COMBO: Conservative Offline Model-Based Policy Optimization

Tianhe Yu[*,1], **Aviral Kumar**[*,2], **Rafael Rafailov**[1], **Aravind Rajeswaran**[3], **Sergey Levine**[2], **Chelsea Finn**[1]

[1]Stanford University, [2]UC Berkeley, [3]Facebook AI Research ([*]Equal Contribution)

tianheyu@cs.stanford.edu, aviralk@berkeley.edu

Abstract

Model-based reinforcement learning (RL) algorithms, which learn a dynamics model from logged experience and perform conservative planning under the learned model, have emerged as a promising paradigm for offline reinforcement learning (offline RL). However, practical variants of such model-based algorithms rely on explicit uncertainty quantification for incorporating conservatism. Uncertainty estimation with complex models, such as deep neural networks, can be difficult and unreliable. We empirically find that uncertainty estimation is not accurate and leads to poor performance in certain scenarios in offline model-based RL. We overcome this limitation by developing a new model-based offline RL algorithm, COMBO, that trains a value function using both the offline dataset and data generated using rollouts under the model while also additionally regularizing the value function on out-of-support state-action tuples generated via model rollouts. This results in a conservative estimate of the value function for out-of-support state-action tuples, without requiring explicit uncertainty estimation. Theoretically, we show that COMBO satisfies a policy improvement guarantee in the offline setting. Through extensive experiments, we find that COMBO attains greater performance compared to prior offline RL on problems that demand generalization to related but previously unseen tasks, and also consistently matches or outperforms prior offline RL methods on widely studied offline RL benchmarks, including image-based tasks.

1 Introduction

Offline reinforcement learning (offline RL) [30, 34] refers to the setting where policies are trained using static, previously collected datasets. This presents an attractive paradigm for data reuse and safe policy learning in many applications, such as healthcare [62], autonomous driving [65], robotics [25, 48], and personalized recommendation systems [59]. Recent studies have observed that RL algorithms originally developed for the online or interactive paradigm perform poorly in the offline case [14, 28, 26]. This is primarily attributed to the distribution shift that arises over the course of learning between the offline dataset and the learned policy. Thus, development of algorithms specialized for offline RL is of paramount importance to benefit from the offline data available in aforementioned applications. In this work, we develop a principled model-based offline RL algorithm that matches or exceeds the performance of prior offline RL algorithms in benchmark tasks.

A major paradigm for algorithm design in offline RL is to incorporate conservatism or regularization into online RL algorithms. Model-free offline RL algorithms [15, 28, 63, 21, 29, 27] directly incorporate conservatism into the policy or value function training and do not require learning a dynamics model. However, model-free algorithms learn only on the states in the offline dataset, which can lead to overly conservative algorithms. In contrast, model-based algorithms [26, 67] learn a pessimistic dynamics model, which in turn induces a conservative estimate of the value function. By generating and training on additional synthetic data, model-based algorithms have the potential for

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

broader generalization and solving new tasks using the offline dataset [67]. However, these methods rely on some sort of strong assumption about uncertainty estimation, typically assuming access to a *model error oracle* that can estimate upper bounds on model error for any state-action tuple. In practice, such methods use more heuristic uncertainty estimation methods, which can be difficult or unreliable for complex datasets or deep network models. It then remains an open question as to whether we can formulate principled model-based offline RL algorithms with concrete theoretical guarantees on performance *without* assuming access to an uncertainty or model error oracle. In this work, we propose precisely such a method, by eschewing direct uncertainty estimation, which we argue is not necessary for offline RL.

Our main contribution is the development of conservative offline model-based policy optimization (COMBO), a new model-based algorithm for offline RL. COMBO learns a dynamics model using the offline dataset. Subsequently, it employs an actor-critic method where the value function is learned using both the offline dataset as well as synthetically generated data from the model, similar to Dyna [57] and a number of recent methods [20, 67, 7, 48]. However, in contrast to Dyna, COMBO learns a conservative critic function by penalizing the value function in state-action tuples that are not in the support of the offline dataset, obtained by simulating the learned model. We theoretically show that for any policy, the Q-function learned by COMBO is a lower-bound on the true Q-function. While the approach of optimizing a performance lower-bound is similar in spirit to prior model-based algorithms [26, 67], COMBO crucially does not assume access to a model error or uncertainty oracle. In addition, we show theoretically that the Q-function learned by COMBO is less conservative than model-free counterparts such as CQL [29], and quantify conditions under which the this lower bound is tighter than the one derived in CQL. This is illustrated through an example in Figure 1. Following prior works [31], we show that COMBO enjoys a safe policy improvement guarantee. By interpolating model-free and model-based components, this guarantee can utilize the best of both guarantees in certain cases. Finally, in our experiments, we find that COMBO achieves the best performance on tasks that require out-of-distribution generalization and outperforms previous latent-space offline model-based RL methods on image-based robotic manipulation benchmarks. We also test COMBO on commonly studied benchmarks for offline RL and find that COMBO generally performs well on the benchmarks, achieving the highest score in 9 out of 12 MuJoCo domains from the D4RL [12] benchmark suite.

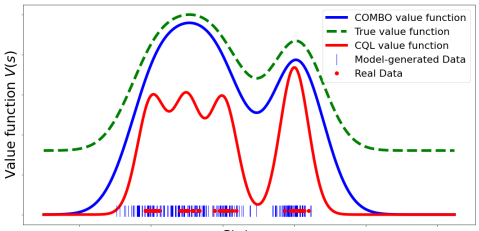

Figure 1: COMBO learns a conservative value function by utilizing both the offline dataset as well as simulated data from the model. Crucially, COMBO does not require uncertainty quantification, and the value function learned by COMBO is less conservative on the transitions seen in the dataset than CQL. This enables COMBO to steer the agent towards higher value states compared to CQL, which may steer towards more optimal states, as illustrated in the figure.

2 Preliminaries

Markov Decision Processes and Offline RL. We study RL in the framework of Markov decision processes (MDPs) specified by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, r, \mu_0, \gamma)$. $\mathcal{S}, \mathcal{A}$ denote the state and action spaces. $T(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ and $r(\mathbf{s}, \mathbf{a}) \in [-R_{\max}, R_{\max}]$ represent the dynamics and reward function respectively. $\mu_0(s)$ denotes the initial state distribution, and $\gamma \in (0, 1)$ denotes the discount factor. We denote the discounted state visitation distribution of a policy π using $d^\pi_\mathcal{M}(\mathbf{s}) := (1-\gamma)\sum_{t=0}^{\infty}\gamma^t\mathcal{P}(s_t = \mathbf{s}|\pi)$, where $\mathcal{P}(s_t = \mathbf{s}|\pi)$ is the probability of reaching state $\mathbf{s}$ at time t by rolling out π in $\mathcal{M}$. Similarly, we denote the state-action visitation distribution with $d^\pi_\mathcal{M}(\mathbf{s}, \mathbf{a}) := d^\pi_\mathcal{M}(\mathbf{s})\pi(\mathbf{a}|\mathbf{s})$. The goal of RL is to learn a policy that maximizes the return, or long term cumulative rewards: $\max_\pi J(\mathcal{M}, \pi) := \frac{1}{1-\gamma}\mathbb{E}_{(\mathbf{s},\mathbf{a})\sim d^\pi_\mathcal{M}(\mathbf{s},\mathbf{a})}[r(\mathbf{s}, \mathbf{a})]$.

Offline RL is the setting where we have access only to a fixed dataset $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, r, \mathbf{s}')\}$, which consists of transition tuples from trajectories collected using a behavior policy π_β. In other words, the dataset $\mathcal{D}$ is sampled from $d^{\pi_\beta}(\mathbf{s}, \mathbf{a}) := d^{\pi_\beta}(\mathbf{s})\pi_\beta(\mathbf{a}|\mathbf{s})$. We define $\widehat{\mathcal{M}}$ as the empirical MDP induced by the dataset $\mathcal{D}$ and $d(\mathbf{s}, \mathbf{a})$ as sampled-based version of $d^{\pi_\beta}(\mathbf{s}, \mathbf{a})$. In the offline setting, the goal is to find the best possible policy using the fixed offline dataset.

Model-Free Offline RL Algorithms. One class of approaches for solving MDPs involves the use of dynamic programming and actor-critic schemes [56, 5], which do not explicitly require the learning

of a dynamics model. To capture the long term behavior of a policy without a model, we define the action value function as $Q^\pi(\mathbf{s}, \mathbf{a}) \coloneqq \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \, r(\mathbf{s}_t, \mathbf{a}_t) \mid \mathbf{s}_0 = \mathbf{s}, \mathbf{a}_0 = \mathbf{a}\right]$, where future actions are sampled from $\pi(\cdot|\mathbf{s})$ and state transitions happen according to the MDP dynamics. Consider the following Bellman operator: $\mathcal{B}^\pi Q(\mathbf{s}, \mathbf{a}) := r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\,\mathbf{s}' \sim T(\cdot|\mathbf{s},\mathbf{a}), \mathbf{a}' \sim \pi(\cdot|\mathbf{s}')}\left[Q(\mathbf{s}', \mathbf{a}')\right]$, and its sample based counterpart: $\widehat{\mathcal{B}}^\pi Q(\mathbf{s}, \mathbf{a}) := r(\mathbf{s}, \mathbf{a}) + \gamma Q(\mathbf{s}', \mathbf{a}')$, associated with a single transition $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ and $\mathbf{a}' \sim \pi(\cdot|\mathbf{s}')$. The action-value function satisfies the Bellman consistency criterion given by $\mathcal{B}^\pi Q^\pi(\mathbf{s}, \mathbf{a}) = Q^\pi(\mathbf{s}, \mathbf{a}) \; \forall (\mathbf{s}, \mathbf{a})$. When given an offline dataset $\mathcal{D}$, standard approximate dynamic programming (ADP) and actor-critic methods use this criterion to alternate between policy evaluation [40] and policy improvement. A number of prior works have observed that such a direct extension of ADP and actor-critic schemes to offline RL leads to poor results due to distribution shift over the course of learning and over-estimation bias in the Q function [14, 28, 63]. To address these drawbacks, prior works have proposed a number of modifications aimed towards regularizing the policy or value function (see Section 6). In this work, we primarily focus on CQL [29], which alternates between:

Policy Evaluation: The Q function associated with the current policy π is approximated conservatively by repeating the following optimization:

$$Q^{k+1} \leftarrow \arg\min_{Q} \beta\left(\mathbb{E}_{\mathbf{s}\sim\mathcal{D}, \mathbf{a}\sim\mu(\cdot|\mathbf{s})}[Q(\mathbf{s}, \mathbf{a})] - \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}[Q(\mathbf{s}, \mathbf{a})]\right) + \frac{1}{2}\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\left(Q(\mathbf{s}, \mathbf{a}) - \widehat{\mathcal{B}}^\pi Q^k(\mathbf{s}, \mathbf{a})\right)^2\right], \quad (1)$$

where $\mu(\cdot|s)$ is a wide sampling distribution such as the uniform distribution over action bounds. CQL effectively penalizes the Q function at states in the dataset for actions not observed in the dataset. This enables a conservative estimation of the value function for any policy [29], mitigating the challenges of over-estimation bias and distribution shift.

Policy Improvement: After approximating the Q function as $\hat{Q}^\pi$, the policy is improved as $\pi \leftarrow \arg\max_{\pi'} \mathbb{E}_{\mathbf{s}\sim\mathcal{D}, \mathbf{a}\sim\pi'(\cdot|\mathbf{s})}\left[\hat{Q}^\pi(\mathbf{s}, \mathbf{a})\right]$. Actor-critic methods with parameterized policies and Q functions approximate $\arg\max$ and $\arg\min$ in above equations with a few gradient descent steps.

Model-Based Offline RL Algorithms. A second class of algorithms for solving MDPs involve the learning of the dynamics function, and using the learned model to aid policy search. Using the given dataset $\mathcal{D}$, a dynamics model $\widehat{T}$ is typically trained using maximum likelihood estimation as: $\min_{\widehat{T}} \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}}\left[\log \widehat{T}(\mathbf{s}'|\mathbf{s}, \mathbf{a})\right]$. A reward model $\hat{r}(\mathbf{s}, \mathbf{a})$ can also be learned similarly if it is unknown. Once a model has been learned, we can construct the learned MDP $\widehat{\mathcal{M}} = (\mathcal{S}, \mathcal{A}, \widehat{T}, \hat{r}, \mu_0, \gamma)$, which has the same state and action spaces, but uses the learned dynamics and reward function. Subsequently, any policy learning or planning algorithm can be used to recover the optimal policy in the model as $\hat{\pi} = \arg\max_\pi J(\widehat{\mathcal{M}}, \pi)$.

This straightforward approach is known to fail in the offline RL setting, both in theory and practice, due to distribution shift and model-bias [51, 26]. In order to overcome these challenges, offline model-based algorithms like MOReL [26] and MOPO [67] use uncertainty quantification to construct a lower bound for policy performance and optimize this lower bound by assuming a model error oracle $u(\mathbf{s}, \mathbf{a})$. By using an uncertainty estimation algorithm like bootstrap ensembles [43, 4, 37], we can estimate $u(\mathbf{s}, \mathbf{a})$. By constructing and optimizing such a lower bound, offline model-based RL algorithms avoid the aforementioned pitfalls like model-bias and distribution shift. While any RL or planning algorithm can be used to learn the optimal policy for $\widehat{\mathcal{M}}$, we focus specifically on MBPO [20, 57] which was used in MOPO. MBPO follows the standard structure of actor-critic algorithms, but in each iteration uses an augmented dataset $\mathcal{D} \cup \mathcal{D}_{\text{model}}$ for policy evaluation. Here, $\mathcal{D}$ is the offline dataset and $\mathcal{D}_{\text{model}}$ is a dataset obtained by simulating the current policy using the learned dynamics model. Specifically, at each iteration, MBPO performs k-step rollouts using $\widehat{T}$ starting from state $\mathbf{s} \in \mathcal{D}$ with a particular rollout policy $\mu(\mathbf{a}|\mathbf{s})$, adds the model-generated data to $\mathcal{D}_{\text{model}}$, and optimizes the policy with a batch of data sampled from $\mathcal{D} \cup \mathcal{D}_{\text{model}}$ where each datapoint in the batch is drawn from $\mathcal{D}$ with probability $f \in [0, 1]$ and $\mathcal{D}_{\text{model}}$ with probability $1 - f$.

3 Conservative Offline Model-Based Policy Optimization

The principal limitation of prior offline model-based algorithms (discussed in Section 2) is the assumption of having access to a model error oracle for uncertainty estimation and strong reliance on heuristics of quantifying the uncertainty. In practice, such heuristics could be challenging for complex datasets or deep neural network models [44]. We argue that uncertainty estimation is not

Algorithm 1 COMBO: Conservative Model Based Offline Policy Optimization

Require: Offline dataset $\mathcal{D}$, rollout distribution $\mu(\cdot|\mathbf{s})$, learned dynamics model $\widehat{T}_\theta$, initialized policy and critic π_ϕ and Q_ψ.

1: Train the probabilistic dynamics model $\widehat{T}_\theta(\mathbf{s}', r|\mathbf{s}, \mathbf{a}) = \mathcal{N}(\mu_\theta(\mathbf{s}, \mathbf{a}), \Sigma_\theta(\mathbf{s}, \mathbf{a}))$ on $\mathcal{D}$.
2: Initialize the replay buffer $\mathcal{D}_{\text{model}} \leftarrow \varnothing$.
3: **for** $i = 1, 2, 3, \cdots$, **do**
4: Collect model rollouts by sampling from μ and $\widehat{T}_\theta$ starting from states in $\mathcal{D}$. Add model rollouts to $\mathcal{D}_{\text{model}}$.
5: Conservatively evaluate π_ϕ^i by repeatedly solving eq. 2 to obtain $\hat{Q}_\psi^{\pi_\phi^i}$ using samples from $\mathcal{D} \cup \mathcal{D}_{\text{model}}$.
6: Improve policy under state marginal of d_f by solving eq. 3 to obtain π_ϕ^{i+1}.
7: **end for**

imperative for offline model-based RL and empirically show that uncertainty estimation could be inaccurate in offline RL problems especially when generalization to unknown behaviors is required in Section 5.1.1. Our goal is to develop a model-based offline RL algorithm that enables optimizing a lower bound on the policy performance, but without requiring uncertainty quantification. We achieve this by extending conservative Q-learning [29], which does not require explicit uncertainty quantification, into the model-based setting. Our algorithm COMBO, summarized in Algorithm 1, alternates between a conservative policy evaluation step and a policy improvement step, which we outline below.

Conservative Policy Evaluation: Given a policy π, an offline dataset $\mathcal{D}$, and a learned model of the MDP $\hat{\mathcal{M}}$, the goal in this step is to obtain a conservative estimate of Q^π. To achieve this, we penalize the Q-values evaluated on data drawn from a particular state-action distribution that is more likely to be out-of-support while pushing up the Q-values on state-action pairs that are trustworthy, which is implemented by repeating the following recursion:

$$\hat{Q}^{k+1} \leftarrow \arg\min_Q \beta \left(\mathbb{E}_{\mathbf{s},\mathbf{a}\sim\rho(\mathbf{s},\mathbf{a})}[Q(\mathbf{s},\mathbf{a})] - \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}[Q(\mathbf{s},\mathbf{a})]\right) + \frac{1}{2}\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim d_f}\left[\left(Q(\mathbf{s},\mathbf{a}) - \widehat{\mathcal{B}}^\pi \hat{Q}^k(\mathbf{s},\mathbf{a})\right)^2\right]. \quad (2)$$

Here, $\rho(\mathbf{s}, \mathbf{a})$ and d_f are sampling distributions that we can choose. Model-based algorithms allow ample flexibility for these choices while providing the ability to control the bias introduced by these choices. For $\rho(\mathbf{s}, \mathbf{a})$, we make the following choice: $\rho(\mathbf{s}, \mathbf{a}) = d^\pi_{\widehat{\mathcal{M}}}(\mathbf{s})\pi(\mathbf{a}|\mathbf{s})$, where $d^\pi_{\widehat{\mathcal{M}}}(\mathbf{s})$ is the discounted marginal state distribution when executing π in the learned model $\widehat{\mathcal{M}}$. Samples from $d^\pi_{\widehat{\mathcal{M}}}(\mathbf{s})$ can be obtained by rolling out π in $\widehat{\mathcal{M}}$. Similarly, d_f is an $f-$interpolation between the offline dataset and synthetic rollouts from the model: $d^\mu_f(\mathbf{s}, \mathbf{a}) := f\ d(\mathbf{s}, \mathbf{a}) + (1 - f)\ d^\mu_{\widehat{\mathcal{M}}}(\mathbf{s}, \mathbf{a})$, where $f \in [0, 1]$ is the ratio of the datapoints drawn from the offline dataset as defined in Section 2 and $\mu(\cdot|\mathbf{s})$ is the rollout distribution used with the model, which can be modeled as π or a uniform distribution. To avoid notation clutter, we also denote $d_f := d^\mu_f$.

Under such choices of ρ and d_f, we push down (or conservatively estimate) Q-values on state-action tuples from model rollouts and push up Q-values on the real state-action pairs from the offline dataset. When updating Q-values with the Bellman backup, we use a mixture of both the model-generated data and the real data, similar to Dyna [57]. Note that in comparison to CQL and other model-free algorithms, COMBO learns the Q-function over a richer set of states beyond the states in the offline dataset. This is made possible by performing rollouts under the learned dynamics model, denoted by $d^\mu_{\widehat{\mathcal{M}}}(\mathbf{s}, \mathbf{a})$. We will show in Section 4 that the Q function learned by repeating the recursion in Eq. 2 provides a lower bound on the true Q function, without the need for explicit uncertainty estimation. Furthermore, we will theoretically study the advantages of using synthetic data from the learned model, and characterize the impacts of model bias.

Policy Improvement Using a Conservative Critic: After learning a conservative critic $\hat{Q}^\pi$, we improve the policy as:

$$\pi' \leftarrow \arg\max_\pi \ \mathbb{E}_{\mathbf{s}\sim\rho,\mathbf{a}\sim\pi(\cdot|\mathbf{s})}\left[\hat{Q}^\pi(\mathbf{s},\mathbf{a})\right] \quad (3)$$

where $\rho(\mathbf{s})$ is the state marginal of $\rho(\mathbf{s}, \mathbf{a})$. When policies are parameterized with neural networks, we approximate the $\arg\max$ with a few steps of gradient descent. In addition, entropy regularization can also be used to prevent the policy from becoming degenerate if required [17]. In Section 4.2, we show that the resulting policy is guaranteed to improve over the behavior policy.

Practical Implementation Details. Our practical implementation largely follows MOPO, with the key exception that we perform conservative policy evaluation as outlined in this section, rather than using uncertainty-based reward penalties. Following MOPO, we represent the probabilistic dynamics model using a neural network, with parameters θ, that produces a Gaussian distribution over the next state and reward: $\widehat{T}_\theta(\mathbf{s}_{t+1}, r|\mathbf{s}, \mathbf{a}) = \mathcal{N}(\mu_\theta(\mathbf{s}_t, \mathbf{a}_t), \Sigma_\theta(\mathbf{s}_t, \mathbf{a}_t))$. The model is trained via maximum likelihood. For conservative policy evaluation (eq. 2) and policy improvement (eq. 3), we augment ρ with states sampled from the offline dataset, which shows more stable improvement in practice. It is relatively common in prior work on model-based offline RL to select various hyperparameters using online policy rollouts [67, 26, 3, 33]. However, we would like to avoid this with our method, since requiring online rollouts to tune hyperparameters contradicts the main aim of offline RL, which is to learn entirely from offline data. Therefore, *we do not use online rollouts for tuning COMBO*, and instead devise an automated rule for tuning important hyperparameters such as β and f in a fully offline manner. We search over a small discrete set of hyperparameters for each task, and use the value of the regularization term $\mathbb{E}_{\mathbf{s},\mathbf{a}\sim\rho(\mathbf{s},\mathbf{a})}[Q(\mathbf{s},\mathbf{a})] - \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}[Q(\mathbf{s},\mathbf{s})]$ (shown in Eq. 2) to pick hyperparameters in an entirely offline fashion. We select the hyperparameter setting that achieves the lowest regularization objective, which indicates that the Q-values on unseen model-predicted state-action tuples are not overestimated. Additional details about the practical implementation and the hyperparameter selection rule are provided in Appendix B.1 and Appendix B.2 respectively.

4 Theoretical Analysis of COMBO

In this section, we theoretically analyze our method and show that it optimizes a lower-bound on the expected return of the learned policy. This lower bound is close to the actual policy performance (modulo sampling error) when the policy's state-action marginal distribution is in support of the state-action marginal of the behavior policy and conservatively estimates the performance of a policy otherwise. By optimizing the policy against this lower bound, COMBO guarantees policy improvement beyond the behavior policy. Furthermore, we use these insights to discuss cases when COMBO is less conservative compared to model-free counterparts.

4.1 COMBO Optimizes a Lower Bound

We first show that training the Q-function using Eq. 2 produces a Q-function such that the expected off-policy policy improvement objective [8] computed using this learned Q-function lower-bounds its actual value. We will reuse notation for d_f and d from Sections 2 and 3. Assuming that the Q-function is tabular, the Q-function found by approximate dynamic programming in iteration k, can be obtained by differentiating Eq. 2 with respect to Q^k (see App. A for details):

$$\hat{Q}^{k+1}(\mathbf{s}, \mathbf{a}) = (\widehat{\mathcal{B}}^\pi Q^k)(\mathbf{s}, \mathbf{a}) - \beta \frac{\rho(\mathbf{s}, \mathbf{a}) - d(\mathbf{s}, \mathbf{a})}{d_f(\mathbf{s}, \mathbf{a})}. \tag{4}$$

Eq. 4 effectively applies a penalty that depends on the three distributions appearing in the COMBO critic training objective (Eq. 2), of which ρ and d_f are free variables that we choose in practice as discussed in Section 3. For a given iteration k of Eq. 4, we further define the expected penalty under $\rho(\mathbf{s}, \mathbf{a})$ as:

$$\nu(\rho, f) := \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\rho(\mathbf{s},\mathbf{a})}\left[\frac{\rho(\mathbf{s}, \mathbf{a}) - d(\mathbf{s}, \mathbf{a})}{d_f(\mathbf{s}, \mathbf{a})}\right]. \tag{5}$$

Next, we will show that the Q-function learned by COMBO lower-bounds the actual Q-function under the initial state distribution μ_0 and any policy π. We also show that the asymptotic Q-function learned by COMBO lower-bounds the actual Q-function of any policy π with high probability for a large enough $\beta \geq 0$, which we include in Appendix A.2. Let $\overline{\mathcal{M}}$ represent the empirical MDP which uses the empirical transition model based on raw data counts. The Bellman backups over the dataset distribution d_f in Eq. 2 that we analyze is an $f-$interpolation of the backup operator in the empirical MDP (denoted by $\mathcal{B}^\pi_{\overline{\mathcal{M}}}$) and the backup operator under the learned model $\widehat{\mathcal{M}}$ (denoted by $\mathcal{B}^\pi_{\widehat{\mathcal{M}}}$). The empirical backup operator suffers from sampling error, but is unbiased in expectation, whereas the model backup operator induces bias but no sampling error. We assume that all of these backups enjoy concentration properties with concentration coefficient $C_{r,T,\delta}$, dependent on the desired confidence value δ (details in Appendix A.2). This is a standard assumption in literature [31]. Now, we state our main results below.

Proposition 4.1. *For large enough β, we have $\mathbb{E}_{\mathbf{s}\sim\mu_0,\mathbf{a}\sim\pi(\cdot|\mathbf{s})}[\hat{Q}^\pi(\mathbf{s},\mathbf{a})] \leq \mathbb{E}_{\mathbf{s}\sim\mu_0,\mathbf{a}\sim\pi(\cdot|\mathbf{s})}[Q^\pi(\mathbf{s},\mathbf{a})]$, where $\mu_0(\mathbf{s})$ is the initial state distribution. Furthermore, when ϵ_s is small, such as in the large*

sample regime, or when the model bias ϵ_m is small, a small β is sufficient to guarantee this condition along with an appropriate choice of f.

The proof for Proposition 4.1 can be found in Appendix A.2. Finally, while Kumar et al. [29] also analyze how regularized value function training can provide lower bounds on the value function at each state in the dataset [29] (Proposition 3.1-3.2), our result shows that COMBO is less conservative in that it does not underestimate the value function at every state in the dataset like CQL (Remark 1) and might even overestimate these values. Instead COMBO penalizes Q-values at states generated via model rollouts from $\rho(\mathbf{s}, \mathbf{a})$. Note that in general, the required value of β may be quite large similar to prior works, which typically utilize a large constant β, which may be in the form of a penalty on a regularizer [36, 29] or as constants in theoretically optimal algorithms [23, 49]. While it is challenging to argue that that either COMBO or CQL attains the tightest possible lower-bound on return, in our final result of this section, we discuss a sufficient condition for the COMBO lower-bound to be tighter than CQL.

Proposition 4.2. *Assuming previous notation, let $\Delta^{\pi}_{COMBO} := \mathbb{E}_{\mathbf{s},\mathbf{a}\sim d_{\widehat{\mathcal{M}}}(\mathbf{s}),\pi(\mathbf{a}|\mathbf{s})}\left[\hat{Q}^{\pi}(\mathbf{s},\mathbf{a})\right]$ and $\Delta^{\pi}_{CQL} := \mathbb{E}_{\mathbf{s},\mathbf{a}\sim d_{\widehat{\mathcal{M}}}(\mathbf{s}),\pi(\mathbf{a}|\mathbf{s})}\left[\hat{Q}^{\pi}_{CQL}(\mathbf{s},\mathbf{a})\right]$ denote the average values on the dataset under the Q-functions learned by COMBO and CQL respectively. Then, $\Delta^{\pi}_{COMBO} \geq \Delta^{\pi}_{CQL}$, if:*

$$\mathbb{E}_{\mathbf{s},\mathbf{a}\sim\rho(\mathbf{s},\mathbf{a})}\left[\frac{\pi(\mathbf{a}|\mathbf{s})}{\pi_{\beta}(\mathbf{a}|\mathbf{s})}\right] - \mathbb{E}_{\mathbf{s},\mathbf{a}\sim d_{\widehat{\mathcal{M}}}(\mathbf{s}),\pi(\mathbf{a}|\mathbf{s})}\left[\frac{\pi(\mathbf{a}|\mathbf{s})}{\pi_{\beta}(\mathbf{a}|\mathbf{s})}\right] \leq 0. \qquad (*)$$

Proposition 4.2 indicates that COMBO will be less conservative than CQL when the action probabilities under learned policy $\pi(\mathbf{a}|\mathbf{s})$ and the probabilities under the behavior policy $\pi_{\beta}(\mathbf{a}|\mathbf{s})$ are closer together on state-action tuples drawn from $\rho(\mathbf{s}, \mathbf{a})$ (i.e., sampled from the model using the policy $\pi(\mathbf{a}|\mathbf{s})$), than they are on states from the dataset and actions from the policy, $d_{\widehat{\mathcal{M}}}(\mathbf{s})\pi(\mathbf{a}|\mathbf{s})$. COMBO's objective (Eq. 2) only penalizes Q-values under $\rho(\mathbf{s}, \mathbf{a})$, which, in practice, are expected to primarily consist of out-of-distribution states generated from model rollouts, and does not penalize the Q-value at states drawn from $d_{\widehat{\mathcal{M}}}(\mathbf{s})$. As a result, the expression $(*)$ is likely to be negative, making COMBO less conservative than CQL.

4.2 Safe Policy Improvement Guarantees

Now that we have shown various aspects of the lower-bound on the Q-function induced by COMBO, we provide policy improvement guarantees for the COMBO algorithm. Formally, Proposition 4.3 discuss safe improvement guarantees over the behavior policy. building on prior work [46, 31, 29].

Proposition 4.3 (ζ-safe policy improvement)**.** *Let $\hat{\pi}_{out}(\mathbf{a}|\mathbf{s})$ be the policy obtained by COMBO. Then, if β is sufficiently large and $\nu(\rho^{\pi}, f) - \nu(\rho^{\beta}, f) \geq C$ for a positive constant C, the policy $\hat{\pi}_{out}(\mathbf{a}|\mathbf{s})$ is a ζ-safe policy improvement over π_{β} in the actual MDP $\mathcal{M}$, i.e., $J(\hat{\pi}_{out}, \mathcal{M}) \geq J(\pi_{\beta}, \mathcal{M}) - \zeta$, with probability at least $1 - \delta$, where ζ is given by,*

$$\mathcal{O}\left(\frac{\gamma f}{(1-\gamma)^2}\right) \underbrace{\mathbb{E}_{\mathbf{s}\sim d^{\hat{\pi}_{out}}_{\widehat{\mathcal{M}}}}\left[\sqrt{\frac{|\mathcal{A}|}{|\mathcal{D}(\mathbf{s})|}\mathrm{D}_{CQL}(\hat{\pi}_{out}, \pi_{\beta})}\right]}_{:=\ (1)} + \mathcal{O}\left(\frac{\gamma(1-f)}{(1-\gamma)^2}\right) \underbrace{\mathrm{D}_{\mathrm{TV}}(\widehat{\mathcal{M}}, \overline{\mathcal{M}})}_{:=\ (2)} - \underbrace{\beta\frac{C}{(1-\gamma)}}_{:=\ (3)}.$$

The complete statement (with constants and terms that grow smaller than quadratic in the horizon) and proof for Proposition 4.3 is provided in Appendix A.4. D_{CQL} denotes a notion of probabilistic distance between policies [29] which we discuss further in Appendix A.4. The expression for ζ in Proposition 4.3 consists of three terms: term (1) captures the decrease in the policy performance due to limited data, and decays as the size of $\mathcal{D}$ increases. The second term (2) captures the suboptimality induced by the bias in the learned model. Finally, as we show in Appendix A.4, the third term (3) comes from $\nu(\rho^{\pi}, f) - \nu(\rho^{\beta}, f)$, which is equivalent to the improvement in policy performance as a result of running COMBO in the empirical and model MDPs. Since the learned model is trained on the dataset $\mathcal{D}$ with transitions generated from the behavior policy π_{β}, the marginal distribution $\rho^{\beta}(\mathbf{s}, \mathbf{a})$ is expected to be closer to $d(\mathbf{s}, \mathbf{a})$ for π_{β} as compared to the counterpart for the learned policy, ρ^{π}. Thus, the assumption that $\nu(\rho^{\pi}, f) - \nu(\rho^{\beta}, f)$ is positive is reasonable, and in such cases, an appropriate (large) choice of β will make term (3) large enough to counteract terms (1) and (2) that reduce policy performance. We discuss this elaborately in Appendix A.4 (Remark 3).

Further note that in contrast to Proposition 3.6 in Kumar et al. [29], note that our result indicates the sampling error (term (1)) is reduced (multiplied by a fraction f) when a near-accurate model is used to augment data for training the Q-function, and similarity, it can avoid the bias of model-based methods by relying more on the model-free component. This allows COMBO to attain the best-of-both model-free and model-based methods, via a suitable choice of the fraction f.

To summarize, through an appropriate choice of f, Proposition 4.3 guarantees safe improvement over the behavior policy without requiring access to an oracle uncertainty estimation algorithm.

5 Experiments

In our experiments, we aim to answer the follow questions: (1) Can COMBO generalize better than previous offline model-free and model-based approaches in a setting that requires generalization to tasks that are different from what the behavior policy solves? (2) How does COMBO compare with prior work in tasks with high-dimensional image observations? (3) How does COMBO compare to prior offline model-free and model-based methods in standard offline RL benchmarks?

To answer those questions, we compare COMBO to several prior methods. In the domains with compact state spaces, we compare with recent model-free algorithms like BEAR [28], BRAC [63], and CQL [29]; as well as MOPO [67] and MOReL [26] which are two recent model-based algorithms. In addition, we also compare with an offline version of SAC [17] (denoted as SAC-off), and behavioral cloning (BC). In high-dimensional image-based domains, which we use to answer question (3), we compare to LOMPO [48], which is a latent space offline model-based RL method that handles image inputs, latent space MBPO (denoted LMBPO), similar to Janner et al. [20] which uses the model to generate additional synthetic data, the fully offline version of SLAC [32] (denoted SLAC-off), which only uses a variational model for state representation purposes, and CQL from image inputs. To our knowledge, CQL, MOPO, and LOMPO are representative of state-of-the-art model-free and model-based offline RL methods. Hence we choose them as comparisons to COMBO. To highlight the distinction between COMBO and a naïve combination of CQL and MBPO, we perform such a comparison in Table 8 in Appendix C. For more details of our experimental set-up, comparisons, and hyperparameters, see Appendix B.

5.1 Results on tasks that require generalization

To answer question (1), we use two environments `halfcheetah-jump` and `ant-angle` constructed in Yu et al. [67], which requires the agent to solve a task that is different from what the behavior policy solved. In both environments, the offline dataset is collected by policies trained with original reward functions of `halfcheetah` and `ant`, which reward the robots to run as fast as possible. The behavior policies are trained with SAC with 1M steps and we take the full replay buffer as the offline dataset. Following Yu et al. [67], we relabel rewards in the offline datasets to reward the halfcheetah to jump as high as possible and the ant to run to the top corner with a 30 degree angle as fast as possible. Following the same manner, we construct a third task `sawyer-door-close` based on the environment in Yu et al. [66], Rafailov et al. [48]. In this task, we collect the offline data with SAC policies trained with a sparse reward function that only gives a reward of 1 when the door is *opened* by the sawyer robot and 0 otherwise. The offline dataset is similar to the "medium-expert" dataset in the D4RL benchmark since we mix equal amounts of data collected by a fully-trained SAC policy and a partially-trained SAC policy. We relabel the reward such that it is 1 when the door is *closed* and 0 otherwise. Therefore, in these datasets, the offline RL methods must generalize beyond behaviors in the offline data in order to learn the intended behaviors. We visualize the `sawyer-door-close` environment in the right image in Figure 3 in Appendix B.4.

Environment	Batch Mean	Batch Max	COMBO (Ours)	MOPO	MOReL	CQL
halfcheetah-jump	-1022.6	1808.6	**5308.7**±575.5	4016.6	3228.7	741.1
ant-angle	866.7	2311.9	**2776.9**±43.6	2530.9	2660.3	2473.4
sawyer-door-close	5%	100%	**98.3**%±3.0%	65.8%	42.9%	36.7%

Table 1: Average returns of `halfcheetah-jump` and `ant-angle` and average success rate of `sawyer-door-close` that require out-of-distribution generalization. All results are averaged over 6 random seeds. We include the mean and max return / success rate of episodes in the batch data (under Batch Mean and Batch Max, respectively) for comparison. We also include the 95%-confidence interval for COMBO.

We present the results on the three tasks in Table 1. COMBO significantly outperforms MOPO, MOReL and CQL, two representative model-based methods and one representative model-free methods respectively, in the `halfcheetah-jump` and `sawyer-door-close` tasks, and achieves an approximately 8%, 4% and 12% improvement over MOPO, MOReL and CQL respectively on the `ant-angle` task. These results validate that COMBO achieves better generalization results in practice by behaving less conservatively than prior model-free offline methods (compare to CQL, which doesn't improve much), and does so more robustly than prior model-based offline methods (compare to MOReL and MOPO).

5.1.1 Empirical analysis on uncertainty estimation in offline model-based RL

To further understand why COMBO outperforms prior model-based methods in tasks that require generalization, we argue that one of the main reasons could be that uncertainty estimation is hard in these tasks where the agent is required to go further away from the data distribution. To test this intuition, we perform empirical evaluations to study whether uncertainty quantification with deep neural networks, especially in the setting of dynamics model learning, is challenging and could cause problems with uncertainty-based model-based offline RL methods such as MOReL [26] and MOPO [67]. In our evaluations, we consider maximum learned variance over the ensemble (denoted as **Max Var**) $\max_{i=1,\ldots,N} \|\Sigma_\theta^i(\mathbf{s},\mathbf{a})\|_{\mathrm{F}}$ (used in MOPO).

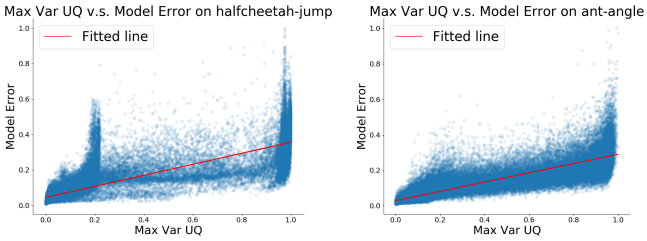

Figure 2: We visualize the fitted linear regression line between the model error and two uncertainty quantification methods maximum learned variance over the ensemble (denoted as **Max Var**) on two tasks that test the generalization abilities of offline RL algorithms (`halfcheetah-jump` and `ant-angle`). We show that **Max Var** struggles to predict the true model error. Such visualizations indicates that uncertainty quantification is challenging with deep neural networks and could lead to poor performance in model-based offline RL in settings where out-of-distribution generalization is needed. In the meantime, COMBO addresses this issue by removing the burden of performing uncertainty quantification.

We consider two tasks `halfcheetah-jump` and `ant-angle`. We normalize both the model error and the uncertainty estimates to be within scale $[0, 1]$ and performs linear regression that learns the mapping between the uncertainty estimates and the true model error. As shown in Figure 2, on both tasks, **Max Var** is unable to accurately predict the true model error, suggesting that uncertainty estimation used by offline model-based methods is not accurate and might be the major factor that results in its poor performance. Meanwhile, COMBO circumvents challenging uncertainty quantification problem and achieves better performances on those tasks, indicating the effectiveness and the robustness of the method.

5.2 Results on image-based tasks

To answer question (2), we evaluate COMBO on two image-based environments: the standard walker (`walker-walk`) task from the the DeepMind Control suite [61] and a visual door opening environment with a Sawyer robotic arm (`sawyer-door`) as used in Section 5.1. For the walker task we construct 4 datasets: medium-replay (M-R), medium (M), medium-expert (M-E), and expert, similar to Fu et al. [12], each consisting of 200 trajectories. For `sawyer-door` task we use only the medium-expert and the expert datasets, due to the sparse reward – the agent is rewarded only when it successfully opens the door. Both environments are visulized in Figure 3 in Appendix B.4. To extend COMBO to the image-based setting, we follow Rafailov et al. [48] and train a recurrent variational model using the offline data and use train COMBO in the latent space of this model. We present

Dataset	Environment	COMBO (Ours)	LOMPO	LMBPO	SLAC -Off	CQL
M-R	walker_walk	**69.2**	66.9	59.8	45.1	15.6
M	walker_walk	57.7	60.2	**61.7**	41.5	38.9
M-E	walker_walk	76.4	**78.9**	47.3	34.9	36.3
expert	walker_walk	**61.1**	55.6	13.2	12.6	43.3
M-E	sawyer-door	**100.0%**	**100.0%**	0.0%	0.0%	0.0%
expert	sawyer-door	**96.7%**	0.0%	0.0%	0.0%	0.0%

Table 2: Results for vision experiments. For the Walker task each number is the normalized score proposed in [12] of the policy at the last iteration of training, averaged over 3 random seeds. For the Sawyer task, we report success rates over the last 100 evaluation runs of training. For the dataset, M refers to medium, M-R refers to medium-replay, and M-E refers to medium expert.

Dataset type	Environment	BC	COMBO (Ours)	MOPO	MOReL	CQL	SAC-off	BEAR	BRAC-p	BRAC-v
random	halfcheetah	2.1	**38.8**±3.7	35.4	25.6	35.4	30.5	25.1	24.1	31.2
random	hopper	1.6	17.9±1.4	11.7	**53.6**	10.8	11.3	11.4	11.0	12.2
random	walker2d	9.8	7.0±3.6	13.6	**37.3**	7.0	4.1	7.3	-0.2	1.9
medium	halfcheetah	36.1	**54.2**±1.5	42.3	42.1	44.4	-4.3	41.7	43.8	46.3
medium	hopper	29.0	**97.2**±2.2	28.0	95.4	86.6	0.8	52.1	32.7	31.1
medium	walker2d	6.6	**81.9**±2.8	17.8	77.8	74.5	0.9	59.1	77.5	81.1
medium-replay	halfcheetah	38.4	**55.1**±1.0	53.1	40.2	46.2	-2.4	38.6	45.4	47.7
medium-replay	hopper	11.8	89.5±1.8	67.5	**93.6**	48.6	3.5	33.7	0.6	0.6
medium-replay	walker2d	11.3	**56.0**±8.6	39.0	49.8	32.6	1.9	19.2	-0.3	0.9
med-expert	halfcheetah	35.8	**90.0**±5.6	63.3	53.3	62.4	1.8	53.4	44.2	41.9
med-expert	hopper	111.9	**111.1**±2.9	23.7	108.7	111.0	1.6	96.3	1.9	0.8
med-expert	walker2d	6.4	**103.3**±5.6	44.6	95.6	98.7	-0.1	40.1	76.9	81.6

Table 3: Results for D4RL datasets. Each number is the normalized score proposed in [12] of the policy at the last iteration of training, averaged over 6 random seeds. We take results of MOPO, MOReL and CQL from their original papers and results of other model-free methods from [12]. We include the performance of behavior cloning (**BC**) for comparison. We include the 95%-confidence interval for COMBO. We bold the highest score across all methods.

results in Table 2. On the `walker-walk` task, COMBO performs in line with LOMPO and previous methods. On the more challenging Sawyer task, COMBO matches LOMPO and achieves 100% success rate on the medium-expert dataset, and substantially outperforms all other methods on the narrow expert dataset, achieving an average success rate of 96.7%, when all other model-based and model-free methods fail.

5.3 Results on the D4RL tasks

Finally, to answer the question (3), we evaluate COMBO on the OpenAI Gym [6] domains in the D4RL benchmark [12], which contains three environments (halfcheetah, hopper, and walker2d) and four dataset types (random, medium, medium-replay, and medium-expert). We include the results in Table 3. The numbers of BC, SAC-off, BEAR, BRAC-P and BRAC-v are taken from the D4RL paper, while the results for MOPO, MOReL and CQL are based on their respective papers [67, 29]. COMBO achieves the best performance in 9 out of 12 settings and comparable result in 1 out of the remaining 3 settings (hopper medium-replay). As noted by Yu et al. [67] and Rafailov et al. [48], model-based offline methods are generally more performant on datasets that are collected by a wide range of policies and have diverse state-action distributions (random, medium-replay datasets) while model-free approaches do better on datasets with narrow distributions (medium, medium-expert datasets). However, in these results, COMBO generally performs well across dataset types compared to existing model-free and model-based approaches, suggesting that COMBO is robust to different dataset types.

6 Related Work

Offline RL [10, 50, 30, 34] is the task of learning policies from a static dataset of past interactions with the environment. It has found applications in domains including robotic manipulation [25, 38, 48, 54], NLP [21, 22] and healthcare [52, 62]. Similar to interactive RL, both model-free and model-based algorithms have been studied for offline RL, with explicit or implicit regularization of the learning algorithm playing a major role.

Model-free offline RL. Prior model-free offline RL algorithms have been designed to regularize the learned policy to be "close" to the behavioral policy either implicitly via regularized variants of importance sampling based algorithms [47, 58, 35, 59, 41], offline actor-critic methods [53, 45, 27, 16, 64], applying uncertainty quantification to the predictions of the Q-values [2, 28, 63, 34], and learning conservative Q-values [29, 55] or explicitly measured by direct state or action constraints [14, 36], KL divergence [21, 63, 69], Wasserstein distance, MMD [28] and auxiliary imitation loss [13]. Different from these works, COMBO uses both the offline dataset as well as model-generated data.

Model-based offline RL. Model-based offline RL methods [11, 9, 24, 26, 67, 39, 3, 60, 48, 33, 68] provide an alternative approach to policy learning that involves the learning of a dynamics model using techniques from supervised learning and generative modeling. Such methods however rely either on uncertainty quantification of the learned dynamics model which can be difficult for deep network models [44], or on directly constraining the policy towards the behavioral policy similar to model-free algorithms [39]. In contrast, COMBO conservatively estimates the value function by penalizing it in out-of-support states generated through model rollouts. This allows COMBO to

retain all benefits of model-based algorithms such as broad generalization, without the constraints of explicit policy regularization or uncertainty quantification.

7 Conclusion

In the paper, we present conservative offline model-based policy optimization (COMBO), a model-based offline RL algorithm that penalizes the Q-values evaluated on out-of-support state-action pairs. In particular, COMBO removes the need of uncertainty quantification as widely used in previous model-based offline RL works [26, 67], which can be challenging and unreliable with deep neural networks [44]. Theoretically, we show that COMBO achieves less conservative Q values compared to prior model-free offline RL methods [29] and guarantees a safe policy improvement. In our empirical study, COMBO achieves the best generalization performances in 3 tasks that require adaptation to unseen behaviors. Moreover, COMBO is able scale to vision-based tasks and outperforms or obtain comparable results in vision-based locomotion and robotic manipulation tasks. Finlly, on standard D4RL benchmark, COMBO generally performs well across dataset types compared to prior methods Despite the advantages of COMBO, there are few challenges left such as the lack of an offline hyperparameter selection scheme that can yield a uniform hyperparameter across different datasets and an automatically selected f conditioned on the model error. We leave them for future work.

Acknowledgments and Disclosure of Funding

We thank members of RAIL and IRIS for their support and feedback. This work was supported in part by ONR grants N00014-20-1-2675 and N00014-21-1-2685 as well as Intel Corporation. AK and SL are supported by the DARPA Assured Autonomy program. AR was supported by the J.P. Morgan PhD Fellowship in AI.

References

[1] Alekh Agarwal, Nan Jiang, and Sham M Kakade. Reinforcement learning: Theory and algorithms. *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep*, 2019.

[2] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pages 104–114. PMLR, 2020.

[3] Arthur Argenson and Gabriel Dulac-Arnold. Model-based offline planning. *arXiv preprint arXiv:2008.05556*, 2020.

[4] Kamyar Azizzadenesheli, Emma Brunskill, and Animashree Anandkumar. Efficient exploration through bayesian deep q-networks. In *ITA*, pages 1–9. IEEE, 2018.

[5] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

[6] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[7] Ignasi Clavera, Violet Fu, and Pieter Abbeel. Model-augmented actor-critic: Backpropagating through paths. *arXiv preprint arXiv:2005.08068*, 2020.

[8] Thomas Degris, Martha White, and Richard S Sutton. Off-policy actor-critic. *arXiv preprint arXiv:1205.4839*, 2012.

[9] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

[10] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

[11] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE, 2017.

[12] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

[13] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *arXiv preprint arXiv:2106.06860*, 2021.

[14] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. *arXiv preprint arXiv:1812.02900*, 2018.

[15] Scott Fujimoto, Herke Van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

[16] Seyed Kamyar Seyed Ghasemipour, Dale Schuurmans, and Shixiang Shane Gu. Emaq: Expected-max q-learning operator for simple yet effective offline and online rl. In *International Conference on Machine Learning*, pages 3682–3691. PMLR, 2021.

[17] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[18] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. International conference on machine learning. In *International Conference on Machine Learning*, 2019.

[19] Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(4), 2010.

[20] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, pages 12498–12509, 2019.

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

[22] Natasha Jaques, Judy Hanwen Shen, Asma Ghandeharioun, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Shane Gu, and Rosalind Picard. Human-centric dialog training via offline reinforcement learning. *arXiv preprint arXiv:2010.05848*, 2020.

[23] Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline rl? In *International Conference on Machine Learning*, pages 5084–5096. PMLR, 2021.

[24] Gregory Kahn, Adam Villaflor, Pieter Abbeel, and Sergey Levine. Composable action-conditioned predictors: Flexible off-policy learning for robot navigation. In *Conference on Robot Learning*, pages 806–816. PMLR, 2018.

[25] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pages 651–673. PMLR, 2018.

[26] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *arXiv preprint arXiv:2005.05951*, 2020.

[27] Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. In *International Conference on Machine Learning*, pages 5774–5783. PMLR, 2021.

[28] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, pages 11761–11771, 2019.

[29] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

[30] Sascha Lange, Thomas Gabel, and Martin A. Riedmiller. Batch reinforcement learning. In *Reinforcement Learning*, volume 12. Springer, 2012.

[31] Romain Laroche, Paul Trichelair, and Remi Tachet Des Combes. Safe policy improvement with baseline bootstrapping. In *International Conference on Machine Learning*, pages 3652–3661. PMLR, 2019.

[32] Alex X. Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. In *Advances in Neural Information Processing Systems*, 2020.

[33] Byung-Jun Lee, Jongmin Lee, and Kee-Eung Kim. Representation balancing offline model-based reinforcement learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=QpNz8r_Ri2Y.

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

[35] Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Off-policy policy gradient with state distribution correction. *CoRR*, abs/1904.08473, 2019.

[36] Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Provably good batch reinforcement learning without great exploration. *arXiv preprint arXiv:2007.08202*, 2020.

[37] Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. Plan Online, Learn Offline: Efficient Learning and Exploration via Model-Based Control. In *International Conference on Learning Representations (ICLR)*, 2019.

[38] Ajay Mandlekar, Fabio Ramos, Byron Boots, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Dieter Fox. Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4414–4420. IEEE, 2020.

[39] Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. Deployment-efficient reinforcement learning via model-based offline optimization. *arXiv preprint arXiv:2006.03647*, 2020.

[40] Rémi Munos and Csaba Szepesvari. Finite-time bounds for fitted value iteration. *J. Mach. Learn. Res.*, 9:815–857, 2008.

[41] Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. Algaedice: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*, 2019.

[42] Ian Osband and Benjamin Van Roy. Why is posterior sampling better than optimism for reinforcement learning? In *International Conference on Machine Learning*, pages 2701–2710. PMLR, 2017.

[43] Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. *CoRR*, abs/1806.03335, 2018.

[44] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua V Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *arXiv preprint arXiv:1906.02530*, 2019.

[45] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

[46] Marek Petrik, Yinlam Chow, and Mohammad Ghavamzadeh. Safe policy improvement by minimizing robust baseline regret. *arXiv preprint arXiv:1607.03842*, 2016.

[47] Doina Precup, Richard S Sutton, and Sanjoy Dasgupta. Off-policy temporal-difference learning with function approximation. In *ICML*, pages 417–424, 2001.

[48] Rafael Rafailov, Tianhe Yu, A. Rajeswaran, and Chelsea Finn. Offline reinforcement learning from images with latent space models. *ArXiv*, abs/2012.11547, 2020.

[49] Paria Rashidinejad, Banghua Zhu, Cong Ma, Jiantao Jiao, and Stuart Russell. Bridging offline reinforcement learning and imitation learning: A tale of pessimism. *arXiv preprint arXiv:2103.12021*, 2021.

[50] Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pages 317–328. Springer, 2005.

[51] Stephane Ross and Drew Bagnell. Agnostic system identification for model-based reinforcement learning. In *ICML*, 2012.

[52] Susan M Shortreed, Eric Laber, Daniel J Lizotte, T Scott Stroup, Joelle Pineau, and Susan A Murphy. Informing sequential clinical decision-making through reinforcement learning: an empirical study. *Machine learning*, 84(1-2):109–136, 2011.

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

[54] Avi Singh, Albert Yu, Jonathan Yang, Jesse Zhang, Aviral Kumar, and Sergey Levine. Cog: Connecting new skills to past experience with offline reinforcement learning. *arXiv preprint arXiv:2010.14500*, 2020.

[55] Samarth Sinha and Animesh Garg. S4rl: Surprisingly simple self-supervision for offline reinforcement learning. *arXiv preprint arXiv:2103.06326*, 2021.

[56] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[57] Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

[58] Richard S Sutton, A Rupam Mahmood, and Martha White. An emphatic approach to the problem of off-policy temporal-difference learning. *The Journal of Machine Learning Research*, 17(1):2603–2631, 2016.

[59] Adith Swaminathan and Thorsten Joachims. Batch learning from logged bandit feedback through counterfactual risk minimization. *J. Mach. Learn. Res*, 16:1731–1755, 2015.

[60] Phillip Swazinna, Steffen Udluft, and Thomas Runkler. Overcoming model bias for robust offline deep reinforcement learning. *arXiv preprint arXiv:2008.05533*, 2020.

[61] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[62] L. Wang, Wei Zhang, Xiaofeng He, and H. Zha. Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

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

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

[65] F. Yu, H. Chen, X. Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, V. Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2633–2642, 2020.

[66] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pages 1094–1100. PMLR, 2020.

[67] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *arXiv preprint arXiv:2005.13239*, 2020.

[68] Xianyuan Zhan, Xiangyu Zhu, and Haoran Xu. Model-based offline planning with trajectory pruning. *arXiv preprint arXiv:2105.07351*, 2021.

[69] Wenxuan Zhou, Sujay Bajracharya, and David Held. Plas: Latent action space for offline reinforcement learning. *arXiv preprint arXiv:2011.07213*, 2020.

A Proofs from Section 4

In this section, we provide proofs for theoretical results in Section 4. Before the proofs, we note that all statements are proven in the case of finite state space (i.e., $|\mathcal{S}| < \infty$) and finite action space (i.e., $|\mathcal{A}| < \infty$) we define some commonly appearing notation symbols appearing in the proof:

- $P_{\mathcal{M}}$ and $r_{\mathcal{M}}$ (or P and r with no subscript for notational simplicity) denote the dynamics and reward function of the actual MDP $\mathcal{M}$
- $P_{\overline{\mathcal{M}}}$ and $r_{\overline{\mathcal{M}}}$ denote the dynamics and reward of the empirical MDP $\overline{\mathcal{M}}$ generated from the transitions in the dataset
- $P_{\widehat{\mathcal{M}}}$ and $r_{\widehat{\mathcal{M}}}$ denote the dynamics and reward of the MDP induced by the learned model $\widehat{\mathcal{M}}$

We also assume that whenever the cardinality of a particular state or state-action pair in the offline dataset $\mathcal{D}$, denoted by $|\mathcal{D}(\mathbf{s}, \mathbf{a})|$, appears in the denominator, we assume it is non-zero. For any non-existent $(\mathbf{s}, \mathbf{a}) \notin \mathcal{D}$, we can simply set $|\mathcal{D}(\mathbf{s}, \mathbf{a})|$ to be a small value < 1, which prevents any bound from producing trivially ∞ values.

A.1 A Useful Lemma and Its Proof

Before proving our main results, we first show that the penalty term in equation 4 is positive in expectation. Such a positive penalty is important to combat any overestimation that may arise as a result of using $\widehat{\mathcal{B}}$.

Lemma A.1 ((Interpolation Lemma)**.** *For any $f \in [0, 1]$, and any given $\rho(\mathbf{s}, \mathbf{a}) \in \Delta^{|\mathcal{S}||\mathcal{A}|}$, let d_f be an f-interpolation of ρ and $\mathcal{D}$, i.e., $d_f(\mathbf{s}, \mathbf{a}) := fd(\mathbf{s}, \mathbf{a}) + (1 - f)\rho(\mathbf{s}, \mathbf{a})$. For a given iteration k of Equation 4, we restate the definition of the expected penalty under $\rho(\mathbf{s}, \mathbf{a})$ in Eq. 5:*

$$\nu(\rho, f) := \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\rho(\mathbf{s},\mathbf{a})}\left[\frac{\rho(\mathbf{s}, \mathbf{a}) - d(\mathbf{s}, \mathbf{a})}{d_f(\mathbf{s}, \mathbf{a})}\right].$$

Then $\nu(\rho, f)$ satisfies, (1) $\nu(\rho, f) \geq 0, \ \ \forall \rho, f$, (2) $\nu(\rho, f)$ is monotonically increasing in f for a fixed ρ, and (3) $\nu(\rho, f) = 0$ iff $\forall\, \mathbf{s}, \mathbf{a}, \ \rho(\mathbf{s}, \mathbf{a}) = d(\mathbf{s}, \mathbf{a})$ or $f = 0$.

Proof. To prove this lemma, we use algebraic manipulation on the expression for quantity $\nu(\rho, f)$ and show that it is indeed positive and monotonically increasing in $f \in [0, 1]$.

$$\begin{aligned}
\nu(\rho, f) &= \sum_{\mathbf{s},\mathbf{a}} \rho(\mathbf{s}, \mathbf{a}) \left(\frac{\rho(\mathbf{s}, \mathbf{a}) - d(\mathbf{s}, \mathbf{a})}{fd(\mathbf{s}, \mathbf{a}) + (1 - f)\rho(\mathbf{s}, \mathbf{a})}\right) \\
&= \sum_{\mathbf{s},\mathbf{a}} \rho(\mathbf{s}, \mathbf{a}) \left(\frac{\rho(\mathbf{s}, \mathbf{a}) - d(\mathbf{s}, \mathbf{a})}{\rho(\mathbf{s}, \mathbf{a}) + f(d(\mathbf{s}, \mathbf{a}) - \rho(\mathbf{s}, \mathbf{a}))}\right) \qquad (6) \\
\implies \frac{d\nu(\rho, f)}{df} &= \sum_{\mathbf{s},\mathbf{a}} \rho(\mathbf{s}, \mathbf{a}) \left(\rho(\mathbf{s}, \mathbf{a}) - d(\mathbf{s}, \mathbf{a})\right)^2 \cdot \left(\frac{1}{(\rho(\mathbf{s}, \mathbf{a}) + f(d(\mathbf{s}, \mathbf{a}) - \rho(\mathbf{s}, \mathbf{a}))}\right)^2 \geq 0 \\
&\qquad \forall f \in [0, 1]. \qquad (7)
\end{aligned}$$

Since the derivative of $\nu(\rho, f)$ with respect to f is always positive, it is an increasing function of f for a fixed ρ, and this proves the second part (2) of the Lemma. Using this property, we can show the part (1) of the Lemma as follows:

$$\begin{aligned}
\forall f \in (0, 1], \ \nu(\rho, f) \geq \nu(\rho, 0) = \sum_{\mathbf{s},\mathbf{a}} \rho(\mathbf{s}, \mathbf{a}) \frac{\rho(\mathbf{s}, \mathbf{a}) - d(\mathbf{s}, \mathbf{a})}{\rho(\mathbf{s}, \mathbf{a})} &= \sum_{\mathbf{s},\mathbf{a}} \left(\rho(\mathbf{s}, \mathbf{a}) - d(\mathbf{s}, \mathbf{a})\right) \\
&= 1 - 1 = 0. \qquad (8)
\end{aligned}$$

Finally, to prove the third part (3) of this Lemma, note that when $f = 0$, $\nu(\rho, f) = 0$ (as shown above), and similarly by setting $\rho(\mathbf{s}, \mathbf{a}) = d(\mathbf{s}, \mathbf{a})$ note that we obtain $\nu(\rho, f) = 0$. To prove the only if side of (3), assume that $f \neq 0$ and $\rho(\mathbf{s}, \mathbf{a}) \neq d(\mathbf{s}, \mathbf{a})$ and we will show that in this case $\nu(\rho, f) \neq 0$. When $d(\mathbf{s}, \mathbf{a}) \neq \rho(\mathbf{s}, \mathbf{a})$, the derivative $\frac{d\nu(\rho,f)}{df} > 0$ (i.e., strictly positive) and hence the function $\nu(\rho, f)$ is a strictly increasing function of f. Thus, in this case, $\nu(\rho, f) > 0 = \nu(\rho, 0)\ \forall f > 0$. Thus we have shown that if $\rho(\mathbf{s}, \mathbf{a}) \neq d(\mathbf{s}, \mathbf{a})$ and $f > 0$, $\nu(\rho, f) \neq 0$, which completes our proof for the only if side of (3). □

A.2 Proof of Proposition 4.1

Before proving this proposition, we provide a bound on the Bellman backup in the empirical MDP, $\mathcal{B}_{\overline{\mathcal{M}}}$. To do so, we formally define the standard concentration properties of the reward and transition dynamics in the empirical MDP, $\overline{\mathcal{M}}$, that we assume so as to prove Proposition A.1. Following prior work [42, 19, 29], we assume:

Assumption A1. $\forall\, \mathbf{s}, \mathbf{a} \in \mathcal{M}$, *the following relationships hold with high probability,* $\geq 1 - \delta$

$$|r_{\overline{\mathcal{M}}}(\mathbf{s}, \mathbf{a}) - r(\mathbf{s}, \mathbf{a})| \leq \frac{C_{r,\delta}}{\sqrt{|\mathcal{D}(\mathbf{s}, \mathbf{a})|}}, \quad ||P_{\overline{\mathcal{M}}}(\mathbf{s}'|\mathbf{s}, \mathbf{a}) - P(\mathbf{s}'|\mathbf{s}, \mathbf{a})||_1 \leq \frac{C_{P,\delta}}{\sqrt{|\mathcal{D}(\mathbf{s}, \mathbf{a})|}}.$$

Under this assumption and assuming that the reward function in the MDP, $r(\mathbf{s}, \mathbf{a})$ is bounded, as $|r(\mathbf{s}, \mathbf{a})| \leq R_{\max}$, we can bound the difference between the empirical Bellman operator, $\mathcal{B}_{\overline{\mathcal{M}}}$ and the actual MDP, $\mathcal{B}_{\mathcal{M}}$,

$$\begin{aligned}
\left|\left(\mathcal{B}_{\overline{\mathcal{M}}}{}^{\pi}\hat{Q}^k\right) - \left(\mathcal{B}^{\pi}_{\mathcal{M}}\hat{Q}^k\right)\right| &= |(r_{\overline{\mathcal{M}}}(\mathbf{s}, \mathbf{a}) - r_{\mathcal{M}}(\mathbf{s}, \mathbf{a})) \\
&\quad + \gamma \sum_{\mathbf{s}'} \left(P_{\overline{\mathcal{M}}}(\mathbf{s}'|\mathbf{s}, \mathbf{a}) - P_{\mathcal{M}}(\mathbf{s}'|\mathbf{s}, \mathbf{a})\right) \mathbb{E}_{\pi(\mathbf{a}'|\mathbf{s}')}\left[\hat{Q}^k(\mathbf{s}', \mathbf{a}')\right]\Big| \\
&\leq |r_{\overline{\mathcal{M}}}(\mathbf{s}, \mathbf{a}) - r_{\mathcal{M}}(\mathbf{s}, \mathbf{a})| \\
&\quad + \gamma \left|\sum_{\mathbf{s}'} \left(P_{\overline{\mathcal{M}}}(\mathbf{s}'|\mathbf{s}, \mathbf{a}) - P_{\mathcal{M}}(\mathbf{s}'|\mathbf{s}, \mathbf{a})\right) \mathbb{E}_{\pi(\mathbf{a}'|\mathbf{s}')}\left[\hat{Q}^k(\mathbf{s}', \mathbf{a}')\right]\right| \\
&\leq \frac{C_{r,\delta} + \gamma C_{P,\delta} 2R_{\max}/(1-\gamma)}{\sqrt{|\mathcal{D}(\mathbf{s}, \mathbf{a})|}}.
\end{aligned}$$

Thus the overestimation due to sampling error in the empirical MDP, $\overline{\mathcal{M}}$ is bounded as a function of a bigger constant, $C_{r,P,\delta}$ that can be expressed as a function of $C_{r,\delta}$ and $C_{P,\delta}$, and depends on δ via a $\sqrt{\log(1/\delta)}$ dependency. For the purposes of proving Proposition A.1, we assume that:

$$\forall \mathbf{s}, \mathbf{a}, \quad \left|\left(\mathcal{B}_{\overline{\mathcal{M}}}{}^{\pi}\hat{Q}^k\right) - \left(\mathcal{B}^{\pi}_{\mathcal{M}}\hat{Q}^k\right)\right| \leq \frac{C_{r,T,\delta} R_{\max}}{(1-\gamma)\sqrt{|\mathcal{D}(\mathbf{s}, \mathbf{a})|}}. \tag{9}$$

Next, we provide a bound on the error between the bellman backup induced by the learned dynamics model and the learned reward, $\mathcal{B}_{\widehat{\mathcal{M}}}$, and the actual Bellman backup, $\mathcal{B}_{\mathcal{M}}$. To do so, we note that:

$$\begin{aligned}
\left|\left(\mathcal{B}_{\widehat{\mathcal{M}}}{}^{\pi}\hat{Q}^k\right) - \left(\mathcal{B}^{\pi}_{\mathcal{M}}\hat{Q}^k\right)\right| &= \left|\left(r_{\widehat{\mathcal{M}}}(\mathbf{s}, \mathbf{a}) - r_{\mathcal{M}}(\mathbf{s}, \mathbf{a})\right)\right. \qquad (10) \\
&\quad \left. + \gamma \sum_{\mathbf{s}'} \left(P_{\widehat{\mathcal{M}}}(\mathbf{s}'|\mathbf{s}, \mathbf{a}) - P_{\mathcal{M}}(\mathbf{s}'|\mathbf{s}, \mathbf{a})\right) \mathbb{E}_{\pi(\mathbf{a}'|\mathbf{s}')}\left[\hat{Q}^k(\mathbf{s}', \mathbf{a}')\right]\right| \\
&\leq |r_{\widehat{\mathcal{M}}}(\mathbf{s}, \mathbf{a}) - r_{\mathcal{M}}(\mathbf{s}, \mathbf{a})| + \gamma \frac{2R_{\max}}{1-\gamma} D(P, P_{\widehat{\mathcal{M}}}), \qquad (11)
\end{aligned}$$

where $D(P, P_{\widehat{\mathcal{M}}})$ is the total-variation divergence between the learned dynamics model and the actual MDP. Now, we show that the asymptotic Q-function learned by COMBO lower-bounds the actual Q-function of any policy π with high probability for a large enough $\beta \geq 0$. We will use Equations 9 and 11 to prove such a result.

Proposition A.1 (Asymptotic lower-bound)**.** *Let P^{π} denote the Hadamard product of the dynamics P and a given policy π in the actual MDP and let $S^{\pi} := (I - \gamma P^{\pi})^{-1}$. Let D denote the total-variation divergence between two probability distributions. For any $\pi(\mathbf{a}|\mathbf{s})$, the Q-function obtained by recursively applying Equation 4, with $\hat{\mathcal{B}}^{\pi} = f\mathcal{B}^{\pi}_{\overline{\mathcal{M}}} + (1-f)\mathcal{B}^{\pi}_{\widehat{\mathcal{M}}}$, with probability at least $1 - \delta$, results in $\hat{Q}^{\pi}$ that satisfies:*

$$\begin{aligned}
\forall \mathbf{s}, \mathbf{a},\ \hat{Q}^{\pi}(\mathbf{s}, \mathbf{a}) \leq Q^{\pi}(\mathbf{s}, \mathbf{a}) - \beta \cdot \left[S^{\pi}\left[\frac{\rho - d}{d_f}\right]\right](\mathbf{s}, \mathbf{a}) + f\left[S^{\pi}\left[\frac{C_{r,T,\delta} R_{\max}}{(1-\gamma)\sqrt{|\mathcal{D}|}}\right]\right](\mathbf{s}, \mathbf{a}) \\
+ (1-f)\left[S^{\pi}\left[|r - r_{\widehat{\mathcal{M}}}| + \frac{2\gamma R_{\max}}{1-\gamma} D(P, P_{\widehat{\mathcal{M}}})\right]\right](\mathbf{s}, \mathbf{a}).
\end{aligned}$$

Proof. We first note that the Bellman backup $\hat{\mathcal{B}}^\pi$ induces the following Q-function iterates as per Equation 4,

$$\begin{aligned}
\hat{Q}^{k+1}(\mathbf{s},\mathbf{a}) &= \left(\hat{\mathcal{B}}^\pi \hat{Q}^k\right)(\mathbf{s},\mathbf{a}) - \beta \frac{\rho(\mathbf{s},\mathbf{a}) - d(\mathbf{s},\mathbf{a})}{d_f(\mathbf{s},\mathbf{a})} \\
&= f\left(\mathcal{B}^\pi_{\mathcal{M}} \hat{Q}^k\right)(\mathbf{s},\mathbf{a}) + (1-f)\left(\mathcal{B}^\pi_{\widehat{\mathcal{M}}} \hat{Q}^k\right)(\mathbf{s},\mathbf{a}) - \beta \frac{\rho(\mathbf{s},\mathbf{a}) - d(\mathbf{s},\mathbf{a})}{d_f(\mathbf{s},\mathbf{a})} \\
&= \left(\mathcal{B}^\pi \hat{Q}^k\right)(\mathbf{s},\mathbf{a}) - \beta \frac{\rho(\mathbf{s},\mathbf{a}) - d(\mathbf{s},\mathbf{a})}{d_f(\mathbf{s},\mathbf{a})} + (1-f)\left(\mathcal{B}_{\widehat{\mathcal{M}}}{}^\pi \hat{Q}^k - \mathcal{B}^\pi \hat{Q}^k\right)(\mathbf{s},\mathbf{a}) \\
&+ f\left(\mathcal{B}_{\overline{\mathcal{M}}}{}^\pi \hat{Q}^k - \mathcal{B}^\pi \hat{Q}^k\right)(\mathbf{s},\mathbf{a}) \\
\forall \mathbf{s},\mathbf{a},\ \hat{Q}^{k+1} &\leq \left(\mathcal{B}^\pi \hat{Q}^k\right) - \beta \frac{\rho - d}{d_f} + (1-f)\left[|r_{\widehat{\mathcal{M}}} - r_{\mathcal{M}}| + \frac{2\gamma R_{\max}}{1-\gamma} D(P, P_{\widehat{\mathcal{M}}})\right] + f \frac{C_{r,T,\delta} R_{\max}}{(1-\gamma)\sqrt{|\mathcal{D}|}}
\end{aligned}$$

Since the RHS upper bounds the Q-function pointwise for each $(\mathbf{s},\mathbf{a})$, the fixed point of the Bellman iteration process will be pointwise smaller than the fixed point of the Q-function found by solving for the RHS via equality. Thus, we get that

$$\begin{aligned}
\hat{Q}^\pi(\mathbf{s},\mathbf{a}) \leq \underbrace{S^\pi r_{\mathcal{M}}}_{=Q^\pi(\mathbf{s},\mathbf{a})} - \beta\left[S^\pi\left[\frac{\rho - d}{d_f}\right]\right](\mathbf{s},\mathbf{a}) + f\left[S^\pi\left[\frac{C_{r,T,\delta} R_{\max}}{(1-\gamma)\sqrt{|\mathcal{D}|}}\right]\right](\mathbf{s},\mathbf{a}) \\
+ (1-f)\left[S^\pi\left[|r - r_{\widehat{\mathcal{M}}}| + \frac{2\gamma R_{\max}}{1-\gamma} D(P, P_{\widehat{\mathcal{M}}})\right]\right](\mathbf{s},\mathbf{a}),
\end{aligned}$$

which completes the proof of this proposition. □

Next, we use the result and proof technique from Proposition A.1 to prove Corollary 4.1, that in expectation under the initial state-distribution, the expected Q-value is indeed a lower-bound.

Corollary A.1 (Corollary 4.1 restated)**.** *For a sufficiently large β, we have a lower-bound that $\mathbb{E}_{\mathbf{s}\sim\mu_0,\mathbf{a}\sim\pi(\cdot|\mathbf{s})}[\hat{Q}^\pi(\mathbf{s},\mathbf{a})] \leq \mathbb{E}_{\mathbf{s}\sim\mu_0,\mathbf{a}\sim\pi(\cdot|\mathbf{s})}[Q^\pi(\mathbf{s},\mathbf{a})]$, where $\mu_0(\mathbf{s})$ is the initial state distribution. Furthermore, when ϵ_s is small, such as in the large sample regime; or when the model bias ϵ_m is small, a small β is sufficient along with an appropriate choice of f.*

Proof. To prove this corollary, we note a slightly different variant of Proposition A.1. To observe this, we will deviate from the proof of Proposition A.1 slightly and will aim to express the inequality using $\mathcal{B}_{\widehat{\mathcal{M}}}$, the Bellman operator defined by the learned model and the reward function. Denoting $(I - \gamma P_{\widehat{\mathcal{M}}})^{-1}$ as $S^\pi_{\widehat{\mathcal{M}}}$, doing this will intuitively allow us to obtain $\beta\left(\mu(\mathbf{s})\pi(\mathbf{a}|\mathbf{s})\right)^T\left(S^\pi_{\widehat{\mathcal{M}}}\left[\frac{\rho-d}{d_f}\right]\right)(\mathbf{s},\mathbf{a})$ as the conservative penalty which can be controlled by choosing β appropriately so as to nullify the potential overestimation caused due to other terms. Formally,

$$\begin{aligned}
\hat{Q}^{k+1}(\mathbf{s},\mathbf{a}) &= \left(\hat{\mathcal{B}}^\pi \hat{Q}^k\right)(\mathbf{s},\mathbf{a}) - \beta \frac{\rho(\mathbf{s},\mathbf{a}) - d(\mathbf{s},\mathbf{a})}{d_f(\mathbf{s},\mathbf{a})} = \left(\mathcal{B}^\pi_{\widehat{\mathcal{M}}} \hat{Q}^k\right)(\mathbf{s},\mathbf{a}) - \beta \frac{\rho(\mathbf{s},\mathbf{a}) - d(\mathbf{s},\mathbf{a})}{d_f(\mathbf{s},\mathbf{a})} \\
&+ f \underbrace{\left(\mathcal{B}^\pi_{\mathcal{M}} - \mathcal{B}^\pi_{\widehat{\mathcal{M}}} \hat{Q}^k\right)(\mathbf{s},\mathbf{a})}_{:=\Delta(\mathbf{s},\mathbf{a})}
\end{aligned}$$

By controlling $\Delta(\mathbf{s},\mathbf{a})$ using the pointwise triangle inequality:

$$\forall \mathbf{s},\mathbf{a},\ \left|\mathcal{B}^\pi_{\mathcal{M}} \hat{Q}^k - \mathcal{B}^\pi_{\widehat{\mathcal{M}}} \hat{Q}^k\right| \leq \left|\mathcal{B}^\pi \hat{Q}^k - \mathcal{B}^\pi_{\widehat{\mathcal{M}}} \hat{Q}^k\right| + \left|\mathcal{B}^\pi_{\mathcal{M}} \hat{Q}^k - \mathcal{B}^\pi \hat{Q}^k\right|, \tag{12}$$

and then iterating the backup $\mathcal{B}^\pi_{\widehat{\mathcal{M}}}$ to its fixed point and finally noting that $\rho(\mathbf{s},\mathbf{a}) = \left((\mu\cdot\pi)^T S^\pi_{\widehat{\mathcal{M}}}\right)(\mathbf{s},\mathbf{a})$, we obtain:

$$\mathbb{E}_{\mu,\pi}[\hat{Q}^\pi(\mathbf{s},\mathbf{a})] \leq \mathbb{E}_{\mu,\pi}[Q^\pi_{\widehat{\mathcal{M}}}(\mathbf{s},\mathbf{a})] - \beta\, \mathbb{E}_{\rho(\mathbf{s},\mathbf{a})}\left[\frac{\rho(\mathbf{s},\mathbf{a}) - d(\mathbf{s},\mathbf{a})}{d_f(\mathbf{s},\mathbf{a})}\right] + \text{terms independent of } \beta. \tag{13}$$

The terms marked as "terms independent of β" correspond to the additional positive error terms obtained by iterating $\left|\mathcal{B}^\pi \hat{Q}^k - \mathcal{B}^\pi_{\widehat{\mathcal{M}}} \hat{Q}^k\right|$ and $\left|\mathcal{B}^\pi_{\widehat{\mathcal{M}}} \hat{Q}^k - \mathcal{B}^\pi \hat{Q}^k\right|$, which can be bounded similar to the proof of Proposition A.1 above. Now by replacing the model Q-function, $\mathbb{E}_{\mu,\pi}[Q^\pi_{\widehat{\mathcal{M}}}(\mathbf{s},\mathbf{a})]$ with the actual Q-function, $\mathbb{E}_{\mu,\pi}[Q^\pi(\mathbf{s},\mathbf{a})]$ and adding an error term corresponding to model error to the bound, we obtain that:

$$\mathbb{E}_{\mu,\pi}[\hat{Q}^\pi(\mathbf{s},\mathbf{a})] \leq \mathbb{E}_{\mu,\pi}[Q^\pi(\mathbf{s},\mathbf{a})] + \text{terms independent of } \beta - \beta \underbrace{\mathbb{E}_{\rho(\mathbf{s},\mathbf{a})}\left[\frac{\rho(\mathbf{s},\mathbf{a}) - d(\mathbf{s},\mathbf{a})}{d_f(\mathbf{s},\mathbf{a})}\right]}_{=\nu(\rho,f)>0}. \tag{14}$$

Hence, by choosing β large enough, we obtain the desired lower bound guarantee. □

Remark 1 (COMBO does not underestimate at every $\mathbf{s} \in \mathcal{D}$ unlike CQL.). *Before concluding this section, we discuss how the bound obtained by COMBO (Equation 14) is tighter than CQL. CQL learns a Q-function such that the value of the policy under the resulting Q-function lower-bounds the true value function at each state $\mathbf{s} \in \mathcal{D}$ individually (in the absence of no sampling error), i.e., $\forall \mathbf{s} \in \mathcal{D}, \hat{V}^\pi_{CQL}(\mathbf{s}) \leq V^\pi(\mathbf{s})$, whereas the bound in COMBO is only valid in expectation of the value function over the initial state distribution, i.e., $\mathbb{E}_{\mathbf{s}\sim\mu_0(\mathbf{s})}[\hat{V}^\pi_{COMBO}(\mathbf{s})] \leq \mathbb{E}_{\mathbf{s}\sim\mu_0(\mathbf{s})}[V^\pi(\mathbf{s})]$, and the value function at a given state may not be a lower-bound. For instance, COMBO can overestimate the value of a state more frequent in the dataset distribution $d(\mathbf{s},\mathbf{a})$ but not so frequent in the $\rho(\mathbf{s},\mathbf{a})$ marginal distribution of the policy under the learned model $\widehat{\mathcal{M}}$. To see this more formally, note that the expected penalty added in the effective Bellman backup performed by COMBO (Equation 4), in expectation under the dataset distribution $d(\mathbf{s},\mathbf{a})$, $\widetilde{\nu}(\rho,d,f)$ is actually* ***negative****:*

$$\widetilde{\nu}(\rho,d,f) = \sum_{\mathbf{s},\mathbf{a}} d(\mathbf{s},\mathbf{a}) \frac{\rho(\mathbf{s},\mathbf{a}) - d(\mathbf{s},\mathbf{a})}{d_f(\mathbf{s},\mathbf{a})} = -\sum_{\mathbf{s},\mathbf{a}} d(\mathbf{s},\mathbf{a}) \frac{d(\mathbf{s},\mathbf{a}) - \rho(\mathbf{s},\mathbf{a})}{f d(\mathbf{s},\mathbf{a}) + (1-f)\rho(\mathbf{s},\mathbf{a})} < 0,$$

where the final inequality follows via a direct application of the proof of Lemma A.1. Thus, COMBO actually overestimates *the values at atleast some states (in the dataset) unlike CQL.*

A.3 Proof of Proposition 4.2

In this section, we will provide a proof for Proposition 4.2, and show that the COMBO can be less conservative in terms of the estimated value. To recall, let $\Delta^\pi_{\text{COMBO}} := \mathbb{E}_{\mathbf{s},\mathbf{a}\sim d_{\widehat{\mathcal{M}}}(\mathbf{s}),\pi(\mathbf{a}|\mathbf{s})}\left[\hat{Q}^\pi(\mathbf{s},\mathbf{a}\right]$ and let $\Delta^\pi_{\text{CQL}} := \mathbb{E}_{\mathbf{s},\mathbf{a}\sim d_{\widehat{\mathcal{M}}},\pi(\mathbf{a}|\mathbf{s})}\left[\hat{Q}^\pi_{\text{CQL}}(\mathbf{s},\mathbf{a})\right]$. From Kumar et al. [29], we obtain that $\hat{Q}^\pi_{\text{CQL}}(\mathbf{s},\mathbf{a}) := Q^\pi(\mathbf{s},\mathbf{a}) - \beta\frac{\pi(\mathbf{a}|\mathbf{s}) - \pi_\beta(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})}$. We shall derive the condition for the real data fraction $f = 1$ for COMBO, thus making sure that $d_f(\mathbf{s}) = d^{\pi_\beta}(\mathbf{s})$. To derive the condition when $\Delta^\pi_{\text{COMBO}} \geq \Delta^\pi_{\text{CQL}}$, we note the following simplifications:

$$\Delta^\pi_{\text{COMBO}} \geq \Delta^\pi_{\text{CQL}} \tag{15}$$

$$\implies \sum_{\mathbf{s},\mathbf{a}} d_{\widehat{\mathcal{M}}}(\mathbf{s})\pi(\mathbf{a}|\mathbf{s})\hat{Q}^\pi(\mathbf{s},\mathbf{a}) \geq \sum_{\mathbf{s},\mathbf{a}} d_{\widehat{\mathcal{M}}}(\mathbf{s})\pi(\mathbf{a}|\mathbf{s})\hat{Q}^\pi_{\text{CQL}}(\mathbf{s},\mathbf{a}) \tag{16}$$

$$\implies \beta \sum_{\mathbf{s},\mathbf{a}} d_{\widehat{\mathcal{M}}}(\mathbf{s})\pi(\mathbf{a}|\mathbf{s}) \left(\frac{\rho(\mathbf{s},\mathbf{a}) - d^{\pi_\beta}(\mathbf{s})\pi_\beta(\mathbf{a}|\mathbf{s})}{d^{\pi_\beta}(\mathbf{s})\pi_\beta(\mathbf{a}|\mathbf{s})}\right) \leq \beta \sum_{\mathbf{s},\mathbf{a}} d_{\widehat{\mathcal{M}}}(\mathbf{s})\pi(\mathbf{a}|\mathbf{s}) \left(\frac{\pi(\mathbf{a}|\mathbf{s}) - \pi_\beta(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})}\right). \tag{17}$$

Now, in the expression on the left-hand side, we add and subtract $d^{\pi_\beta}(\mathbf{s})\pi(\mathbf{a}|\mathbf{s})$ from the numerator inside the paranthesis.

$$\sum_{\mathbf{s},\mathbf{a}} d_{\overline{\mathcal{M}}}(\mathbf{s},\mathbf{a}) \left(\frac{\rho(\mathbf{s},\mathbf{a}) - d^{\pi_\beta}(\mathbf{s})\pi_\beta(\mathbf{a}|\mathbf{s})}{d^{\pi_\beta}(\mathbf{s})\pi_\beta(\mathbf{a}|\mathbf{s})} \right) \tag{18}$$

$$= \sum_{\mathbf{s},\mathbf{a}} d_{\overline{\mathcal{M}}}(\mathbf{s},\mathbf{a}) \left(\frac{\rho(\mathbf{s},\mathbf{a}) - d^{\pi_\beta}(\mathbf{s})\pi(\mathbf{a}|\mathbf{s}) + d^{\pi_\beta}(\mathbf{s})\pi(\mathbf{a}|\mathbf{s}) - d^{\pi_\beta}(\mathbf{s})\pi_\beta(\mathbf{a}|\mathbf{s})}{d^{\pi_\beta}(\mathbf{s})\pi_\beta(\mathbf{a}|\mathbf{s})} \right) \tag{19}$$

$$= \underbrace{\sum_{\mathbf{s},\mathbf{a}} d_{\overline{\mathcal{M}}}(\mathbf{s},\mathbf{a}) \frac{\pi(\mathbf{a}|\mathbf{s}) - \pi_\beta(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})}}_{(1)} + \sum_{\mathbf{s},\mathbf{a}} d_{\overline{\mathcal{M}}}(\mathbf{s},\mathbf{a}) \cdot \frac{\rho(\mathbf{s}) - d^{\pi_\beta}(\mathbf{s})}{d^{\pi_\beta}(\mathbf{s})} \cdot \frac{\pi(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} \tag{20}$$

The term marked (1) is identical to the CQL term that appears on the right in Equation 17. Thus the inequality in Equation 17 is satisfied when the second term above is negative. To show this, first note that $d^{\pi_\beta}(\mathbf{s}) = d_{\overline{\mathcal{M}}}(\mathbf{s})$ which results in a cancellation. Finally, re-arranging the second term into expectations gives us the desired result. An analogous condition can be derived when $f \neq 1$, but we omit that derivation as it will be hard to interpret terms appear in the final inequality.

A.4 Proof of Proposition 4.3

To prove the policy improvement result in Proposition 4.3, we first observe that using Equation 4 for Bellman backups amounts to finding a policy that maximizes the return of the policy in the a modified "f-interpolant" MDP which admits the Bellman backup $\widehat{\mathcal{B}}^\pi$, and is induced by a linear interpolation of backups in the empirical MDP $\overline{\mathcal{M}}$ and the MDP induced by a dynamics model $\widehat{\mathcal{M}}$ and the return of a policy π in this effective f-interpolant MDP is denoted by $J(\overline{\mathcal{M}}, \widehat{\mathcal{M}}, f, \pi)$. Alongside this, the return is penalized by the conservative penalty where ρ^π denotes the marginal state-action distribution of policy π in the learned model $\widehat{\mathcal{M}}$.

$$\hat{J}(f,\pi) = J(\overline{\mathcal{M}}, \widehat{\mathcal{M}}, f, \pi) - \beta \frac{\nu(\rho^\pi, f)}{1-\gamma}. \tag{21}$$

We will require bounds on the return of a policy π in this f-interpolant MDP, $J(\overline{\mathcal{M}}, \widehat{\mathcal{M}}, f, \pi)$, which we first prove separately as Lemma A.2 below and then move to the proof of Proposition 4.3.

Lemma A.2 (Bound on return in f-interpolant MDP)**.** *For any two MDPs, $\mathcal{M}_1$ and $\mathcal{M}_2$, with the same state-space, action-space and discount factor, and for a given fraction $f \in [0,1]$, define the f-interpolant MDP $\mathcal{M}_f$ as the MDP on the same state-space, action-space and with the same discount as the MDP with dynamics: $P_{\mathcal{M}_f} := fP_{\mathcal{M}_1} + (1-f)P_{\mathcal{M}_2}$ and reward function: $r_{\mathcal{M}_f} := fr_{\mathcal{M}_1} + (1-f)r_{\mathcal{M}_2}$. Then, given any auxiliary MDP, $\mathcal{M}$, the return of any policy π in $\mathcal{M}_f$, $J(\pi, \mathcal{M}_f)$, also denoted by $J(\mathcal{M}_1, \mathcal{M}_2, f, \pi)$, lies in the interval:*

$$\big[J(\pi,\mathcal{M}) - \alpha,\ \ J(\pi,\mathcal{M}) + \alpha\big], \qquad \textit{where } \alpha \textit{ is given by:}$$

$$\alpha = \frac{2\gamma(1-f)}{(1-\gamma)^2} R_{\max} D\left(P_{\mathcal{M}_2}, P_{\mathcal{M}}\right) + \frac{\gamma f}{1-\gamma} \left| \mathbb{E}_{d^\pi_{\mathcal{M}}\pi} \left[\left(P^\pi_{\mathcal{M}} - P^\pi_{\mathcal{M}_1} \right) Q^\pi_{\mathcal{M}} \right] \right|$$
$$+ \frac{f}{1-\gamma} \mathbb{E}_{\mathbf{s},\mathbf{a}\sim d^\pi_{\mathcal{M}}\pi} [|r_{\mathcal{M}_1}(\mathbf{s},\mathbf{a}) - r_{\mathcal{M}}(\mathbf{s},\mathbf{a})|] + \frac{1-f}{1-\gamma} \mathbb{E}_{\mathbf{s},\mathbf{a}\sim d^\pi_{\mathcal{M}}\pi} [|r_{\mathcal{M}_2}(\mathbf{s},\mathbf{a}) - r_{\mathcal{M}}(\mathbf{s},\mathbf{a})|]. \tag{22}$$

Proof. To prove this lemma, we note two general inequalities. First, note that for a fixed transition dynamics, say P, the return decomposes linearly in the components of the reward as the expected return is linear in the reward function:

$$J(P, r_{\mathcal{M}_f}) = J(P, fr_{\mathcal{M}_1} + (1-f)r_{\mathcal{M}_2}) = fJ(P, r_{\mathcal{M}_1}) + (1-f)J(P, r_{\mathcal{M}_2}).$$

As a result, we can bound $J(P, r_{\mathcal{M}_f})$ using $J(P, r)$ for a new reward function r of the auxiliary MDP, $\mathcal{M}$, as follows

$$\begin{aligned}
J(P, r_{\mathcal{M}_f}) &= J(P, f r_{\mathcal{M}_1} + (1-f) r_{\mathcal{M}_2}) = J(P, r + f(r_{\mathcal{M}_1} - r) + (1-f)(r_{\mathcal{M}_2} - r) \\
&= J(P, r) + f J(P, r_{\mathcal{M}_1} - r) + (1-f) J(P, r_{\mathcal{M}_2} - r) \\
&= J(P, r) + \frac{f}{1-\gamma} \mathbb{E}_{\mathbf{s},\mathbf{a} \sim d^{\pi}_{\mathcal{M}}(\mathbf{s})\pi(\mathbf{a}|\mathbf{s})} \left[r_{\mathcal{M}_1}(\mathbf{s}, \mathbf{a}) - r(\mathbf{s}, \mathbf{a}) \right] \\
&+ \frac{1-f}{1-\gamma} \mathbb{E}_{\mathbf{s},\mathbf{a} \sim d^{\pi}_{\mathcal{M}}(\mathbf{s})\pi(\mathbf{a}|\mathbf{s})} \left[r_{\mathcal{M}_2}(\mathbf{s}, \mathbf{a}) - r(\mathbf{s}, \mathbf{a}) \right].
\end{aligned}$$

Second, note that for a given reward function, r, but a linear combination of dynamics, the following bound holds:

$$\begin{aligned}
J(P_{\mathcal{M}_f}, r) &= J(f P_{\mathcal{M}_1} + (1-f) P_{\mathcal{M}_2}, r) \\
&= J(P_{\mathcal{M}} + f(P_{\mathcal{M}_1} - P_{\mathcal{M}}) + (1-f)(P_{\mathcal{M}_2} - P_{\mathcal{M}}), r) \\
&= J(P_{\mathcal{M}}, r) - \frac{\gamma(1-f)}{1-\gamma} \mathbb{E}_{\mathbf{s},\mathbf{a} \sim d^{\pi}_{\mathcal{M}}(\mathbf{s})\pi(\mathbf{a}|\mathbf{s})} \left[\left(P^{\pi}_{\mathcal{M}_2} - P^{\pi}_{\mathcal{M}} \right) Q^{\pi}_{\mathcal{M}} \right] \\
&- \frac{\gamma f}{1-\gamma} \mathbb{E}_{\mathbf{s},\mathbf{a} \sim d^{\pi}_{\mathcal{M}}(\mathbf{s})\pi(\mathbf{a}|\mathbf{s})} \left[\left(P^{\pi}_{\mathcal{M}} - P^{\pi}_{\mathcal{M}_1} \right) Q^{\pi}_{\mathcal{M}} \right] \\
&\in \Bigg[J(P_{\mathcal{M}}, r) \pm \Bigg(\frac{\gamma f}{(1-\gamma)} \left| \mathbb{E}_{\mathbf{s},\mathbf{a} \sim d^{\pi}_{\mathcal{M}}(\mathbf{s})\pi(\mathbf{a}|\mathbf{s})} \left[\left(P^{\pi}_{\mathcal{M}} - P^{\pi}_{\mathcal{M}_1} \right) Q^{\pi}_{\mathcal{M}} \right] \right| \\
&+ \frac{2\gamma(1-f) R_{\max}}{(1-\gamma)^2} D(P_{\mathcal{M}_2}, P_{\mathcal{M}}) \Bigg) \Bigg].
\end{aligned}$$

To observe the third equality, we utilize the result on the difference between returns of a policy π on two different MDPs, $P_{\mathcal{M}_1}$ and $P_{\mathcal{M}_f}$ from Agarwal et al. [1] (Chapter 2, Lemma 2.2, Simulation Lemma), and additionally incorporate the auxiliary MDP $\mathcal{M}$ in the expression via addition and subtraction in the previous (second) step. In the fourth step, we finally bound one term that corresponds to the learned model via the total-variation divergence $D(P_{\mathcal{M}_2}, P_{\mathcal{M}})$ and the other term corresponding to the empirical MDP $\overline{\mathcal{M}}$ is left in its expectation form to be bounded later.

Using the above bounds on return for reward-mixtures and dynamics-mixtures, proving this lemma is straightforward:

$$\begin{aligned}
&J(\mathcal{M}_1, \mathcal{M}_2, f, \pi) := J(P_{\mathcal{M}_f}, f r_{\mathcal{M}_1} + (1-f) r_{\mathcal{M}_2}) = J(f P_{\mathcal{M}_1} + (1-f) P_{\mathcal{M}_2}, r_{\mathcal{M}_f}) \\
&\in \Big[J(P_{\mathcal{M}_f}, r_{\mathcal{M}}) \pm \\
&\underbrace{\left(\frac{f}{1-\gamma} \mathbb{E}_{\mathbf{s},\mathbf{a} \sim d^{\pi}_{\mathcal{M}}\pi} [|r_{\mathcal{M}_1}(\mathbf{s}, \mathbf{a}) - r_{\mathcal{M}}(\mathbf{s}, \mathbf{a})|] + \frac{1-f}{1-\gamma} \mathbb{E}_{\mathbf{s},\mathbf{a} \sim d^{\pi}_{\mathcal{M}}\pi} [|r_{\mathcal{M}_2}(\mathbf{s}, \mathbf{a}) - r_{\mathcal{M}}(\mathbf{s}, \mathbf{a})|] \right)}_{:= \Delta_R} \Big],
\end{aligned}$$

where the second step holds via linear decomposition of the return of π in $\mathcal{M}_f$ with respect to the reward interpolation, and bounding the terms that appear in the reward difference. For convenience, we refer to these offset terms due to the reward as Δ_R. For the final part of this proof, we bound $J(P_{\mathcal{M}_f}, r_{\mathcal{M}})$ in terms of the return on the actual MDP, $J(P_{\mathcal{M}}, r_{\mathcal{M}})$, using the inequality proved above that provides intervals for mixture dynamics but a fixed reward function. Thus, the overall bound is given by $J(\pi, \mathcal{M}_f) \in [J(\pi, \mathcal{M}) - \alpha, J(\pi, \mathcal{M}) + \alpha]$, where α is given by:

$$\alpha = \frac{2\gamma(1-f)}{(1-\gamma)^2} R_{\max} D\left(P_{\mathcal{M}_2}, P_{\mathcal{M}}\right) + \frac{\gamma f}{1-\gamma} \left| \mathbb{E}_{d^{\pi}_{\mathcal{M}}\pi} \left[\left(P^{\pi}_{\mathcal{M}} - P^{\pi}_{\mathcal{M}_1} \right) Q^{\pi}_{\mathcal{M}} \right] \right| + \Delta_R. \tag{23}$$

This concludes the proof of this lemma. □

Finally, we prove Theorem 4.3 that shows how policy optimization with respect to $\hat{J}(f, \pi)$ affects the performance in the actual MD by using Equation 21 and building on the analysis of pure model-free algorithms from Kumar et al. [29]. We restate a more complete statement of the theorem below and present the constants at the end of the proof.

Theorem 2 (Formal version of Proposition 4.3)**.** *Let $\hat{\pi}_{out}(\mathbf{a}|\mathbf{s})$ be the policy obtained by COMBO. Assume $\nu(\rho^{\pi_{out}}, f) - \nu(\rho^{\beta}, f) \geq C$ for some constant $C > 0$. Then, the policy $\pi_{out}(\mathbf{a}|\mathbf{s})$ is a ζ-safe policy improvement over π_β in the actual MDP $\mathcal{M}$, i.e., $J(\pi_{out}, \mathcal{M}) \geq J(\pi_\beta, \mathcal{M}) - \zeta$, with probability at least $1 - \delta$, where ζ is given by (where $\rho^{\beta}(\mathbf{s}, \mathbf{a}) := d^{\pi_\beta}_{\widehat{\mathcal{M}}}(\mathbf{s}, \mathbf{a})$):*

$$\mathcal{O}\left(\frac{\gamma f}{(1-\gamma)^2}\right)\left[\mathbb{E}_{\mathbf{s}\sim d^{\pi_{out}}_{\mathcal{M}}}\left[\sqrt{\frac{|\mathcal{A}|}{|\mathcal{D}(\mathbf{s})|}(\mathrm{D}_{CQL}(\pi_{out}, \pi_\beta) + 1)}\right]\right] + \mathcal{O}\left(\frac{\gamma(1-f)}{(1-\gamma)^2}\right)\mathrm{D}_{\mathrm{TV}}(P_{\mathcal{M}}, P_{\widehat{\mathcal{M}}}) - \beta\frac{C}{(1-\gamma)}.$$

Proof. We first note that since policy improvement is not being performed in the same MDP, $\mathcal{M}$ as the f-interpolant MDP, $\mathcal{M}_f$, we need to upper and lower bound the amount of improvement occurring in the actual MDP due to the f-interpolant MDP. As a result our first is to relate $J(\pi, \mathcal{M})$ and $J(\pi, \mathcal{M}_f) := J(\overline{\mathcal{M}}, \widehat{\mathcal{M}}, f, \pi)$ for any given policy π.

Step 1: Bounding the return in the actual MDP due to optimization in the f-interpolant MDP. By directly applying Lemma A.2 stated and proved previously, we obtain the following upper and lower-bounds on the return of a policy π:

$$J(\overline{\mathcal{M}}, \widehat{\mathcal{M}}, f, \pi) \in \left[J(\pi, \mathcal{M}) - \alpha,\ \ J(\pi, \mathcal{M}) + \alpha\right],$$

where α is shown in Equation 22. As a result, we just need to bound the terms appearing the expression of α to obtain a bound on the return differences. We first note that the terms in the expression for α are of two types: **(1)** terms that depend only on the reward function differences (captured in Δ_R in Equation 23), and **(2)** terms that depend on the dynamics (the other two terms in Equation 23).

To bound Δ_R, we simply appeal to concentration inequalities on reward (Assumption A1), and bound Δ_R as:

$$\begin{aligned}\Delta_R &:= \frac{f}{1-\gamma}\mathbb{E}_{\mathbf{s},\mathbf{a}\sim d^{\pi}_{\mathcal{M}}\pi}[|r_{\mathcal{M}_1}(\mathbf{s}, \mathbf{a}) - r_{\mathcal{M}}(\mathbf{s}, \mathbf{a})|] + \frac{1-f}{1-\gamma}\mathbb{E}_{\mathbf{s},\mathbf{a}\sim d^{\pi}_{\mathcal{M}}\pi}[|r_{\mathcal{M}_2}(\mathbf{s}, \mathbf{a}) - r_{\mathcal{M}}(\mathbf{s}, \mathbf{a})|] \\ &\leq \frac{C_{r,\delta}}{1-\gamma}\mathbb{E}_{\mathbf{s},\mathbf{a}\sim d^{\pi}_{\mathcal{M}}\pi}\left[\frac{1}{\sqrt{D(\mathbf{s}, \mathbf{a})}}\right] + \frac{1}{1-\gamma}||R_{\mathcal{M}} - R_{\widehat{\mathcal{M}}}|| := \Delta^u_R.\end{aligned}$$

Note that both of these terms are of the order of $\mathcal{O}(1/(1-\gamma))$ and hence they don't figure in the informal bound in Theorem 4.3 in the main text, as these are dominated by terms that grow quadratically with the horizon. To bound the remaining terms in the expression for α, we utilize a result directly from Kumar et al. [29] for the empirical MDP, $\overline{\mathcal{M}}$, which holds for any policy $\pi(\mathbf{a}|\mathbf{s})$, as shown below.

$$\begin{aligned}&\frac{\gamma}{(1-\gamma)}\left|\mathbb{E}_{\mathbf{s},\mathbf{a}\sim d^{\pi}_{\mathcal{M}}(\mathbf{s})\pi(\mathbf{a}|\mathbf{s})}\left[\left(P^{\pi}_{\mathcal{M}} - P^{\pi}_{\mathcal{M}_1}\right)Q^{\pi}_{\mathcal{M}}\right]\right| \\ &\leq \frac{2\gamma R_{\max} C_{P,\delta}}{(1-\gamma)^2}\mathbb{E}_{\mathbf{s}\sim d^{\pi}_{\mathcal{M}}(\mathbf{s})}\left[\frac{\sqrt{|\mathcal{A}|}}{\sqrt{|\mathcal{D}(\mathbf{s})|}}\sqrt{D_{\mathrm{CQL}}(\pi, \pi_\beta)(\mathbf{s}) + 1}\right].\end{aligned}$$

Step 2: Incorporate policy improvement in the f-inrerpolant MDP. Now we incorporate the improvement of policy π_{out} over the policy π_β on a weighted mixture of $\widehat{\mathcal{M}}$ and $\overline{\mathcal{M}}$. In what follows, we derive a lower-bound on this improvement by using the fact that policy π_{out} is obtained by maximizing $\hat{J}(f, \pi)$ from Equation 21. As a direct consequence of Equation 21, we note that

$$\hat{J}(f, \pi_{\text{out}}) = J(\overline{\mathcal{M}}, \widehat{\mathcal{M}}, f, \pi_{\text{out}}) - \beta\frac{\nu(\rho^{\pi}, f)}{1-\gamma} \geq \hat{J}(f, \pi_\beta) = J(\overline{\mathcal{M}}, \widehat{\mathcal{M}}, f, \pi_\beta) - \beta\frac{\nu(\rho^{\beta}, f)}{1-\gamma} \quad (24)$$

Following **Step 1**, we will use the upper bound on $J(\overline{\mathcal{M}}, \widehat{\mathcal{M}}, f, \pi)$ for policy $\pi = \pi_{\text{out}}$ and a lower-bound on $J(\overline{\mathcal{M}}, \widehat{\mathcal{M}}, f, \pi)$ for policy $\pi = \pi_\beta$ and obtain the following inequality:

$$J(\pi_{\text{out}}, \mathcal{M}) - \beta \frac{\nu(\rho^\pi, f)}{1-\gamma} \geq \Big\{ J(\pi_\beta, \mathcal{M}) - \beta \frac{\nu(\rho^\beta, f)}{1-\gamma} - \frac{4\gamma(1-f)R_{\max}}{(1-\gamma)^2} D(P_{\mathcal{M}}, P_{\widehat{\mathcal{M}}}) \\ \underbrace{- \frac{2\gamma f}{(1-\gamma)} \left| \mathbb{E}_{d^{\pi_{\text{out}}}_{\mathcal{M}}} \left[\left(P^{\pi_{\text{out}}}_{\mathcal{M}} - P^{\pi_{\text{out}}}_{\overline{\mathcal{M}}} \right) Q^{\pi_{\text{out}}}_{\mathcal{M}} \right] \right|}_{:=(*)} \\ \underbrace{- \frac{4\gamma R_{\max} C_{P,\delta} f}{(1-\gamma)^2} \mathbb{E}_{\mathbf{s} \sim d^{\pi_\beta}_{\mathcal{M}}} \left[\sqrt{\frac{|\mathcal{A}|}{|\mathcal{D}(\mathbf{s})|}} \right]}_{:=(\wedge)} - \Delta^u_R \Big\}.$$

The term marked by $(*)$ in the above expression can be upper bounded by the concentration properties of the dynamics as done in Step 1 in this proof:

$$(*) \leq \frac{4\gamma f C_{P,\delta} R_{\max}}{(1-\gamma)^2} \mathbb{E}_{\mathbf{s} \sim d^{\pi_{\text{out}}}_{\mathcal{M}}(\mathbf{s})} \left[\frac{\sqrt{|\mathcal{A}|}}{\sqrt{|\mathcal{D}(\mathbf{s})|}} \sqrt{D_{\text{CQL}}(\pi_{\text{out}}, \pi_\beta)(\mathbf{s}) + 1} \right]. \tag{25}$$

Finally, using Equation 25, we can lower-bound the policy return difference as:

$$J(\pi_{\text{out}}, \mathcal{M}) - J(\pi_\beta, \mathcal{M}) \geq \beta \frac{\nu(\rho^\pi, f)}{1-\gamma} - \beta \frac{\nu(\rho^\beta, f)}{1-\gamma} - \frac{4\gamma(1-f)R_{\max}}{(1-\gamma)^2} D(P_{\mathcal{M}}, P_{\widehat{\mathcal{M}}}) - (*) - \Delta^u_R \\ \geq \beta \frac{C}{1-\gamma} - \frac{4\gamma(1-f)R_{\max}}{(1-\gamma)^2} D(P_{\mathcal{M}}, P_{\widehat{\mathcal{M}}}) - (*) - \Delta^u_R.$$

Plugging the bounds for terms (a), (b) and (c) in the expression for ζ where $J(\pi_{\text{out}}, \mathcal{M}) - J(\pi_\beta, \mathcal{M}) \geq \zeta$, we obtain:

$$\zeta = \left(\frac{4f\gamma R_{\max} C_{P,\delta}}{(1-\gamma)^2} \right) \mathbb{E}_{\mathbf{s} \sim d^{\pi_{\text{out}}}_{\mathcal{M}}(\mathbf{s})} \left[\frac{\sqrt{|\mathcal{A}|}}{\sqrt{|\mathcal{D}(\mathbf{s})|}} \sqrt{D_{\text{CQL}}(\pi_{\text{out}}, \pi_\beta)(\mathbf{s}) + 1} \right] + (\wedge) - \Delta^u_R \\ + \frac{4(1-f)\gamma R_{\max}}{(1-\gamma)^2} D(P_{\mathcal{M}}, P_{\widehat{\mathcal{M}}}) - \beta \frac{C}{1-\gamma}. \tag{26}$$

□

Remark 3 (Interpretation of Proposition 4.3). *Now we will interpret the theoretical expression for ζ in Equation 26, and discuss the scenarios when it is* negative. *When the expression for ζ is negative, the policy π_{out} is an improvement over π_β in the original MDP, $\mathcal{M}$.*

- *We first discuss if the assumption of $\nu(\rho^{\pi_{out}}, f) - \nu(\rho^\beta, f) \geq C > 0$ is reasonable in practice. Note that we have never used the fact that the learned model $P_{\widehat{\mathcal{M}}}$ is close to the actual MDP, $P_{\mathcal{M}}$ on the states visited by the behavior policy π_β in our analysis. We will use this fact now: in practical scenarios, $\nu(\rho^\beta, f)$ is expected to be smaller than $\nu(\rho^\pi, f)$, since $\nu(\rho^\beta, f)$ is directly controlled by the difference and density ratio of $\rho^\beta(\mathbf{s}, \mathbf{a})$ and $d(\mathbf{s}, \mathbf{a})$: $\nu(\rho^\beta, f) \leq \nu(\rho^\beta, f = 1) = \sum_{\mathbf{s},\mathbf{a}} d^{\pi_\beta}_{\widehat{\mathcal{M}}}(\mathbf{s}, \mathbf{a}) \left(d^{\pi_\beta}_{\widehat{\mathcal{M}}}(\mathbf{s}, \mathbf{a}) / d^{\pi_\beta}_{\overline{\mathcal{M}}}(\mathbf{s}, \mathbf{a}) - 1 \right)^2$ by Lemma A.1 which is expected to be small for the behavior policy π_β in cases when the behavior policy marginal in the empirical MDP, $d^{\pi_\beta}_{\overline{\mathcal{M}}}(\mathbf{s}, \mathbf{a})$, is broad. This is a direct consequence of the fact that the learned dynamics integrated with the policy under the learned model: $P^{\pi_\beta}_{\widehat{\mathcal{M}}}$ is closer to its counterpart in the empirical MDP: $P^{\pi_\beta}_{\overline{\mathcal{M}}}$ for π_β. Note that this is not true for any other policy besides the behavior policy that performs several counterfactual actions in a rollout and deviates from the data. For such a learned policy π, we incur an extra error which depends on the importance ratio of policy densities, compounded over the horizon and manifests as the D_{CQL} term (similar to Equation 25, or Lemma D.4.1 in Kumar et al. [29]). Thus, in practice, we argue that we are interested in situations where the assumption $\nu(\rho^{\pi_{out}}, f) - \nu(\rho^\beta, f) \geq C > 0$ holds, in which case by increasing β, we can make the expression for ζ in Equation 26 negative, allowing for policy improvement.*

- *In addition, note that when f is close to 1, the bound reverts to a standard model-free policy improvement bound and when f is close to 0, the bound reverts to a typical model-based policy improvement bound. In scenarios with high sampling error (i.e. smaller $|\mathcal{D}(\mathbf{s})|$), if we can learn a good model, i.e., $D(P_{\mathcal{M}}, P_{\widehat{\mathcal{M}}})$ is small, we can attain policy improvement better than model-free methods by relying on the learned model by setting f closer to 0. A similar argument can be made in reverse for handling cases when learning an accurate dynamics model is hard.*

B Experimental details

In this section, we include all details of our empirical evaluations of COMBO.

B.1 Practical algorithm implementation details

Model training. In the setting where the observation space is low-dimensional, as mentioned in Section 3, we represent the model as a probabilistic neural network that outputs a Gaussian distribution over the next state and reward given the current state and action:

$$\widehat{T}_\theta(\mathbf{s}_{t+1}, r|\mathbf{s}, \mathbf{a}) = \mathcal{N}(\mu_\theta(\mathbf{s}_t, \mathbf{a}_t), \Sigma_\theta(\mathbf{s}_t, \mathbf{a}_t)).$$

We train an ensemble of 7 such dynamics models following [20] and pick the best 5 models based on the validation prediction error on a held-out set that contains 1000 transitions in the offline dataset $\mathcal{D}$. During model rollouts, we randomly pick one dynamics model from the best 5 models. Each model in the ensemble is represented as a 4-layer feedforward neural network with 200 hidden units. For the generalization experiments in Section 5.1, we additionally use a two-head architecture to output the mean and variance after the last hidden layer following [67].

In the image-based setting, we follow Rafailov et al. [48] and use a variational model with the following components:

$$\begin{aligned}
&\text{Image encoder:} && \mathbf{h}_t = E_\theta(\mathbf{o}_t) \\
&\text{Inference model:} && \mathbf{s}_t \sim q_\theta(\mathbf{s}_t|\mathbf{h}_t, \mathbf{s}_{t-1}, \mathbf{a}_{t-1}) \\
&\text{Latent transition model:} && \mathbf{s}_t \sim \widehat{T}_\theta(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_{t-1}) \\
&\text{Reward predictor:} && r_t \sim p_\theta(r_t|\mathbf{s}_t) \\
&\text{Image decoder:} && \mathbf{o}_t \sim D_\theta(\mathbf{o}_t|\mathbf{s}_t).
\end{aligned} \tag{27}$$

We train the model using the evidence lower bound:

$$\max_\theta \sum_{\tau=0}^{T-1} \left[\mathbb{E}_{q_\theta}[\log D_\theta(\mathbf{o}_{\tau+1}|\mathbf{s}_{\tau+1})]\right] - \mathbb{E}_{q_\theta}\left[D_{KL}[q_\theta(\mathbf{o}_{\tau+1}, \mathbf{s}_{\tau+1}|\mathbf{s}_\tau, \mathbf{a}_\tau)\|\widehat{T}_{\theta_\tau}(\mathbf{s}_{\tau+1}, a_{\tau+1})]\right]$$

At each step τ we sample a latent forward model $\widehat{T}_{\theta_\tau}$ from a fixed set of K models $[\widehat{T}_{\theta_1}, \ldots, \widehat{T}_{\theta_K}]$. For the encoder E_θ we use a convolutional neural network with kernel size 4 and stride 2. For the Walker environment we use 4 layers, while the Door Opening task has 5 layers. The D_θ is a transposed convolutional network with stride 2 and kernel sizes $[5, 5, 6, 6]$ and $[5, 5, 5, 6, 6]$ respectively. The inference network has a two-level structure similar to Hafner et al. [18] with a deterministic path using a GRU cell with 256 units and a stochastic path implemented as a conditional diagonal Gaussian with 128 units. We only train an ensemble of stochastic forward models, which are also implemented as conditional diagonal Gaussians.

Policy Optimization. We sample a batch size of 256 transitions for the critic and policy learning. We set $f = 0.5$, which means we sample 50% of the batch of transitions from $\mathcal{D}$ and another 50% from $\mathcal{D}_{\text{model}}$. The equal split between the offline data and the model rollouts strikes the balance between conservatism and generalization in our experiments as shown in our experimental results in Section 5. We represent the Q-networks and policy as 3-layer feedforward neural networks with 256 hidden units.

For the choice of $\rho(\mathbf{s}, \mathbf{a})$ in Equation 2, we can obtain the Q-values that lower-bound the true value of the learned policy π by setting $\rho(\mathbf{s}, \mathbf{a}) = d^{\pi}_{\widehat{\mathcal{M}}}(\mathbf{s})\pi(\mathbf{a}|\mathbf{s})$. However, as discussed in [29], computing π by alternating the full off-policy evaluation for the policy $\hat{\pi}^k$ at each iteration k and one step of policy improvement is computationally expensive. Instead, following [29], we pick a particular distribution $\psi(\mathbf{a}|\mathbf{s})$ that approximates the the policy that maximizes the Q-function at the current iteration and set $\rho(\mathbf{s}, \mathbf{a}) = d^{\pi}_{\widehat{\mathcal{M}}}(\mathbf{s})\psi(\mathbf{a}|\mathbf{s})$. We formulate the new objective as follows:

$$\hat{Q}^{k+1} \leftarrow \arg\min_{Q} \beta \left(\mathbb{E}_{\mathbf{s} \sim d^{\pi}_{\widehat{\mathcal{M}}}(\mathbf{s}), \mathbf{a} \sim \psi(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s}, \mathbf{a})] - \mathbb{E}_{\mathbf{s}, \mathbf{a} \sim \mathcal{D}}[Q(\mathbf{s}, \mathbf{a})] \right) + \frac{1}{2} \mathbb{E}_{\mathbf{s}, \mathbf{a}, \mathbf{s}' \sim d_f} \left[\left(Q(\mathbf{s}, \mathbf{a}) - \widehat{\mathcal{B}}^{\pi} \hat{Q}^k(\mathbf{s}, \mathbf{a}) \right)^2 \right] + \mathcal{R}(\psi), \tag{28}$$

where $\mathcal{R}(\psi)$ is a regularizer on ψ. In practice, we pick $\mathcal{R}(\psi)$ to be the $-D_{\text{KL}}(\psi(\mathbf{a}|\mathbf{s})\|\text{Unif}(\mathbf{a}))$ and under such a regularization, the first term in Equation 28 corresponds to computing softmax of the Q-values at any state $\mathbf{s}$ as follows:

$$\hat{Q}^{k+1} \leftarrow \arg\min_{Q} \max_{\psi} \beta \left(\mathbb{E}_{\mathbf{s} \sim d^{\pi}_{\widehat{\mathcal{M}}}(\mathbf{s})} \left[\log \sum_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a}) \right] - \mathbb{E}_{\mathbf{s}, \mathbf{a} \sim \mathcal{D}}[Q(\mathbf{s}, \mathbf{a})] \right) + \frac{1}{2} \mathbb{E}_{\mathbf{s}, \mathbf{a}, \mathbf{s}' \sim d_f} \left[\left(Q(\mathbf{s}, \mathbf{a}) - \widehat{\mathcal{B}}^{\pi} \hat{Q}^k(\mathbf{s}, \mathbf{a}) \right)^2 \right]. \tag{29}$$

We estimate the `log-sum-exp` term in Equation 29 by sampling 10 actions at every state $\mathbf{s}$ in the batch from a uniform policy $\text{Unif}(\mathbf{a})$ and the current learned policy $\pi(\mathbf{a}|\mathbf{s})$ with importance sampling following [29].

B.2 Hyperparameter Selection

In this section, we discuss the hyperparameters that we use for COMBO. In the D4RL and generalization experiments, our method are built upon the implementation of MOPO provided at: https://github.com/tianheyu927/mopo. The hyperparameters used in COMBO that relates to the backbone RL algorithm SAC such as twin Q-functions and number of gradient steps follow from those used in MOPO with the exception of smaller critic and policy learning rates, which we will discuss below. In the image-based domains, COMBO is built upon LOMPO without any changes to the parameters used there. For the evaluation of COMBO, we follow the evaluation protocol in D4RL [12] and a variety of prior offline RL works [29, 67, 26] and report the normalized score of the smooth undiscounted averaged return over 3 random seeds for all environments except `sawyer-door-close` and `sawyer-door` where we report the average success rate over 3 random seeds. As mentioned in Section 3, we use the regularization objective in Eq. 2 to select the hyperparameter from a range of pre-specified candidates in a fully offline manner, unlike prior model-based offline RL schemes such as [67] and [26] that similar hyperparameters as COMBO and tune them manually based on policy performance obtained via online rollouts.

We now list the additional hyperparameters as follows.

- **Rollout length h.** We perform a short-horizon model rollouts in COMBO similar to Yu et al. [67] and Rafailov et al. [48]. For the D4RL experiments and generalization experiments, we followed the defaults used in MOPO and used $h = 1$ for walker2d and `sawyer-door-close`, $h = 5$ for hopper, halfcheetah and `halfcheetah-jump`, and $h = 25$ for `ant-angle`. In the image-based domain we used rollout length of $h = 5$ for both the the `walker-walk` and `sawyer-door-open` environments following the same hyperparameters used in Rafailov et al. [48].
- **Q-function and policy learning rates.** On state-based domains, we apply our automatic selection rule to the set $\{1e-4, 3e-4\}$ for the Q-function learning rate and the set $\{1e-5, 3e-5, 1e-4\}$ for the policy learning rate. We found that $3e-4$ for the Q-function learning rate (also used previously in Kumar et al. [29]) and $1e-4$ for the policy learning rate (also recommended previously in Kumar et al. [29] for gym domains) work well for almost all domains except that on walker2d where a smaller Q-function learning rate of $1e-4$ and a correspondingly smaller policy learning rate of $1e-5$ works the best according to our automatic hyperparameter selection scheme. In the image-based domains, we followed the defaults from prior work [48] and used $3e-4$ for both the policy and Q-function.

- **Conservative coefficient β.** We use our hyperparameter selection rule to select the right β from the set $\{0.5, 1.0, 5.0\}$ for β, which correspond to low conservatism, medium conservatism and high conservatism. A larger β would be desirable in more narrow dataset distributions with lower-coverage of the state-action space that propagates error in a backup whereas a smaller β is desirable with diverse dataset distributions. On the D4RL experiments, we found that $\beta = 0.5$ works well for halfcheetah agnostic of dataset quality, while on hopper and walker2d, we found that the more "narrow" dataset distributions: medium and medium-expert datasets work best with larger $\beta = 5.0$ whereas more "diverse" dataset distributions: random and medium-replay datasets work best with smaller $\beta = 0.5$ which is consistent with the intuition. On generalization experiments, $\beta = 1.0$ works best for all environments. In the image-domains we use $\beta = 0.5$ for the medium-replay `walker-walk` task and and $\beta = 1.0$ for all other domains, which again is in accordance with the impact of β on performance.
- **Choice of $\rho(\mathbf{s}, \mathbf{a})$.** We first decouple $\rho(\mathbf{s}, \mathbf{a}) = \rho(\mathbf{s})\rho(\mathbf{a}|\mathbf{s})$ for convenience. As discussed in Appendix B.1, we use $\rho(\mathbf{a}|\mathbf{s})$ as the soft-maximum of the Q-values and estimated with `log-sum-exp`. For $\rho(\mathbf{s})$, we apply the automatic hyperparameter selection rule to the set $\{d^{\pi}_{\widehat{\mathcal{M}}}, \rho(\mathbf{s}) = d_f\}$. We found that $d^{\pi}_{\widehat{\mathcal{M}}}$ works better the hopper task in D4RL while d_f is better for the rest of the environments. For the remaining domains, we found $\rho(\mathbf{s}) = d_f$ works well.
- **Choice of $\mu(\mathbf{a}|\mathbf{s})$.** For the rollout policy μ, we use our automatic selection rule on the set $\{\text{Unif}(\mathbf{a}), \pi(\mathbf{a}|\mathbf{s})\}$, i.e. the set that contains a random policy and a current learned policy. We found that $\mu(\mathbf{a}|\mathbf{s}) = \text{Unif}(\mathbf{a})$ works well on the hopper task in D4RL and also in the `ant-angle` generalization experiment. For the remaining state-based environments, we discovered that $\mu(\mathbf{a}|\mathbf{s}) = \pi(\mathbf{a}|\mathbf{s})$ excels. In the image-based domain, we found that $\mu(\mathbf{a}|\mathbf{s}) = \text{Unif}(\mathbf{a})$ works well in the `walker-walk` domain and $\mu(\mathbf{a}|\mathbf{s}) = \pi(\mathbf{a}|\mathbf{s})$ is better for the `sawyer-door` environment. We observed that $\mu(\mathbf{a}|\mathbf{s}) = \text{Unif}(\mathbf{a})$ behaves less conservatively and is suitable to tasks where dynamics models can be learned fairly precisely.
- **Choice of f.** For the ratio between model rollouts and offline data f, we input the set $\{0.5, 0.8\}$ to our automatic hyperparameter selection rule to figure out the best f on each domain. We found that $f = 0.8$ works well on the medium and medium-expert in the walker2d task in D4RL. For the remaining environments, we find $f = 0.5$ works well.

We also provide additional experimental results on how our automatic hyperparameter selection rule selects hyperparameters. As shown in Table 4, 5, 6 and 7, our automatic hyperparameter selection rule is able to pick the hyperparameters β, $\mu(\mathbf{a}|\mathbf{s})$, $\rho(\mathbf{s})$ and f and that correspond to the best policy performance based on the regularization value.

Task	$\beta = 0.5$ performance	$\beta = 0.5$ regularizer value	$\beta = 5.0$ performance	$\beta = 5.0$ regularizer value
halfcheetah-medium	**54.2**	**-778.6**	40.8	-236.8
halfcheetah-medium-replay	**55.1**	**28.9**	9.3	283.9
halfcheetah-medium-expert	89.4	189.8	**90.0**	**6.5**
hopper-medium	75.0	-740.7	**97.2**	**-2035.9**
hopper-medium-replay	**89.5**	**37.7**	28.3	107.2
hopper-medium-expert	**111.1**	**-705.6**	75.3	-64.1
walker2d-medium	1.9	51.5	**81.9**	**-1991.2**
walker2d-medium-replay	**56.0**	**-157.9**	27.0	53.6
walker2d-medium-expert	10.3	-788.3	**103.3**	**-3891.4**

Table 4: We include our automatic hyperparameter selection rule of β on a set of representative D4RL environments. We show the policy performance (bold with the higher number) and the regularizer value (bold with the lower number). Lower regularizer value consistently corresponds to the higher policy return, suggesting the effectiveness of our automatic selection rule.

B.3 Details of generalization environments

For `halfcheetah-jump` and `ant-angle`, we follow the same environment used in MOPO. For `sawyer-door-close`, we train the `sawyer-door` environment in https://github.com/rlworkgroup/metaworld with dense rewards for opening the door until convergence. We collect 50000 transitions with half of the data collected by the final expert policy and a policy that reaches the performance of about half the expert level performance. We relabel the reward such that

Task	$\mu(\mathbf{a}\|\mathbf{s}) = \text{Unif}(\mathbf{a})$ performance	$\mu(\mathbf{a}\|\mathbf{s}) = \text{Unif}(\mathbf{a})$ regularizer value	$\mu(\mathbf{a}\|\mathbf{s}) = \pi(\mathbf{a}\|\mathbf{s})$ performance	$\mu(\mathbf{a}\|\mathbf{s}) = \pi(\mathbf{a}\|\mathbf{s})$ regularizer value
hopper-medium	**97.2**	**-2035.9**	52.6	-14.9
walker2d-medium	7.9	-106.8	**81.9**	**-1991.2**

Table 5: We include our automatic hyperparameter selection rule of $\mu(\mathbf{a}|\mathbf{s})$ on the medium datasets in the hopper and walker2d environments from D4RL. We follow the same convention defined in Table 4 and find that our automatic selection rule can effectively select μ offline.

Task	$\rho(\mathbf{s}) = d^{\pi}_{\hat{\mathcal{M}}}$ performance	$\rho(\mathbf{s}) = d^{\pi}_{\hat{\mathcal{M}}}$ regularizer value	$\rho(\mathbf{s}) = d_f$ performance	$\rho(\mathbf{s}) = d_f$ regularizer value
hopper-medium	**97.2**	**-2035.9**	56.0	-6.0
walker2d-medium	1.8	14617.4	**81.9**	**-1991.2**

Table 6: We include our automatic hyperparameter selection rule of $\rho(\mathbf{s})$ on the medium datasets in the hopper and walker2d environments from D4RL. We follow the same convention defined in Table 4 and find that our automatic selection rule can effectively select ρ offline.

the reward is 1 when the door is fully closed and 0 otherwise. Hence, the offline RL agent is required to learn the behavior that is different from the behavior policy in a sparse reward setting. We provide the datasets in the following anonymous link[1].

B.4 Details of image-based environments

Figure 3: Our image-based environments: The observations are 64×64 and 128×128 raw RGB images for the `walker-walk` and `sawyer-door` tasks respectively. The `sawyer-door-close` environment used in in Section 5.1 also uses the `sawyer-door` environment.

We visualize our image-based environments in Figure 3. We use the standard `walker-walk` environment from Tassa et al. [61] with 64×64 pixel observations and an action repeat of 2. Datasets were constructed the same way as Fu et al. [12] with 200 trajectories each. For the `sawyer-door` we use 128×128 pixel observations. The medium-expert dataset contains 1000 rollouts (with a rollout length of 50 steps) covering the state distribution from grasping the door handle to opening the door. The expert dataset contains 1000 trajectories samples from a fully trained (stochastic) policy. The data was obtained from the training process of a stochastic SAC policy using dense reward function as defined in Yu et al. [66]. However, we relabel the rewards, so an agent receives a reward of 1 when the door is fully open and 0 otherwise. This aims to evaluate offline-RL performance in a sparse-reward setting. All the datasets are from [48].

B.5 Computation Complexity

For the D4RL and generalization experiments, COMBO is trained on a single NVIDIA GeForce RTX 2080 Ti for one day. For the image-based experiments, we utilized a single NVIDIA GeForce RTX 2070. We trained the `walker-walk` tasks for a day and the `sawyer-door-open` tasks for about two days.

B.6 License of datasets

We acknowledge that all datasets used in this paper use the MIT license.

[1]The datasets of the generalization environments are available at the anonymous link: https://drive.google.com/file/d/1pn6dS5OgPQVp_ivGws-tmWdZoU7m_LvC/view?usp=sharing.

Task	$f = 0.5$ performance	$f = 0.5$ regularizer value	$f = 0.8$ performance	$f = 0.8$ regularizer value
hopper-medium	**97.2**	**-2035.9**	93.8	-21.3
walker2d-medium	70.9	-1707.0	**81.9**	**-1991.2**

Table 7: We include our automatic hyperparameter selection rule of f on the medium datasets in the hopper and walker2d environments from D4RL. We follow the same convention defined in Table 4 and find that our automatic selection rule can effectively select f offline.

Environment	**Batch Mean**	**Batch Max**	**COMBO (Ours)**	**CQL+MBPO**
halfcheetah-jump	-1022.6	1808.6	**5392.7**$\pm$575.5	4053.4$\pm$176.9
ant-angle	866.7	2311.9	**2764.8**$\pm$43.6	809.2$\pm$135.4
sawyer-door-close	5%	100%	**100**%$\pm$0.0%	62.7%$\pm$24.8%

Table 8: Comparison between COMBO and CQL+MBPO on tasks that require out-of-distribution generalization. Results are in average returns of `halfcheetah-jump` and `ant-angle` and average success rate of `sawyer-door-close`. All results are averaged over 6 random seeds, $\pm$ the 95%-confidence interval.

C Comparison to the Naive Combination of CQL and MBPO

In this section, we stress the distinction between COMBO and a direct combination of two previous methods CQL and MBPO (denoted as CQL + MBPO). CQL+MBPO performs Q-value regularization using CQL while expanding the offline data with MBPO-style model rollouts. While COMBO utilizes Q-value regularization similar to CQL, the effect is very different. CQL only penalizes the Q-value on unseen actions on the states observed in the dataset whereas COMBO penalizes Q-values on states generated by the learned model while maximizing Q values on state-action tuples in the dataset. Additionally, COMBO also utilizes MBPO-style model rollouts for also augmenting samples for training Q-functions.

To empirically demonstrate the consequences of this distinction, CQL + MBPO performs quite a bit worse than COMBO on generalization experiments (Section 5.1) as shown in Table 8. The results are averaged across 6 random seeds ($\pm$ denotes 95%-confidence interval of the various runs). This suggests that carefully considering the state distribution, as done in COMBO, is crucial.