse direction to the dataset D_{reverse}.

By following the path of the anchor-seeking trajectory, the anchor-seeking policy is trained to select actions η that guide the agent in a direction moving from the external boundary towards the seen area illustrated in Figure 1a. Given, the anchor-seeking rollout is generated by the anchor-seeking policy $\tilde{\pi}(a|s)$ and the dynamics model $\widehat{T}(s, a)$, the reverse transition D_{reverse} relies on:

$$\widehat{T}_r(s, r|s', a)$$

Given that the reverse transition was designed to diverge from the offline dataset, the anchor-seeking trajectory, with its reversed direction, ensures the transition converges back to the dataset from unfamiliar states. Thus, we train the anchor-seeking policy to minimize the MSE loss between the predicted action and the action in the dataset D_{reverse}.

The loss function is defined as:

$$\mathcal{L}_{\text{anchor}}(\theta) = \mathbb{E}_{\substack{(s',a,s)\sim\mathcal{D}_{\text{reverse}} \\ \eta\sim\tilde{\pi}_\theta(a|s)}} \left[(\eta - a)^2\right] \tag{6}$$

In this way, The anchor-seeking policy $\tilde{\pi}(a|s)$ can provide proper action inducing current state to find the proper anchor which is in-distributed data, and that action is utilized by dynamics model $\widehat{T}(s,a)$ for the next state. Thereby, the anchor-seeking rollout (Figure 1) is a sequence of transitions that starts from the current state s and ends at the anchor state $\tilde{s}$.

Algorithm 2 Anchor-Seeking Rollout

Require: Current state s, anchor-seeking policy $\tilde{\pi}$, forward dynamics model $\widehat{T}$, anchor-seeking horizon h

Set the initial state for anchor-seeking: $\tilde{s} = s$
for $i = 1$ to h **do**
    Compute the anchor-seeking action $\eta \leftarrow \tilde{\pi}(\tilde{s})$
    Update anchor $\tilde{s} \leftarrow \widehat{T}(\tilde{s}, \eta)$
end for
return $\tilde{s}$

3.3 Method Summary

In this section, we illustrate the integration of the anchor-seeking model into the bilinear transduction framework. We choose the Soft Actor-Critic (SAC) algorithm as a representative RL algorithm that employs function approximators. In our formulation, we utilize the bilinear transduction, supplemented with anchor-seeking, into both the actor and critic networks of SAC.

Given an input state s_n, we use rollout to obtain the action from the anchor-seeking policy. The anchor $\tilde{s}$ is derived by this action through the forward step of dynamics. This anchor is then updated to be the next state s_{n+1}, and the process iterates to determine the following anchor. After rolling out a few times in a row, we can pinpoint the final anchor and use this anchor to decompose the state into the anchor and delta. The difference between the initial state s_n and this anchor $\tilde{s}$, delta, is computed as $\Delta s = \tilde{s} - s_n$.

We then carry out the bilinear transduction as described in Equation 2. Here, we embed Δs and $\tilde{s}$ as $\boldsymbol{\varphi_1}(\Delta s)$ and $\boldsymbol{\varphi_2}(\tilde{s})$, respectively, and compute the inner product of these two embeddings. The output is then fed into a small MLP layer to enhance the flexibility of the function approximator. This step is necessary to introduce nonlinearity, as the policy or Q-function may not be linear to their inputs.

For a detailed description of this process, please refer to Algorithm 3 which outlines the actor module. For the critic module's forward operation, we concatenate the action to both the anchor and delta before executing bilinear transduction. Subsequently, the derived values $\bar{f} = \widehat{a}$ and $\bar{f} = \widehat{Q}$ are employed to update the actor and critic networks of the SAC policy.

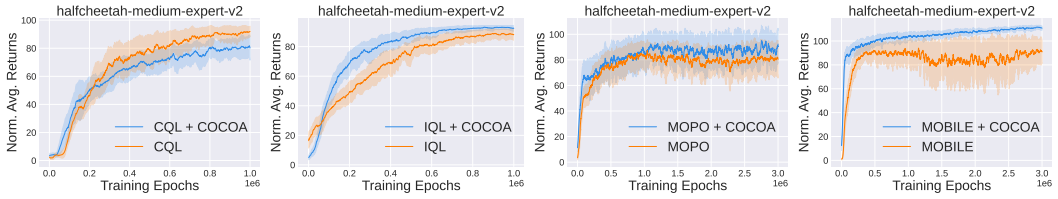
Algorithm 3 Bilinear Transduction with Anchor-Seeking (Actor)

Require: Input state s, anchor-seeking policy $\tilde{\pi}$, forward dynamics model $\widehat{T}$, anchor-seeking horizon h, embedding layers $\{\boldsymbol{\varphi_1}, \boldsymbol{\varphi_2}\}$

Set $s_0 \leftarrow s$
Perform anchor-seeking rollout $\widehat{\tau} = \{(s_i, a_i, s_{i+1})\}_{i=0}^{h-1}$ by using the anchor-seeking policy $\tilde{\pi}(a_i|s_i)$ and forward dynamics $\widehat{T}(s_{i+1}|s_i, a_i)$
Decompose the state s by using the final state as an anchor: $\tilde{s} \leftarrow s_h$, $\Delta s \leftarrow s - s_h$
Bilinear transduction of the target function: $\bar{f} \leftarrow \boldsymbol{\varphi_1}(s - \tilde{s}) \cdot \boldsymbol{\varphi_2}(\tilde{s})$
Process $\bar{f}$ with an additional MLP layer: $\bar{f} \leftarrow \text{MLP}(f)$
return $\bar{f}$

Table 1: D4RL benchmark results. We report the normalized average return of the last 10 training epochs across 4 seeds on the D4RL benchmark tasks.

Task	BC	CQL		IQL		MOPO		MOBILE	
		Alone	+COCOA	Alone	+COCOA	Alone	+COCOA	Alone	+COCOA
halfcheetah-random	2.2	31.3	8.9 ± 0.6	-	-	37.3	24.0 ± 13.5	42.5	39.8 ± 1.1
hopper-random	3.7	5.3	**8.6 ± 1.0**	-	-	31.7	**32.8 ± 1.4**	7.6	**20.3 ± 11.3**
walker2d-random	1.3	5.4	2.7 ± 0.3	-	-	4.1	**8.4 ± 8.6**	9.3	**21.4 ± 0.2**
halfcheetah-medium	43.2	46.9	**50.6 ± 0.2**	47.4	**48.1 ± 0.2**	72.4	**73.0 ± 4.7**	73.5	**73.7 ± 0.9**
hopper-medium	54.1	61.9	**63.7 ± 1.8**	66.3	62.6 ± 2.8	62.8	39.6 ± 2.4	81.9	**107.0 ± 0.0**
walker2d-medium	70.9	79.5	**82.8 ± 0.2**	78.3	76.6 ± 1.0	84.1	71.8 ± 8.9	80.0	**84.7 ± 1.2**
halfcheetah-medium-replay	37.6	45.3	**46.6 ± 0.3**	44.2	38.2 ± 1.3	72.1	72.1 ± 1.4	68.8	**69.1 ± 1.1**
hopper-medium-replay	16.6	86.3	**94.5 ± 2.2**	94.7	62.8 ± 8.7	92.8	56.6 ± 14.5	100.2	**107.0 ± 0.7**
walker2d-medium-replay	20.3	76.8	**85.7 ± 0.8**	73.9	45.2 ± 5.8	85.2	**92.3 ± 1.9**	91.3	86.7 ± 0.4
halfcheetah-medium-expert	44.0	95.0	80.2 ± 3.1	86.7	**92.4 ± 1.0**	83.6	**85.6 ± 9.6**	91.4	**110.5 ± 1.5**
hopper-medium-expert	53.9	96.9	**104.2 ± 4.7**	91.5	**104.0 ± 2.5**	74.9	**86.2 ± 29.8**	112.5	**111.8 ± 0.8**
walker2d-medium-expert	90.1	109.1	**109.5 ± 0.5**	109.6	109.0 ± 0.0	105.3	**110.5 ± 1.1**	112.2	112.0 ± 1.2



(a) CQL (b) IQL (c) MOPO (d) MOBILE

Figure 2: Performance of CQL, IQL, MOPO, MOBILE with and without COCOA on D4RL halfcheetah-medium-expert-v2 task.

4 Experiments

We aim to empirically answer the following two questions: a) how much does our method improve the performance of prior model-free and model-based algorithms?, and b) what is the effect of the anchor-seeking on performance?

4.1 D4RL Benchmark Tasks

We evaluate our method on the Gym-MuJoCo tasks in D4RL benchmark (Fu et al., 2020), which consists of 12 tasks from the OpenAI Gym (Brockman et al., 2016) and MuJoCo (Todorov et al., 2012) environments. Refer to A.1 for the details of the tasks.

Baselines. We apply COCOA to several prior offline RL algorithms, both model-based and model-free approaches. These include CQL (Kumar et al., 2020), which penalizes Q-values on out-of-distribution samples for safety; IQL (Kostrikov et al., 2022), which leverages the generalization capabilities of the function approximator by viewing the state value function as a random variable; MOPO (Yu et al., 2020b), a model-based approach that penalizes rewards based on uncertainty from predicting subsequent states; and MOBILE (Sun et al., 2023), which quantify uncertainty through the inconsistency of Bellman estimations using an ensemble of dynamics models. We also include the results of Behavior Cloning (BC), which learns tasks by imitating expert data.

Results. The results of our experiments are summarized in Table 1. The baseline algorithms are denoted as "Alone", and our method is denoted as "+COCOA". We report the average return of the last 10 training epochs across 4 seeds, with the standard deviation. FFor IQL, we refer to the results from the original papers. For MOBILE, we reproduce the results with the codebase described in Appendix A.3 For CQL and MOPO, which originally use the MuJoCo-v0 dataset, we reference the results reproduced with the MuJoCo-v2 dataset as reported in the MOBILE paper. Our method improves the performance of 9 out of 12 tasks in CQL, 3 out of 9 tasks in IQL, 7 out of 12 tasks in MOPO, and 10 out of 12 tasks in MOBILE.

4.2 Ablation Study: The Effect of Anchor-Seeking

To examine the impact of anchor selection on performance, we experiment with a variant of our method that does not use anchor-seeking. For this ablation study, we use CQL (Kumar et al., 2020) as the base algorithm and evaluate with D4RL benchmark tasks.

Baselines. Instead of using the anchor-seeking policy, we follow the anchor selection procedure in Netanyahu et al. (2023) with some modifications to reduce the computation cost. This variant is denoted as "+COCOA (w/o A.S.)" in Table 2.

We first take N number of candidate anchors s_i from the dataset and compute the difference, Δs, between the candidate anchors and the current state.

$$\Delta s_n = s - s_n, \quad n \in \{1, \dots, N\}, \quad s_n \in D_{\text{env}} \tag{7}$$

Then, we compare this delta with the difference between two arbitrary states in the offline dataset D_{env}, which is computed with N sample states.

$$\Delta s_{i,j} = s_i - s_j, \quad i, j \in \{1, \dots, N\}, \quad i \neq j, \quad s_i, s_j \in D_{\text{env}} \tag{8}$$

Then we select the candidate anchor that has the closest distance to the current state.

$$\arg\min_n \left\{ \min_{i,j} \|\Delta s_n - \Delta s_{i,j}\| \right\} \tag{9}$$

This method enforces the results of state decomposition to be close to in-distribution data through direct distance calculation. While it can be strong if the dataset is small and N is sufficiently large, it is not scalable as the amount of computation increases with the amount of data. Since the computation cost escalates in a cubic manner to the sample size, we set N to 30.

Table 2: Ablation study for anchor-seeking. We report the normalized average return of the last 10 training epochs across 4 seeds on the D4RL benchmark tasks.

Task	CQL		
	Alone	+COCOA (w/o A.S.)	+COCOA
halfcheetah-random	**31.3**	22.5 ± 0.5	8.9 ± 0.6
hopper-random	5.3	25.8 ± 7.4	**8.6 ± 1.0**
walker2d-random	5.4	**8.7 ± 5.3**	2.7 ± 0.3
halfcheetah-medium	46.9	47.6 ± 0.2	**50.6 ± 0.2**
hopper-medium	61.9	54.0 ± 2.4	**63.7 ± 1.8**
walker2d-medium	79.5	80.3 ± 0.9	**82.8 ± 0.2**
halfcheetah-medium-replay	45.3	45.1 ± 0.3	**46.6 ± 0.3**
hopper-medium-replay	86.3	84.7 ± 2.9	**94.5 ± 2.2**
walker2d-medium-replay	76.8	78.7 ± 2.2	**85.7 ± 0.8**
halfcheetah-medium-expert	**95.0**	15.8 ± 3.2	80.2 ± 3.1
hopper-medium-expert	96.9	30.4 ± 9.8	**104.2 ± 4.7**
walker2d-medium-expert	109.1	86.8 ± 2.3	**109.5 ± 0.5**

Results. We examine whether this variant improves the performance of CQL. The results are summarized in Table 2. We report the average return of the last 10 training epochs across 4 seeds, with the standard deviation. As shown in the table, "+COCOA (w/o A.S.)" achieves a higher performance in only one task, "walker2d-random", and comparable or lower performance in the other tasks compared to the baseline. In contrast, our method, "+COCOA", improves the performance in 9 out of 12 tasks. This suggests that the anchor-seeking is a crucial element of our method.

5 Related Work

5.1 Offline RL

In offline RL, agents use a predefined dataset without additional interactions, typically following either the model-based or model-free strategy. Model-free RL algorithms (Kim & Oh, 2023; Ran et al., 2023; Kostrikov et al., 2022; Kumar et al., 2020; Wu et al., 2019; Kumar et al., 2019; Fujimoto et al., 2019) optimize policy directly using prior experiences (replay buffer) and data, applying conservatism to the value function or policy. In contrast, model-based offline RL uses a model trained in the environment to create an additional dataset, which is then employed for policy learning. Through this synthesized data, the approach becomes stronger in generalization, becoming robust even to unseen states. Previous model-based offline RL algorithms (Sun et al., 2023; Rigter et al., 2022; Wang et al., 2021; Yu et al., 2021; 2020b; Kidambi et al., 2020) have been able to achieve significant performance improvements than before.

5.2 Out-of-Distribution Generalization in Offline RL

Many works explicitly tried to improve the out-of-distribution (OOD) generalization of offline RL algorithms. Lou et al. (2022) tackled the action distributional shift problem by introducing an action embedding model, employing a mutual information-based approach to learn this model. In a similar pursuit, Gu et al. (2022) proposed a pseudometric action representation learning method that measures both behavioral relation and data-distributional relation between actions. Pitis et al. (2022) developed a method to improve the generalization of offline RL algorithms by local factorization of transition dynamics and state augmentation. They also provided theoretical proofs for sample complexity and generalization ability. Our method is similar to theirs in that we also employ the factorized architecture of the policy and Q-function. Unlike them, however, we do not use a factorized dynamics model and instead leverage the bilinear transduction framework. Bai et al. (2022) is an uncertainty-driven method that uses the disagreement of bootstrapped Q-functions. It involves augmenting the dataset with OOD data points to impose a more refined penalty on these points. Similar to them, we also aim to address the state distributional shift problem by state augmentation. However, our method is different in that we exploit the reverse model rollouts and that we use them as a guide to select the anchor.

5.3 Compositional Generalization and Extrapolation

Compositional generalization, which strives to generalize to unseen combinations of components, is explored by various studies. Wiedemer et al. (2023) highlight a two-step generative procedure as essential for tackling a wide range of compositional problems. This procedure involves the complex generation of individual components and their straightforward combination into a single output. They provided a set of sufficient conditions under which models trained on the data can generalize compositionally. On a related note, Shah et al. (2020) presented a sample-efficient RL algorithm that exploits the low-rank structure of the optimal Q-function, which is a bilinear function of state and action. They proved a quantitative sample complexity improvement for RL with continuous state and action spaces via low-rank structure. Dong & Ma (2023) explored the extrapolation of nonlinear models for structured domain shift. They proved that a specific family of nonlinear models can successfully extrapolate to unseen distributions, provided the feature covariance is well-conditioned. (Netanyahu et al., 2023) proposed an extrapolation strategy based on bilinear embeddings to enable combinatorial generalization, thereby addressing the out-of-support problem under certain conditions.

6 Conclusion

In conclusion, we explored a new perspective of conservatism for offline RL, that does not regard the behavior space of the agent but the compositional input space of the policy and Q-function. We propose a practical framework, COCOA, for finding the better decomposition of states to encourage such conservatism. COCOA is a simple yet effective approach that can be applied to any offline RL algorithm that utilizes a function approximator. We empirically find that our method generally enhanced the performance of offline RL algorithms through our experiments across various tasks in Gym-MuJoCo environment of the D4RL benchmark.

As our study primarily engages in empirical exploration, the mechanism behind the performance improvement or the properties of the compositional input space may demand further investigation for a more comprehensive understanding. Moreover, the experiments were limited to control-based robotics environments with continuous state and action spaces. A valuable extension of this work would involve applying the compositional conservatism framework to other domains, including those with discrete action spaces, image-based environments, or environments with highly complex dynamics.

7 Reproducibility Statement

For reproducibility, we provide the demo code of our method in the supplementary material. About the codebase for the baseline algorithms, please refer to the Appendix A.3. The hyperparameters and the model architecture are described in Appendix A.4 and Appendix A.2, respectively.

REFERENCES

Chenjia Bai, Lingxiao Wang, Zhuoran Yang, Zhi-Hong Deng, Animesh Garg, Peng Liu, and Zhaoran Wang. Pessimistic bootstrapping for uncertainty-driven offline reinforcement learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=Y4cs1Z3HnqL.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning, 2020.

Kefan Dong and Tengyu Ma. First steps toward understanding the extrapolation of nonlinear models to unseen domains. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=7wrq3vHcMM.

Ashley D Edwards, Laura Downs, and James C Davidson. Forward-backward reinforcement learning. *arXiv preprint arXiv:1803.10227*, 2018.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

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

Omer Gottesman, Fredrik Johansson, Matthieu Komorowski, Aldo Faisal, David Sontag, Finale Doshi-Velez, and Leo Anthony Celi. Guidelines for reinforcement learning in healthcare. *Nature medicine*, 25(1):16–18, 2019.

Anirudh Goyal, Philemon Brakel, William Fedus, Soumye Singhal, Timothy Lillicrap, Sergey Levine, Hugo Larochelle, and Yoshua Bengio. Recall traces: Backtracking models for efficient reinforcement learning. *arXiv preprint arXiv:1804.00379*, 2018.

Pengjie Gu, Mengchen Zhao, Chen Chen, Dong Li, Jianye Hao, and Bo An. Learning pseudometric-based action representations for offline reinforcement learning. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 7902–7918. PMLR, 2022.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Keith J Holyoak and Dan Simon. Bidirectional reasoning in decision making by constraint satisfaction. *Journal of Experimental Psychology: General*, 128(1):3, 1999.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33: 21810–21823, 2020.

Byeongchan Kim and Min-hwan Oh. Model-based offline reinforcement learning with count-based conservatism. 2023.

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

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

Hang Lai, Jian Shen, Weinan Zhang, and Yong Yu. Bidirectional model-based policy optimization. In *International Conference on Machine Learning*, pp. 5618–5627. PMLR, 2020.

Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning: State-of-the-art*, pp. 45–73. Springer, 2012.

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

Xingzhou Lou, Qiyue Yin, Junge Zhang, Chao Yu, Zhaofeng He, Nengjie Cheng, and Kaiqi Huang. Offline reinforcement learning with representations for actions. *Information Sciences*, 610:746–758, 2022.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PMLR, 2016.

Aviv Netanyahu, Abhishek Gupta, Max Simchowitz, Kaiqing Zhang, and Pulkit Agrawal. Learning to extrapolate: A transductive approach. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=lid14UkLPd4.

Cristina Pinneri, Sarah Bechtle, Markus Wulfmeier, Arunkumar Byravan, Jingwei Zhang, William F. Whitney, and Martin Riedmiller. Equivariant data augmentation for generalization in offline reinforcement learning, 2023.

Silviu Pitis, Elliot Creager, Ajay Mandlekar, and Animesh Garg. Mocoda: Model-based counterfactual data augmentation. *Advances in Neural Information Processing Systems*, 35:18143–18156, 2022.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Yuhang Ran, Yi-Chen Li, Fuxiang Zhang, Zongzhang Zhang, and Yang Yu. Policy regularization with dataset constraint for offline reinforcement learning. In *International Conference on Machine Learning*, 2023.

Marc Rigter, Bruno Lacerda, and Nick Hawes. Rambo-rl: Robust adversarial model-based offline reinforcement learning. *Advances in neural information processing systems*, 35:16082–16097, 2022.

Devavrat Shah, Dogyoon Song, Zhi Xu, and Yuzhe Yang. Sample efficient reinforcement learning via low-rank matrix estimation. *Advances in Neural Information Processing Systems*, 33:12092–12103, 2020.

Yihao Sun. Offlinerl-kit: An elegant pytorch offline reinforcement learning library. https://github.com/yihaosun1124/OfflineRL-Kit, 2023.

Yihao Sun, Jiaji Zhang, Chengxing Jia, Haoxin Lin, Junyin Ye, and Yang Yu. Model-Bellman inconsistency for model-based offline reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 33177–33194. PMLR, 23–29 Jul 2023.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Jianhao Wang, Wenzhe Li, Haozhe Jiang, Guangxiang Zhu, Siyuan Li, and Chongjie Zhang. Offline reinforcement learning with reverse model-based imagination. *Advances in Neural Information Processing Systems*, 34:29420–29432, 2021.

Thaddäus Wiedemer, Prasanna Mayilvahanan, Matthias Bethge, and Wieland Brendel. Compositional generalization from first principles. *arXiv preprint arXiv:2307.05596*, 2023.

Marco A Wiering and Martijn Van Otterlo. Reinforcement learning. *Adaptation, learning, and optimization*, 12(3):729, 2012.

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

Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2636–2645, 2020a.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020b.

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

A Experiment Settings and Implementation Details

A.1 D4RL Benchmark Tasks

HalfCheetah: The half-cheetah is a two-dimensional bipedal robot composed of 8 solid links, encompassing two legs and a torso, coupled with 6 motorized joints. The state space is 17-dimensional, encompassing both joint angles and velocities. An adversary destabilizes it by exerting a 6-dimensional action with 2-dimensional forces on the torso and each foot.

Hopper: The hopper is a planar monopod robot, assembled with 4 solid links that represent the torso, upper leg, lower leg, and foot, and includes 3 motorized joints. It has an 11-dimensional state space including joint angles and velocities. An adversary employs a 2-dimensional force on the foot to disrupt its stability.

Walker2D: The walker operates as a two-dimensional bipedal robot with a structure of 7 links, representing two legs and a torso, along with 6 actuated joints. Within its 17-dimensional state space, joint angles and velocities are included. An adversary employs a 4-dimensional action with 2-dimensional forces on both feet to disrupt its equilibrium.

A.2 Model Architecture

Dynamics Model Architecture: As with previous works, we used a neural network as the backbone for our dynamics model, which outputs a Gaussian distribution for the next state and reward. By ensembling these networks, we achieved greater stability and enhanced performance. From an ensemble of seven, we selected the top five models based on validation error. The backbone of the dynamics model comprises four layers, each with a hidden dimension of 200.

Actor & Critic Architecture: The actor-critic framework like SAC (Haarnoja et al., 2018) comprise actor and critic modules. Typically, an actor possesses a backbone constructed from a neural network. Features embedded within this backbone are relayed through a last layer that outputs a Gaussian distribution, yielding a non-deterministic result. Although MOPO, MOBILE, CQL, and IQL (Yu et al., 2020b; Sun et al., 2023; Kostrikov et al., 2022; Kumar et al., 2020), traditionally use 2, 2, 3, and 2 backbone layers with a dimension of 256 respectively, upon integrating COCOA, we standardized the use of two backbone layers with 100 hidden dimensions.

Anchor-seeking Policy Architecture: The anchor-seeking policy acts as an add-on module, shared between the actor and critic. The input data, consisting of delta and anchor, is embedded through a neural network and subsequently processed by a bilinear architecture. Initially, inputs are embedded to the dimension of 64 with two neural networks, and the bilinear architecture produces an output

with the dimension of 4 using those embedded features. Then, these embedded features from bilinear architecture are channeled through the actor and critic backbone architectures, resulting in the determination of the action and Q value, respectively.

Parameter Size: The anchor-seeking policy is built upon a compact neural network. For model-based algorithms like MOPO and MOBILE, the dynamics parameter size is approximately 1.9M, similar to that of COCOA. However, the parameter size needed for training the actor and critic for MOPO and MOBILE is identical to 0.21M. However, when COCOA is added to these Algorithms, the size of the parameter decreases to 0.19M. Given the significant magnitude of the dynamics parameters, the cumulative parameter requirement for training across COCOA-added model-based algorithms consistently stands at 2.2M. In contrast, IQL+COCOA and CQL+COCOA, which operate without a dynamics model, each have a parameter size of 2.0M.

A.3 Code Implementation

Our method is designed as an add-on enhancement to existing offline RL algorithms. Consequently, rather than developing a new implementation, we adapted the established codebases of base algorithms. For consistent and reliable code adaptation, we relied on Sun (2023) as the foundation for all base algorithms, including CQL (Kumar et al., 2020), IQL (Kostrikov et al., 2022), MOPO (Yu et al., 2020b) and MOBILE (Sun et al., 2023). The reliability of this codebase is supported by detailed training logs and results that align with those in the original papers. Additionally, Sun (2023) offers results for the Gym-MuJoCo-v2 datasets not present in the original CQL and MOPO papers, satisfying our needs. An interesting point is that this codebase is furnished by an author of MOBILE (Sun et al., 2023). Our adaptations to the code are shared as a demo in the supplementary material.

A.4 Hyperparameters for each Algorithm

CQL. For both CQL and CQL+COCOA, we use $\alpha = 5.0$ for all D4RL-Gym tasks because the reproduced codebase (Sun, 2023) which provides the results for MuJoCo-v2 tasks, which are not included in the original paper (Kumar et al., 2020), uses this value. For COCOA, the anchor seeking horizon length h was set to 1 for most tasks, except for "halfcheetah-medium-expert-v2", "hopper-medium-expert-v2", and "walker2d-medium-expert-v2", where h was set to 3.

IQL. For both IQL and IQL+COCOA, we use the same hyperparameters as described in the original paper (Kostrikov et al., 2022), $\tau = 0.7$ and $\beta = 0.3$, which is also used in the reproduced codebase (Sun, 2023). For COCOA, the anchor seeking horizon length h was set to 1 for all tasks.

MOPO. For MOPO, we use the hyperparameters used in the reproduced codebase (Sun, 2023) which provides the results for MuJoCo-v2 tasks, which are not included in the original paper (Yu et al., 2020b). Note that we use aleatoric uncertainty for both MOPO and MOPO+COCOA, as in the original paper. For MOPO+COCOA, we search for the best penalty coefficient λ and rollout length h_r for each task in the following ranges: $\lambda \in \{0.1, 0.5, 1.0, 2.5, 5.0, 10.0\}$, $h_r \in \{1, 5, 7, 10, 15\}$. The best hyperparameters are described in Table 3. For COCOA, the anchor seeking horizon length h was set to 1 for all tasks.

MOBILE. For MOBILE, we use the same hyperparameters as described in the original paper (Sun et al., 2023). For MOBILE+COCOA, we search for the best penalty coefficient λ and rollout length h_r for each task in the following ranges: $\lambda \in \{0.1, 0.5, 1.0, 2.0\}$, $h_r \in \{1, 3, 5, 7, 10, 15\}$. The best hyperparameters are described in Table 3. For COCOA, the anchor seeking horizon length h was set to 1 for all tasks.

B Detailed Results

B.1 Performance Graphs of D4RL Benchmark Tasks

In this section, we provide the performance graphs of each algorithm on D4RL benchmark tasks. We include only the 9 tasks that are not "random" tasks because the checkpoints of the baseline methods for the "random" tasks are not provided.

Table 3: Hyperparameters for MOPO+COCOA and MOBILE+COCOA.

Task	MOPO+COCOA		MOBILE+COCOA	
	λ	h_r	λ	h_r
halfcheetah-random	0.1	7	1.0	10
hopper-random	10.0	5	0.1	1
walker2d-random	0.5	5	1.0	5
halfcheetah-medium	0.5	7	1.0	5
hopper-medium	10.0	15	1.5	10
walker2d-medium	5.0	1	1.0	10
halfcheetah-medium-replay	1.0	10	1.0	10
hopper-medium-replay	1.0	10	1.0	10
walker2d-medium-replay	0.5	10	1.0	3
halfcheetah-medium-expert	2.5	5	2.5	10
hopper-medium-expert	5.0	10	2.0	10
walker2d-medium-expert	1.0	10	1.0	10

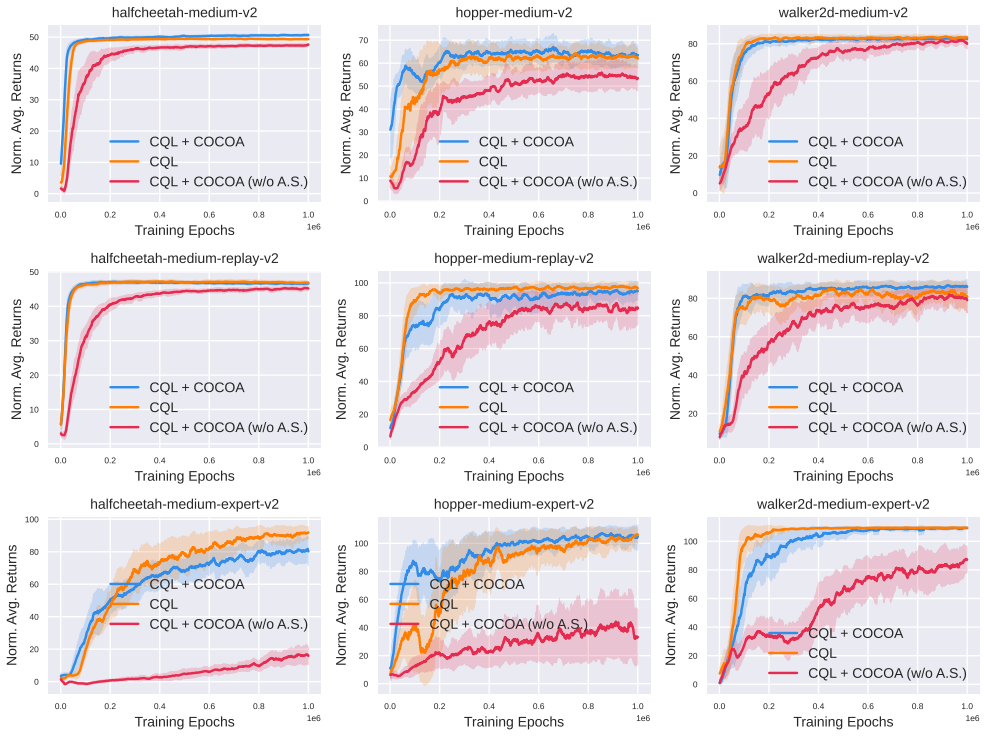


Figure 3: Performance comparison of CQL, CQL+COCOA and CQL+COCOA without anchor-seeking across all D4RL tasks except for "random" tasks.

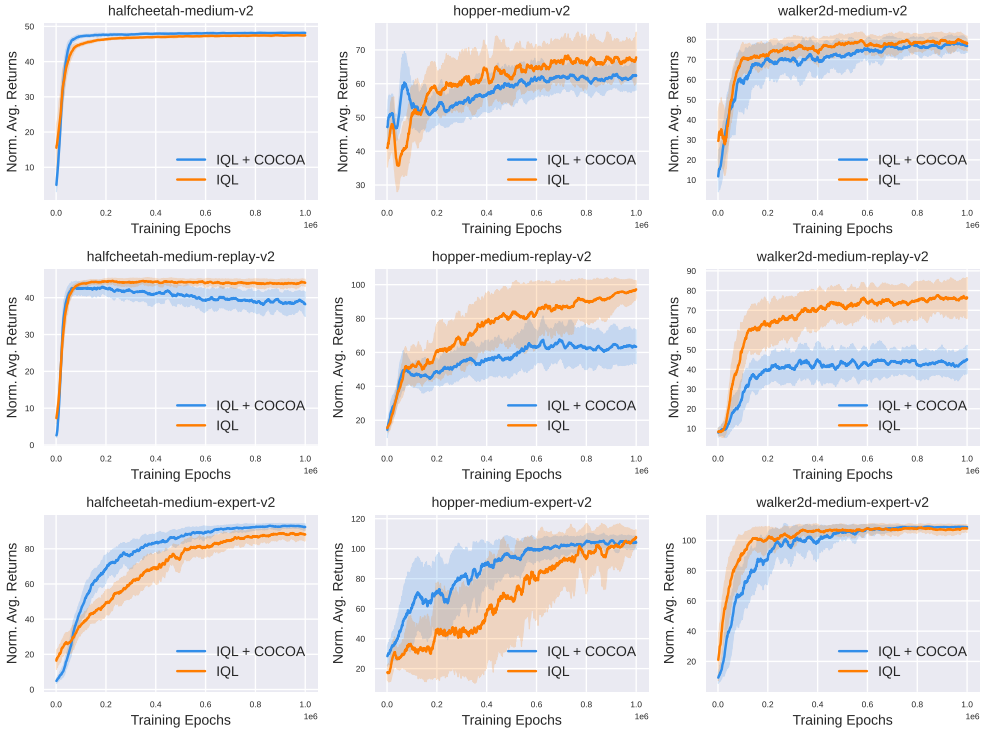


Figure 4: Performance comparison of IQL and IQL+COCOA across all D4RL tasks except for "random" tasks.

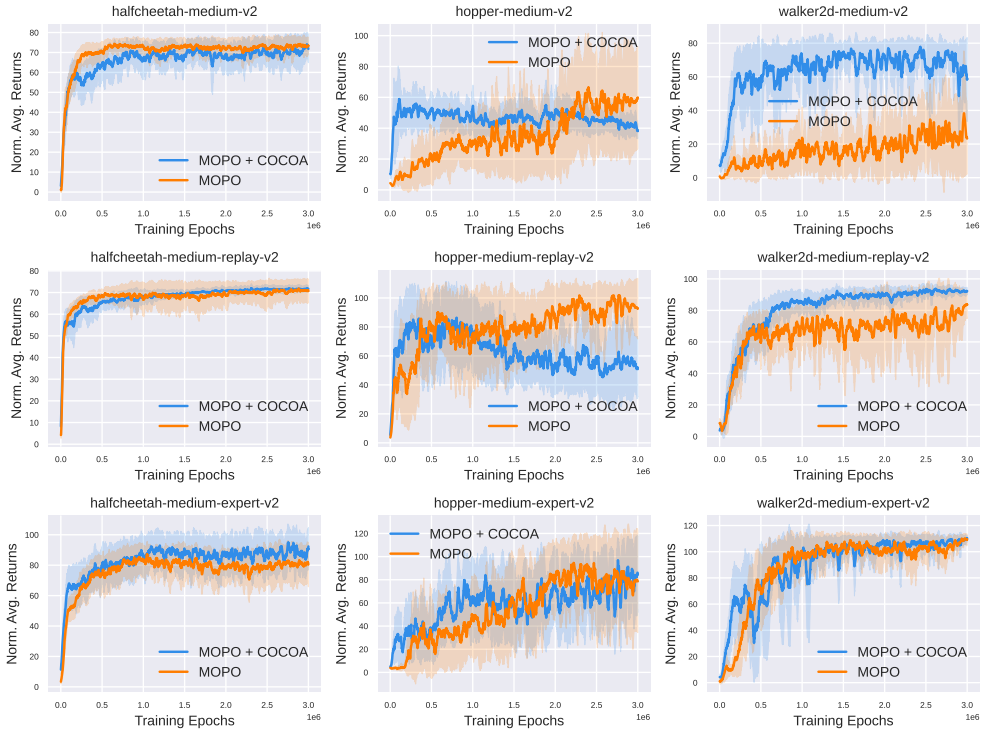


Figure 5: Performance comparison of MOPO and MOPO+COCOA across all D4RL tasks except for "random" tasks.

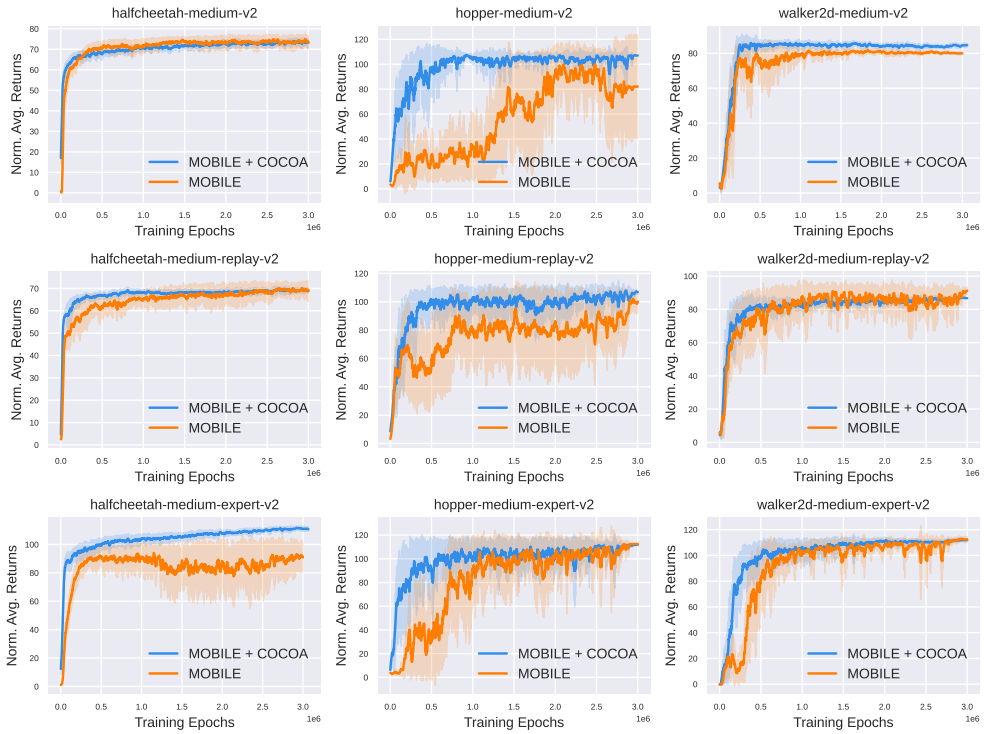


Figure 6: Performance comparison of MOBILE and MOBILE+COCOA across all D4RL tasks except for "random" tasks.