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]University of Washington
`tianheyu@cs.stanford.edu` & `aviralk@berkeley.edu`

ABSTRACT

Model-based algorithms, which learn a dynamics model from logged experience and perform some sort of pessimistic 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 pessimism. Uncertainty estimation with complex models, such as deep neural networks, can be difficult and unreliable. We overcome this limitation by developing a new model-based offline RL algorithm, COMBO, that regularizes the value function on out-of-support state-action tuples generated via rollouts under the learned model. This results in a conservative estimate of the value function for out-of-support state-action tuples, without requiring explicit uncertainty estimation. We theoretically show that our method optimizes a lower bound on the true policy value, that this bound is tighter than that of prior methods, and our approach satisfies a policy improvement guarantee in the offline setting. Through experiments, we find that COMBO consistently performs as well or better as compared to prior offline model-free and model-based methods on widely studied offline RL benchmarks, including image-based tasks.

1 INTRODUCTION

Offline reinforcement learning (offline RL) (Lange et al., 2012; Levine et al., 2020) 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 Wang et al. (2018), autonomous driving Yu et al. (2020a), robotics Kalashnikov et al. (2018); Rafailov et al. (2020), and personalized recommendation systems Swaminathan & Joachims (2015). Recent studies have observed that direct use of RL algorithms originally developed for the online or interactive paradigm leads to poor results in the offline RL setting Fujimoto et al. (2018a); Kumar et al. (2019); Kidambi et al. (2020). 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 datasets available in the aforementioned application domains. In this work, we develop a new model-based offline RL algorithm that enjoys strong theoretical guarantees, while also matching or improving over state-of-the-art methods in offline RL benchmarks.

One paradigm for algorithm design in offline RL is to incorporate conservatism or regularization to the learning algorithm. Model-free offline RL algorithms Fujimoto et al. (2018b); Kumar et al. (2019); Wu et al. (2019); Jaques et al. (2019); Kumar et al. (2020) 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 Kidambi et al. (2020); Yu et al. (2020c) learn a pessimistic dynamics model, which in turn induces a conservative estimate of the value function. By generating data, model-based algorithms can achieve better generalization and e.g. have demonstrated the ability to solve new tasks using the offline dataset Yu et al. (2020c). However, such algorithms rely crucially on uncertainty quantification of the learned dynamics model to incorporate conservatism, which can be difficult or unreliable for complex datasets or deep network models. Furthermore, these

[∗]denotes equal contribution.

Figure 1: COMBO learns a conservative value function by utilizing both the of ine dataset as well as simulated data from the model. Crucially, COMBO does not require uncertainty quanti cation, and the value function learned by COMBO is a tighter lower-bound of the true value compared to CQL. This enables COMBO to steer the agent towards higher value states compared to CQL, which may steer towards sub-optimal states as illustrated in the gure.

methods do not adapt the uncertainty estimates as the policy and value function change over the course of learning. In this work, our goal is to develop a new algorithm that retains the bene ts of model-based algorithms while removing the reliance on uncertainty estimation, which we argue is not necessary for of ine RL.

Our main contribution is the development of conservative of ine model-based policy optimization (COMBO), a new model-based algorithm for of ine RL. COMBO learns a dynamics model using the of ine dataset. Subsequently, it employs an actor-critic method where the value function is learned using both the of ine dataset as well as synthetically generated data from the model, similar to Dyna Sutton (1991). 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 of ine dataset, obtained by simulating the learned model. We theoretically show that for any policy, the Q-function learned with COMBO is a lower bound of the true Q-function, making it a good surrogate for policy optimization. While the approach of optimizing a performance lower-bound is similar in spirit to prior model-based algorithms Kidambi et al. (2020); Yu et al. (2020c), COMBO crucially does not require uncertainty quanti cation. In addition, we show theoretically that the Q-function learned by COMBO represents a tighter lower bound of the true Q-function when the model bias is low compared to prior model-free algorithms like CQL Kumar et al. (2020). Thus, as a consequence of optimizing a tighter lower bound, COMBO has the potential to learn higher rewarding policies compared to prior model-free algorithms. This is illustrated through an example in Figure 1. Finally, in our experiments, we nd that COMBO matches or exceeds the state-of-the-art results in commonly studied benchmark tasks for of ine RL. Speci cally, COMBO achieves the highest score in 9 out of 12 continuous control domains we consider from the D4RL Fu et al. (2020) benchmark suite, while the next best algorithm achieves the highest score in only 3 out of the 12 domains. We also show that COMBO achieves the best performance in tasks that require out-of-distribution generalization and outperforms previous latent-space of ine model-based RL methods in the image-based robotic manipulation task.

2 PRELIMINARIES

2.1 MARKOV DECISION PROCESSES AND OFFLINE RL

We study RL in the framework of Markov decision processes (MDPs) speci ed by the tuple $M = (S; A; T; r; {}_0;)$. $S; A$ denote the state and action spaces. $T(s^0|s; a)$ and $r(s; a) 2 [R_{max}; R_{max}]$ represent the dynamics and reward function respectively. ${}_0(s)$ denotes the initial state distribution, and $2 (0; 1)$ denotes the discount factor. We denote the discounted state visitation distribution of a policy using $d_M (s) := (1 \quad) \sum_{t=0}^{1} {}^t P(s_t = s| \quad)$, where $P(s_t = s| \quad)$ is the probability of reaching state s at time t by rolling out in M . Similarly, we denote the state-action visitation distribution with $d_M (s; a) := d_M (s) \ (a|s)$. The goal of RL is to learn a policy that maximizes the return, or long term cumulative rewards:

$$\max J(M\ ;\ ) := \frac{1}{1\ } E_{(s;a)\ \ d_M\ (s;a)}[r(s;a)]: \tag{1}$$

Of ine RL is the setting where we have access only to a xed dataset $D = f(s;a;r;s^0)g$, which consists of transition tuples from trajectories collected using a behavior policy. In other words, the dataset D is sampled from $d\ \ (s;a) := d\ \ (s)\ \ (a j s)$. We de ne $\overline{M}$ as the empirical MDP induced by the dataset D and $d(s;a)$ as sampled-based version of $d\ (s;a)$. In the of ine setting, even in the limit of an in nite size dataset, it may not be possible to nd the optimal policy for the underlying MDP Chen & Jiang (2019); Kidambi et al. (2020). Thus, we typically forgo the goal of nding the optimal policy, and instead aim to nd the best possible policy using the xed of ine dataset.

2.2 MODEL-FREE OFFLINE RL ALGORITHMS

One class of approaches for solving MDPs involves the use of dynamic programming and actor-critic schemes Sutton & Barto (1998); Bertsekas & Tsitsiklis (1996), which do not explicitly require the learning of a dynamics model. To capture the long term behavior of a policy without a model, we de ne the action value function as

$$Q\ \ (s;a) := E\ \ "\sum_{t=0}^{1}\ \ ^t r(s_t;a_t)\ j\ s_0 = s; a_0 = a\ \ \# ; \tag{2}$$

where future actions are sampled from $\ (j s)$ and state transitions happen according to the MDP dynamics. Consider the following Bellman operator:

$$B\ \ Q(s;a) := r(s;a) + \ \ E_{s^0\ \ T(\ j s;a);a^0\ \ \ (\ j s^0)}[Q(s^0;a^0)];$$

and its sample based counterpart:

$$\hat{B}\ \ Q(s;a) := r(s;a) + \ \ Q(s^0;a^0);$$

associated with a single transition $(s;a;s^0)$ and $a^0\ \ \ (\ j s^0)$. The action-value function satis es the Bellman consistency criterion given by $B\ \ Q\ \ (s;a) = Q\ \ (s;a)\ 8(s;a)$. When given an of ine dataset D, standard approximate dynamic programming (ADP) and actor-critic methods use this criterion to alternate between policy evaluation Munos & Szepesvari (2008) and policy improvement. A number of prior works have observed that such a direct extension of ADP and actor-critic schemes to of ine RL leads to poor results due to distribution shift over the course of learning and over-estimation bias in the Q function Fujimoto et al. (2018a); Kumar et al. (2019); Wu et al. (2019). To address these drawbacks, prior works have proposed a number of modi cations aimed towards regularizing the policy or value function (see Section 5). In this work, we primarily focus on CQL Kumar et al. (2020), which alternates between the two following steps.

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

$$Q^{k+1}\ \ \arg\min_{Q}\ \ E_{s\ D\ \ ;a\ \ \ (\ j s)}[Q(s;a)]\ \ E_{s;a\ D}[Q(s;a)]$$
$$+\frac{1}{2}E_{s;a;s^0\ D}\ \ Q(s;a)\ \ \hat{B}\ \ Q^k(s;a))\ \ ^2\ ; \tag{3}$$

where $\ (\ j s)$ 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 Kumar et al. (2020), mitigating the challenges of over-estimation bias and distribution shift.

Policy Improvement: After approximating the Q function as $\hat{Q}$, the policy is improved as

$$\arg\max_{0} E_{s\ D\ \ ;a\ \ \ ^0(\ j s)}\ h\ \hat{Q}\ \ (s;a)\ i :$$

Actor-critic schemes with parameterized policies and Q functions approximate the arg max and arg min in above equations with a few steps of gradient descent.

2.3 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 D, a dynamics model $\widehat{T}$ is typically trained using maximum likelihood estimation as: $\min_{\widehat{T}} \ E_{(s;a;s^0)\ D} \quad \log \widehat{T}(s^0 j s; a)$. A reward model $\hat{r}(s; a)$ can also be learned similarly if it is unknown. Once a model has been learned, we can construct the learned MDP $\widehat{M} = (S; A; \widehat{T}; \hat{r}; \ _0; \ )$, 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 $\hat{\ } = \arg\max \ J(\widehat{M}; \ ):$

This straightforward approach is known to fail in the of ine RL setting, both in theory and practice, due to distribution shift and model-bias Ross & Bagnell (2012); Kidambi et al. (2020). In order to overcome these challenges, of ine model-based algorithms like MOReL Kidambi et al. (2020) and MOPO Yu et al. (2020c) use uncertainty quanti cation to construct a lower bound for policy performance and optimize this lower bound. By using an uncertainty estimation algorithm like bootstrap ensembles Osband et al. (2018); Azizzadenesheli et al. (2018); Lowrey et al. (2019), we can obtain $u(s; a)$, an estimate of uncertainty in dynamics model prediction. In the case of MOPO, an uncertainty penalized MDP is constructed where the reward is given by $\tilde{r}(s; a) = \hat{r}(s; a) \quad u(s; a)$, and the learned dynamics model is used without modi cation. MOPO learns a policy in this "uncertainty-penalized" MDP $\widetilde{M} = (S; A; \widehat{T}; \tilde{r}; \ _0; \ )$ which has the property that $J(\widetilde{M}; \ ) \quad J(M; \ )\ 8$. By constructing and optimizing such a lower bound, of ine 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 $\widetilde{M}$, we focus speci cally on MBPO Janner et al. (2019); Sutton (1991) which was used in MOPO. MBPO follows the standard structure of actor-critic algorithms, but in each iteration uses an augmented dataset $D [D_{\text{model}}$ for policy evaluation. Here, D is the of ine dataset and D_{model} is a dataset obtained by simulating the current policy using the learned dynamics model. Speci cally, at each iteration, MBPO performs k-step rollouts using $\widehat{T}$ starting from state $s\ 2\ D$ with a particular rollout policy $(a j s)$, adds the model-generated data to D_{model}, and optimizes the policy with a batch of data sampled from $D [D_{\text{model}}$ where each datapoint in the batch is drawn from D with probability $f\ 2\ [0; 1]$ and D_{model} with probability $1 \quad f$.

3 Conservative Offline Model-Based Policy Optimization

The principal challenge in practice with prior of ine model-based algorithms (discussed in Section 2) is the strong reliance on uncertainty quanti cation, which can be challenging for complex datasets or deep neural network models Ovadia et al. (2019). We also empirically verify this in the of ine dynamics model learning setting in Appendix E. Our goal is to develop a model-based of ine RL algorithm that enables optimizing a lower bound on the policy performance, but without requiring uncertainty quanti cation. We achieve this by extending conservative Q-learning Kumar et al. (2020), which does not require explicit uncertainty quanti cation, into the model-based setting. Our algorithm, summarized in Algorithm 1 in Appendix A, alternates between a conservative policy evaluation step and a policy improvement step. We also provide theoretical analysis of COMBO in Appendix B.

Conservative Policy Evaluation: Given a policy , an of ine dataset D, and a learned model of the MDP $\widehat{M}$, the goal in this step is to obtain a conservative estimate of Q . 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:

$$\hat{Q}^{k+1} \quad \arg\min_{Q} \quad E_{s;a \quad (s;a)}[Q(s; a)] \quad E_{s;a\ D}\ [Q(s; a)] + \frac{1}{2} E_{s;a;s^0 \ d_f} \quad Q(s; a) \quad \widehat{B}\ \hat{Q}^k(s; a)) \ ^2 : \tag{4}$$

Here, $(s; a)$ and d_f are sampling distributions that we can choose. Model-based algorithms allow ample exibility for these choices while providing the ability to control the bias introduced by these choices. For $(s; a)$, we make the following choice:

$$(s; a) = d_{\widehat{M}}\ (s) \ (a j s);$$

where $d_{\widehat{M}}(s)$ is the discounted marginal state distribution when executing in the learned model $\widehat{M}$. Samples from $d_{\widehat{M}}(s)$ can be obtained by rolling out in $\widehat{M}$. Similarly, d_f is an f interpolation between the of ine dataset and synthetic rollouts from the model:

$$d_f(s;a) := f\, d(s;a) + (1 \quad f)\, d_{\widehat{M}}(s;a);$$

where $f \; 2 \; [0;1]$ is the ratio of the datapoints drawn from the of ine dataset as de ned in Section 2.3 and $(\,|s)$ is the rollout distribution used with the model, which can be modeled as or a uniform distribution. To avoid notation clutter, we also denote $d_f := d_f$.

Under such choices of 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 of ine 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 Sutton (1991). 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 of ine dataset. This is made possible by performing rollouts under the learned dynamics model, denoted by $d_{\widehat{M}}(s;a)$. We will show in Appendix B that the Q function learned by repeating the recursion in Equation 4 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 $\hat{Q}$, we improve the policy as:

$$^{0} \quad \arg\max \; \mathbb{E}_{s \quad ;\, a \quad (\,|s)} \left[\hat{Q} \;(s;a) \right] \tag{5}$$

where (s) is the state marginal of $(s;a)$. 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 Haarnoja et al. (2018). In Appendix B.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 , that produces a Gaussian distribution over the next state and reward: $\hat{P}\,(s_{t+1}; r|s;a) = N(\;(s_t;a_t);\;(s_t;a_t))$. The model is trained via maximum likelihood. For training the conservative critic, which is the major distinction between COMBO and MOPO, the xed constant is tuned with an of ine cross-validation scheme for all low-dimensional continuous control tasks and is decided with a limited number of rollouts in the actual environment in the vision-based environments. We set the ratio $f = 0{:}5$ to have an equal split between model rollouts and data from the of ine dataset. For conservative policy evaluation (eq. 4) and policy improvement (eq. 5), we augment with states sampled from the of ine dataset, which shows more stable improvement in practice. Additional details about practical implementation are provided in Appendix D.1.

4 EXPERIMENTS

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

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 (Kumar et al., 2019), BRAC (Wu et al., 2019), and CQL (Kumar et al., 2020); as well as MOPO (Yu et al., 2020c) which is a model-based algorithm. In addition, we also compare with an of ine version of SAC (Haarnoja et al., 2018) (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 (Rafailov et al., 2020), which is a latent space of ine model-based RL method that handles image inputs, latent space MBPO (denoted

Table 1: Results for D4RL datasets. Each number is the normalized score proposed in Fu et al. (2020) of the policy at the last iteration of training, averaged over 3 random seeds. We take the results of MOPO and CQL from their original papers and results of the other model-free methods from the D4RL paper (Fu et al., 2020). We include the performance of behavior cloning (BC) from the of ine dataset for comparison. We bold the highest score across all methods.

Dataset type	Environment	BC	COMBO (ours)	MOPO	CQL	SAC-off	BEAR	BRAC-p	BRAC-v
random	halfcheetah	2.1	38.8	35.4	35.4	30.5	25.1	24.1	31.2
random	hopper	1.6	17.9	11.7	10.8	11.3	11.4	11.0	12.2
random	walker2d	9.8	7.0	13.6	7.0	4.1	7.3	-0.2	1.9
medium	halfcheetah	36.1	54.2	42.3	44.4	-4.3	41.7	43.8	46.3
medium	hopper	29.0	94.9	28.0	86.6	0.8	52.1	32.7	31.1
medium	walker2d	6.6	75.5	17.8	74.5	0.9	59.1	77.5	81.1
medium-replay	halfcheetah	38.4	55.1	53.1	46.2	-2.4	38.6	45.4	47.7
medium-replay	hopper	11.8	73.1	67.5	48.6	3.5	33.7	0.6	0.6
medium-replay	walker2d	11.3	56.0	39.0	32.6	1.9	19.2	-0.3	0.9
med-expert	halfcheetah	35.8	90.0	63.3	62.4	1.8	53.4	44.2	41.9
med-expert	hopper	111.9	111.1	23.7	111.0	1.6	96.3	1.9	0.8
med-expert	walker2d	6.4	96.1	44.6	98.7	-0.1	40.1	76.9	81.6

LMBPO), similar to Janner et al. (2019) which uses the model to generate additional synthetic data, the fully of ine version of SLAC (Lee et al., 2020) (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 of ine RL methods. Hence we choose them as comparisons to COMBO. For more details of our experimental set-up, comparisons, and hyperparameters, see Appendix D.

4.1 RESULTS ON THE D4RL BENCHMARK

To answer the question (1), we evaluate COMBO on the OpenAI Gym (Brockman et al., 2016) domains in the D4RL benchmark (Fu et al., 2020), 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 1. The numbers of BC, SAC-off, BEAR, BRAC-P and BRAC-v are taken from the D4RL paper, while the results for MOPO and CQL are based on their respective papers (Yu et al., 2020c; Kumar et al., 2020). COMBO achieves the best performance in 9 out of 12 settings while attaining similar performance to the best-performing method in the remaining 3 settings. As noted by Yu et al. (2020c) and Rafailov et al. (2020), model-based of ine 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 outperforms or performs comparably to the best method among existing model-free and model-based approaches, suggesting that COMBO is robust to different dataset types. Such results can be explained by COMBO being less conservative compared to prior model-free of ine methods and enjoying lower worst-case suboptimality when the learned model is inaccurate compared to previous model-based of ine approaches as shown in Appendix B. COMBO also does not rely on the heuristics of uncertainty estimation as in prior model-based of ine RL methods, which also potentially leads to COMBO's superior performance in various dataset types since uncertainty estimation is particularly challenging in settings where the learned model is not precise. We also empirically show that the heuristics of uncertainty estimation used in prior model-based of ine RL methods are inaccurate on the medium datasets in D4RL in Appendix E and might be the major reason of the poor results of prior model-based approaches on those datasets, which further corroborates the importance of removing uncertainty estimation in model-based of ine RL.

4.2 RESULTS ON TASKS THAT REQUIRE GENERALIZATION

To answer question (2), we use the two environments halfcheetah-jump and ant-angle constructed in Yu et al. (2020c), which requires the agent to solve a task that is different from what the behavior policy solved. In both environments, the of ine dataset is collected by policies trained with the original reward functions of halfcheetah and ant, which reward the halfcheetah and the ant 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 of ine dataset. Following Yu et al. (2020c), we relabel the rewards in the of ine 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

Table 2: 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 3 random seeds. We include the mean and max undiscounted return / success rate of the episodes in the batch data (under Batch Mean and Batch Max, respectively) for comparison.

Environment	Batch Mean	Batch Max	COMBO (Ours)	MOPO	CQL
halfcheetah-jump	-1022.6	1808.6	5392.7	4016.6	741.1
ant-angle	866.7	2311.9	2764.8	2530.9	2473.4
sawyer-door-close	5%	100%	100%	65.8%	36.7%

task sawyer-door-close based on the environment in Yu et al. (2020b); Rafailov et al. (2020). In this task, we collect the of ine 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 of ine 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 of ine RL methods must generalize beyond behaviors in the of ine data in order to learn the intended behaviors. We visualize the sawyer-door-close environment in the right image in Figure 2.

We present the results on the three tasks in Table 2. COMBO outperforms MOPO and CQL, two representative model-based and model-free methods respectively, in the halfcheetah-jump and sawyer-door-close tasks, and achieves an approximately 8% and 12% improvement over MOPO and CQL respectively on the ant-angle task. These results validate that COMBO achieves better generalization by behaving less conservatively than prior model-free of ine methods and more robustly than prior model-based of ine methods, as shown theoretically in Appendix B.

4.3 Results on image-based tasks

To answer question (3), we evaluate COMBO on two image-based environments: the standard walker (walker-walk) task from the the DeepMind Control suite Tassa et al. (2018) and a visual door opening environment with a Sawyer robotic arm (sawyer-door) as used in Section 4.2. 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. (2020), 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 2. To extend COMBO to the image-based setting, we follow Rafailov et al. (2020) and train a recurrent variational model using the of ine data and use train COMBO in the latent space of this model.

Figure 2: Visualization of our image-based environments. The observations are 64 64 and 128 128 raw pixel images for the walker-walk and sawyer-door tasks respectively. The sawyer-door-close environment used in our generalization experiments in Section 4.2 also uses the same environment as sawyer-door.

We present results in Table 3. 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 Related Work

Of ine RL (Ernst et al., 2005; Riedmiller, 2005; Lange et al., 2012; Levine et al., 2020) 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 (Kalashnikov et al., 2018; Mandlekar et al., 2020; Rafailov et al., 2020; Singh et al., 2020), NLP (Jaques et al., 2019; 2020) and healthcare (Shortreed et al., 2011; Wang et al., 2018). Similar to interactive RL, both model-free and model-based

Table 3: Results for vision experiments. For the Walker task each number is the normalized score proposed in Fu et al. (2020) 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	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%

algorithms have been studied for of ine RL, with explicit or implicit regularization of the learning algorithm playing a major role.

Model-free of ine RL. Prior model-free of ine 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 (Precup et al., 2001; Sutton et al., 2016; Liu et al., 2019; Swaminathan & Joachims, 2015; Nachum et al., 2019), of ine actor-critic methods (Siegel et al., 2020; Peng et al., 2019), applying uncertainty quanti cation to the predictions of the Q-values (Agarwal et al., 2020; Kumar et al., 2019; Wu et al., 2019; Levine et al., 2020), and learning conservative Q-values (Kumar et al., 2020) or explicitly measured by direct state or action constraints Fujimoto et al. (2018a); Liu et al. (2020), KL divergence (Jaques et al., 2019; Wu et al., 2019; Zhou et al., 2020), Wasserstein distance, and MMD (Kumar et al., 2019). Different from these works, COMBO uses both the of ine dataset as well as simulated data from a learned dynamics model. This allows COMBO to behave less conservatively by optimizing a tighter lower bound of policy performance, as well as gain broader generalization when the model bias is small, as demonstrated through our theoretical analysis and experiments.

Model-based of ine RL. Model-based of ine RL methods (Finn & Levine, 2017; Ebert et al., 2018; Kahn et al., 2018; Kidambi et al., 2020; Yu et al., 2020c; Matsushima et al., 2020; Argenson & Dulac-Arnold, 2020; Swazinna et al., 2020; Rafailov et al., 2020; Lee et al., 2021) 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 quanti cation of the learned dynamics model which can be dif cult for deep network models Ovadia et al. (2019), or on directly constraining the policy towards the behavioral policy similar to model-free algorithms Matsushima et al. (2020). 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 bene ts of model-based algorithms such as broad generalization, without the constraints of explicit policy regularization or uncertainty quanti cation.

6 CONCLUSION

In the paper, we present conservative of ine model-based policy optimization (COMBO), a model-based of ine RL algorithm that penalizes the Q-values evaluated on out-of-support state-action pairs. In particular, COMBO removes the need of uncertainty quanti cation as widely used in previous model-based of ine RL works (Kidambi et al., 2020; Yu et al., 2020c), which can be challenging and unreliable with deep neural networks (Ovadia et al., 2019). Theoretically, we show that COMBO achieves a tighter lower-bound on the true policy value compared to prior model-free of ine RL methods (Kumar et al., 2020) and guarantees a safe policy improvement. In our empirical study, COMBO outperforms prior model-based and model-free of ine RL methods in 9 out of 12 datasets in the standard D4RL benchmark and attains comparable performance in rest of 3 datasets. COMBO also achieves the best generalization performances in 3 tasks that require adaptation to unseen behaviors. Finally, COMBO is able scale to vision-based tasks and outperforms or obtain comparable results in vision-based locomotion and robotic manipulation tasks. Despite the advantages of COMBO, there are few challenges left such as the lack of a fully of ine hyperparameter tuning mechanism and an automatically tuned β that takes the model error into account. We leave them for future work.

REFERENCES

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

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on of ine reinforcement learning. In *International Conference on Machine Learning*, pp. 104–114. PMLR, 2020.

Arthur Argenson and Gabriel Dulac-Arnold. Model-based of ine planning. *arXiv preprint arXiv:2008.05556*, 2020.

Kamyar Azizzadenesheli, Emma Brunskill, and Animashree Anandkumar. Ef cient exploration through bayesian deep q-networks. In *ITA*, pp. 1–9. IEEE, 2018.

Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scienti c, Belmont, MA, 1996.

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

J. Chen and Nan Jiang. Information-theoretic considerations in batch reinforcement learning. *ArXiv*, abs/1905.00360, 2019.

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

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.

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

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

Justin Fu, Aviral Kumar, O r Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

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

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

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.

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.

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

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*, pp. 12498–12509, 2019.

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.

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

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

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*, pp. 651–673. PMLR, 2018.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based of ine reinforcement learning. *arXiv preprint arXiv:2005.05951*, 2020.

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*, pp. 11761–11771, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for of ine reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

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

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

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.

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

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

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

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

Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. Plan Online, Learn Of ine: Ef cient Learning and Exploration via Model-Based Control. In *International Conference on Learning Representations (ICLR)*, 2019.

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 of ine robot manipulation data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4414–4420. IEEE, 2020.

Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, O r Nachum, and Shixiang Gu. Deployment-ef cient reinforcement learning via model-based of ine optimization. *arXiv preprint arXiv:2006.03647*, 2020.

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

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

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

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

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.

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.

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

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

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

Martin Riedmiller. Neural tted q iteration– rst experiences with a data ef cient neural reinforcement learning method. In *European Conference on Machine Learning*, pp. 317–328. Springer, 2005.

Stephane Ross and Drew Bagnell. Agnostic system identi cation for model-based reinforcement learning. In *ICML*, 2012.

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.

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 of ine reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

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

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

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

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.

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

Phillip Swazinna, Steffen Udluft, and Thomas Runkler. Overcoming model bias for robust of ine deep reinforcement learning. *arXiv preprint arXiv:2008.05533*, 2020.

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.

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.

Yifan Wu, George Tucker, and O r Nachum. Behavior regularized of ine reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

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)*, pp. 2633–2642, 2020a.

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*, pp. 1094–1100. PMLR, 2020b.

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

Wenxuan Zhou, Sujay Bajracharya, and David Held. Plas: Latent action space for of ine reinforcement learning. *arXiv preprint arXiv:2011.07213*, 2020.

Algorithm 1 COMBO: Conservative Model Based Of ine Policy Optimization

```
Require: Of ine dataset D, rollout distribution ( |s), learned dynamics model $\hat{T}$ , initialized policy and critic
        and Q .
 1: Train the probabilistic dynamics model $\hat{T}(s^0; r|s; a)$ = N ( (s; a); (s; a)) on D.
 2: Initialize the replay buffer D_model ? .
 3: for i = 1; 2; 3; ; do
 4:     Perform model rollouts by drawing samples from and $\hat{T}$ starting from states in D. Add model rollouts
        to D_model.
 5:     Conservatively evaluate current policy by repeatedly solving eq. 4 to obtain $\hat{Q}^i$ using data sampled from
        D [ D_model.
 6:     Improve policy under state marginal of d_f by solving eq. 5 to obtain $^{i+1}$ .
 7: end for
```

Appendices

A Algorithm Overview

We present the COMBO algorithm in Algorithm 1.

B 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, the lower-bound of the expected return in the case of COMBO is a tighter lower bound compared to model-free counterparts.

B.1 COMBO Optimizes a Lower-Bound

We rst show that training the Q-function using Equation 4 obtains a Q-function such that the expected off-policy policy improvement objective (Degris et al., 2012) computed using this learned Q-function lower-bounds its actual value. We will reuse notation for d_f, and d from Section 2 and Section 3. Assuming that the Q-function is represented as a table, the Q-function found by approximate dynamic programming in iteration k, can be obtained by differentiating Equation 4 with respect to Q^k (see Appendix C for details):

$$\hat{Q}^{k+1}(s;a) = (\hat{B}\ Q^k)(s;a) \qquad \frac{(s;a) \quad d(s;a)}{d_f\,(s;a)}: \tag{6}$$

Equation 6 effectively applies a penalty that depends on the three distributions appearing in the COMBO critic training objective (Equation 4), of which and d_f are free variables that we choose in practice as discussed in Section 3. Before stating our main result, we will rst show that the penalty term in equation 6 is positive in expectation. Such a positive penalty is important to combat any overestimation that may arise as a result of using $\hat{B}$.

Lemma 1 (Interpolation Lemma) For any f 2 [0; 1], and any given (s; a) 2 $^{|S||A|}$, let d_f be an f-interpolation of and D, i.e., $d_f(s;a) := f d(s;a) + (1 \quad f)\ (s;a)$. For a given iteration k of Equation 6, de ne the expected penalty under (s; a) as:

$$(; f) := E \quad \frac{(s;a) \quad d(s;a)}{d_f\,(s;a)} \quad :$$

Then (; f) satis es, (1) (; f) 0; 8 ; f , (2) (; f) is monotonically increasing in f for a xed , and (3) (; f) = 0 iff 8 s; a; (s; a) = d(s; a) or f = 0.

The proof can be found in Appendix C.1. Lemma 1 characterizes $(s;a)$ for which COMBO (Equation 4) induces a conservative penalty. COMBO sets $(s) = d_{\hat{M}}(s)$ and uses $(a|s) = (a|s)$, and hence each step of update (Equation 6) penalizes the Q-function making it more conservative. The total amount of conservatism induced in the Q-function, given by (f), is controlled by the choice of f. Based on result (2) in Lemma 1, we note that by controlling the amount of real data, f, we can control the amount of conservatism: $f = 1$ induces the maximum conservatism, and $f = 0$ induces no conservatism at all.

Next, we will show that the asymptotic Q-function learned by COMBO lower-bounds the actual Q-function of any policy with high probability for a large enough 0. Let $\overline{M}$ 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 equation 4 can be interpreted as an interpolation of the backup operator in the empirical MDP (denoted by $B_{\overline{M}}$) and the backup operator under the learned model $\hat{M}$ (denoted by $B_{\hat{M}}$). 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 coef cient $C_{r;T;}$, dependent on the desired con dence value (details in Appendix C.2). This is a standard assumption in literature (Laroche et al., 2019). Now, we state our main results below.

Theorem 2 (Asymptotic lower-bound) Let P denote the Hadamard product of the dynamics P and a given policy in the actual MDP and let $S := (I \quad P)^{1} c_S$, where c_S is a suitably chosen positive constant. Let D denote the total-variation divergence between two probability distributions. For any $(a|s)$, the Q-function obtained by recursively applying Equation 6, with $\hat{B} = f B_{\overline{M}} + (1 \quad f) B_{\hat{M}}$, with probability at least 1, results in $\hat{Q}$ that satis es:

$$8s; a; \hat{Q}(s;a) \quad Q(s;a) \qquad {}_c + f \; {}_s + (1 \quad f)\; {}_m;$$

where, ${}_s$, ${}_c$ and ${}_m$ are given by:

$$ {}_c(s;a) := \frac{1}{c_S} S \quad \frac{d}{d_f} \quad (s;a); $$

$$ {}_m(s;a) := S \quad |R \quad R_{\hat{M}}| + \frac{2 R_{max}}{1} D(P; P_{\hat{M}}) \quad (s;a); $$

$$ {}_s(s;a) := S \quad \frac{C_{r;T;} \; R_{max}}{(1 \quad) \; p\overline{|D|}} \quad (s;a): $$

The proof for Theorem 2 can be found in Appendix C.2.

Corollary 3. For a suf ciently large , we have that $E_{s \; 0; a \; (|s)}[\hat{Q}(s;a)]$ $E_{s \; 0; a \; (|s)}[Q(s;a)]$, where ${}_0(s)$ is the initial state distribution. Furthermore, when ${}_s$ is small, such as in the large sample regime; or when the model bias ${}_m$ is small, a small is suf cient along with an appropriate choice of f.

Corollary 3 directly appeals to Theorem 2 and Lemma 1. Theorem 2 further implies that for large , deviating away from the behavior policy is expected to lead to smaller values. Finally, while Kumar et al. (2020) also analyze how regularized value function training can provide lower bounds on the value function at each state in the dataset (Kumar et al., 2020) (Theorem 3.1-3.2), our result goes further, showing that we can handle unseen states that were not seen in the dataset by virtue of using a learned dynamics model, and more importantly, attains a tighter lower bound: COMBO does not underestimate the value function at every state in the dataset like Kumar et al. (2020), but only lower-bounds the expected value function under the initial state distribution as discussed in Corollary 3. We elaborate on how COMBO is less conservative as it attains tighter lower bounds in Remark 8 (Appendix C.2).

B.2 SAFE POLICY IMPROVEMENT GUARANTEES

Now that we have shown that learned Q-function penalizes deviation from the behavior policy, we provide policy improvement guarantees for the COMBO algorithm. Formally, Theorem 4 discuss safe improvement guarantees over the behavior policy. building on prior work (Petrik et al., 2016; Laroche et al., 2019; Kumar et al., 2020).

Theorem 4 (-safe policy improvement). Let $\hat{\ }_{out}(a|s)$ be the policy obtained by COMBO. Then, the policy $\hat{\ }_{out}(a|s)$ is a -safe policy improvement over in the actual MDP M, i.e., $J(\hat{\ }_{out}; M) \ \ J(\ ; M) \ \ $, with probability at least $1 \ \ $, where is given by,

$$= O\left(\frac{f}{(1 \ \ )^2}\right) \underbrace{\mathbb{E}_{s \ \ d_M^{\hat{\ }_{out}}} \left[\sqrt{\frac{|A|}{|D(s)|} D_{CQL}(\hat{\ }_{out}; \ \ )} \right]}_{:= (1)}$$

$$+ \ O\left(\frac{(1 \ \ f)}{(1 \ \ )^2}\right) \underbrace{D_{TV}(\overline{M}; \hat{M})}_{:= (2)} \ \ \underbrace{\frac{(\ ; f) \ \ \ (\ ; f)}{(1 \ \ )}}_{:= (3)} .$$

The complete statement (with constants and terms that grow smaller than quadratic in the horizon) and proof for Theorem 4 is provided in Appendix C.3. D_{CQL} denotes a notion of probabilistic distance between policies (Kumar et al., 2020) which we discuss further in Appendix C.3. The expression for in Theorem 4 consists of three terms: term (1) captures the decrease in the policy performance due to limited data, and decays as the size of D increases. The second term (2) captures the suboptimality induced by the bias in the learned model. Finally, as we show in Appendix C.3, the third term (3) 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 D with transitions generated from the behavior policy , the marginal distribution $(s; a)$ is expected to be closer to $d(s; a)$ for as compared to the counterpart for the learned policy, . Thus, term (3) is expected to be positive in practice, and in such cases, an appropriate (large) choice of will make term (3) large enough to counteract terms (1) and (2) that reduce policy performance. We discuss this elaborately in Appendix C.3 (Remark 11).

Further note that in contrast to Theorem 3.6 in Kumar et al. (2020), note that our result indicates the sampling error (term (1)) is signi cantly reduced (multiplied by a fraction f) when a near-accurate model is used to augment data for training the Q-function, and in contrast to prior model-based of ine RL results Kidambi et al. (2020); Yu et al. (2020c), our bound implies that COMBO will incur lower worst-case suboptimality in cases when the learned dynamics model is inaccurate by biasing the backups towards the empirical MDP via f. To summarize, through an appropriate choice of f, Theorem 4 guarantees safe improvement over the behavior policy without requiring access to an oracle uncertainty estimation algorithm.

C PROOFS FROM APPENDIX B

In this section, we provide proofs for theoretical results in Appendix B. Before the proofs, we note that all statements are proven in the case of nite state space (i.e., $|S| < 1$) and nite action space (i.e., $|A| < 1$) we de ne some commonly appearing notation symbols appearing in the proof:

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

We also assume that whenever the cardinality of a particular state or state-action pair in the of ine dataset D, denoted by $|D(s; a)|$, appears in the denominator, we assume it is non-zero. For any non-existent $(s; a) \ 2 \ D$, we can simply set $|D(s; a)|$ to be a small value < 1, which prevents any bound from producing trivially 1 values.

C.1 PROOF OF LEMMA 1

Lemma 5 (Lemma 1 restated). For any $f \ 2 \ [0; 1]$, and any given $(s; a) \ 2 \ ^{|S||A|}$, let d_f be an f-interpolation of and D, i.e., $d_f(s; a) := f d(s; a) + (1 \ \ f) \ (s; a)$. For a given iteration k of Equation 6, de ne the expected penalty under $(s; a)$ as:

$$(\ ; f) := \mathbb{E}_{s;a \ \ (s;a)} \left[\frac{(s; a) \ \ \ d(s; a)}{d_f(s; a)} \right] .$$

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 $\forall\, s, a, \; \rho(s,a) = d(s,a)$ 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 $f \in [0, 1]$.

$$\nu(\rho; f) = \sum_{s,a} \rho(s,a) \left(\frac{\rho(s,a) - d(s,a)}{f d(s,a) + (1-f)\rho(s,a)} \right) = \sum_{s,a} \rho(s,a) \left(\frac{\rho(s,a) - d(s,a)}{\rho(s,a) + f(d(s,a) - \rho(s,a))} \right) \tag{7}$$

$$\implies \frac{d\nu(\rho; f)}{df} = \sum_{s,a} \rho(s,a) \left(\rho(s,a) - d(s,a)\right)^2 \left(\frac{1}{(\rho(s,a) + f(d(s,a) - \rho(s,a)))} \right)^2 \geq 0 \quad \forall f \in [0, 1]. \tag{8}$$

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:

$$\forall f \in (0, 1], \; \nu(\rho; f) \geq \nu(\rho; 0) = \sum_{s,a} \rho(s,a) \frac{\rho(s,a) - d(s,a)}{\rho(s,a)} = \sum_{s,a} \left(\rho(s,a) - d(s,a)\right) = 1 - 1 = 0. \tag{9}$$

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 $\rho(s,a) = d(s,a)$ note that we obtain $\nu(\rho; f) = 0$. To prove the only if side of (3), assume that $f \neq 0$ and $\rho(s,a) \neq d(s,a)$ and we will show that in this case $\nu(\rho; f) \neq 0$. When $d(s,a) \neq \rho(s,a)$, 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 $\rho(s,a) \neq d(s,a)$ and $f > 0$, $\nu(\rho; f) \neq 0$, which completes our proof for the only if side of (3). □

C.2 Proof of Theorem 2 and Corollary 3

Before proving this theorem, we provide a bound on the Bellman backup in the empirical MDP, $\mathcal{B}_{\overline{M}}$. To do so, we formally define the standard concentration properties of the reward and transition dynamics in the empirical MDP, $\overline{M}$, that we assume so as to prove Theorem 1. Following prior work (Osband & Van Roy, 2017; Jaksch et al., 2010; Kumar et al., 2020), we assume:

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

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

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

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

Thus the overestimation due to sampling error in the empirical MDP, $\overline{M}$ 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 Theorem 2, we assume that:

$$\forall s,a, \quad \left|\mathcal{B}_{\overline{M}}^{\pi} \hat{Q}^k - \mathcal{B}_{M}^{\pi} \hat{Q}^k\right| \le \frac{C_{r,T,\delta} R_{\max}}{(1-\gamma)\sqrt{|\mathcal{D}(s,a)|}}. \tag{10}$$

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

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

where $D(P, P_{\widehat{M}})$ is the total-variation divergence between the learned dynamics model and the actual MDP. Now, we will use Equations 10 and 11 to prove Theorem 2. We restate Theorem 2 for convenience.

Theorem 6 (Theorem 2 restated). *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 $\pi(a|s)$, the Q-function obtained by recursively applying Equation 6, with $\hat{\mathcal{B}}^{\pi} = f\mathcal{B}_{\overline{M}}^{\pi} + (1-f)\mathcal{B}_{\widehat{M}}^{\pi}$, with probability at least $1-\delta$, results in $\hat{Q}^{\pi}$ that satisfies:*

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

Proof. We first note that the Bellman backup $\hat{\mathcal{B}}^{\pi}$ induces the following Q-function iterates as per Equation 6,

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

$$\forall s,a,\ \hat{Q}^{k+1} \le \left(\mathcal{B}^{\pi}\hat{Q}^k\right) - \beta\frac{\rho - d}{d_f} + (1-f)\left(|r_{\widehat{M}} - r_M| + \frac{2\gamma R_{\max}}{1-\gamma} D(P, P_{\widehat{M}})\right) + f\frac{C_{r,T,\delta} R_{\max}}{(1-\gamma)\sqrt{|\mathcal{D}|}}$$

Since the RHS upper bounds the Q-function pointwise for each (s,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

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

which completes the proof of this theorem. □

Next, we use the result and proof technique from Theorem 2 to prove Corollary 3, that in expectation under the initial state-distribution, the expected Q-value is indeed a lower-bound.

Corollary 7 (Corollary 3 restated) For a suf ciently large , we have a lower-bound that $E_{s \quad 0; a \quad (| s)}[\hat{Q} \ (s;a)] \quad E_{s \quad 0; a \quad (| s)}[Q \ (s;a)]$, where $_0(s)$ is the initial state distribution. Furthermore, when $_s$ is small, such as in the large sample regime; or when the model bias $_M$ is small, a small is suf cient along with an appropriate choice of f.

Proof. To prove this corollary, we note a slightly different variant of Theorem 2. To observe this, we will deviate from the proof of Theorem 2 slightly and will aim to express the inequality using $B_{\hat{M}}$, the Bellman operator de ned by the learned model and the reward function. Denoting $(I \quad P_{\hat{M}})^{-1}$ as $S_{\hat{M}}$, doing this will intuitively allow us to obtain $(\ (s) \ (a|s))^T \ S_{\hat{M}} \ \frac{d}{d_f} \ (s;a)$ as the conservative penalty which can be controlled by choosing appropriately so as to nullify the potential overestimation caused due to other terms. Formally,

$$\hat{Q}^{k+1}(s;a) = \hat{B} \ \hat{Q}^k \ (s;a) \quad \frac{(s;a) \quad d(s;a)}{d_f \ (s;a)}$$

$$= B_{\hat{M}} \ \hat{Q}^k \ (s;a) \quad \frac{(s;a) \quad d(s;a)}{d_f \ (s;a)} + f \ \underbrace{B_{\overline{M}} \quad B_{\hat{M}} \ \hat{Q}^k \ (s;a)}_{:= (\ s;a)}$$

By controlling $(\ s;a)$ using the pointwise triangle inequality:

$$\forall s;a; \ B_{\overline{M}}\hat{Q}^k \ B_{\hat{M}} \ \hat{Q}^k \quad B \ \hat{Q}^k \ B_{\hat{M}} \ \hat{Q}^k + B_{\overline{M}}\hat{Q}^k \ B \ \hat{Q}^k ; \tag{12}$$

and then iterating the backup $B_{\hat{M}}$ to its xed point and nally noting that $(s;a) = (\quad)^T S_{\hat{M}} \quad (s;a)$, we obtain:

$$E_{;} \ [\hat{Q} \ (s;a)] \quad E_{;} \ [Q_{\hat{M}} \ (s;a)] \quad E_{(s;a)} \ \frac{(s;a) \quad d(s;a)}{d_f \ (s;a)} + \text{terms independent of} \ : \tag{13}$$

The terms marked as "terms independent of " correspond to the additional positive error terms obtained by iterating $B \ \hat{Q}^k \ B_{\hat{M}} \ \hat{Q}^k$ and $B_{\overline{M}}\hat{Q}^k \ B \ \hat{Q}^k$, which can be bounded similar to the proof of Theorem 2 above. Now by replacing the model Q-function, $E_{;} \ [Q_{\hat{M}} \ (s;a)]$ with the actual Q-function, $E_{;} \ [Q \ (s;a)]$ and adding an error term corresponding to model error to the bound, we obtain that:

$$E_{;} \ [\hat{Q} \ (s;a)] \quad E_{;} \ [Q \ (s;a)] + \text{terms independent of} \quad \underbrace{E_{(s;a)} \ \frac{(s;a) \quad d(s;a)}{d_f \ (s;a)}}_{= \ (\ ; f \ ) > 0} \ : \tag{14}$$

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

Remark 8 (COMBO attains a tighter lower-bound than 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 $s \in D$ individually (in the absence of no sampling error), i.e., $\forall s \in D; \hat{V}_{CQL}(s) \quad V \ (s)$, whereas the bound in COMBO is only valid in expectation of the value function over the initial state distribution, i.e., $E_{s \quad 0(s)}[\hat{V}_{COMBO}(s)] \quad E_{s \quad 0(s)}[V \ (s)]$, 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 $d(s;a)$ but not so frequent in the $(s;a)$ marginal distribution of the policy under the learned model $\hat{M}$. To see this more formally, note that the expected penalty added in the effective Bellman backup performed by COMBO (Equation 6), in expectation under the dataset distribution $d(s;a)$, $e(\ ; d; f \ )$ is actually negative:

$$e(\ ; d; f \ ) = \sum_{s;a} d(s;a) \frac{(s;a) \quad d(s;a)}{d_f \ (s;a)} = \sum_{s;a} d(s;a) \frac{d(s;a) \quad (s;a)}{f d \ (s;a) + (1 \quad f \ ) \ (s;a)} < 0;$$

where the nal inequality follows via a direct application of the proof of Lemma 1. Thus, COMBO actually overestimates the values at atleast some states (in the dataset) unlike CQL, making it a tighter lower bound.

C.3 PROOF OF THEOREM 4

To prove the policy improvement result in Theorem 4, we rst observe that using Equation 6 for Bellman backups amounts to nding a policy that maximizes the return of the policy in the a modi ed "f-interpolant" MDP which admits the Bellman backup $\hat{B}$, and is induced by a linear interpolation of backups in the empirical MDP $\overline{M}$ and the MDP induced by a dynamics model $\hat{M}$ and the return of a policy in this effective f-interpolant MDP is denoted by $J(\overline{M}; \hat{M}; f;)$. Alongside this, the return is penalized by the conservative penalty where denotes the marginal state-action distribution of policy in the learned model $\hat{M}$.

$$\hat{J}(f; \quad) = J(\overline{M}; \hat{M}; f; \quad) \qquad \frac{(\quad ; f)}{1}: \tag{15}$$

We will require bounds on the return of a policy in this f-interpolant MDP, $J(\overline{M}; \hat{M}; f; \quad)$, which we rst prove separately as Lemma 9 below and then move to the proof of Theorem 4.

Lemma 9 (Bound on return in f-interpolant MDP). For any two MDPs, M_1 and M_2, with the same state-space, action-space and discount factor, and for a given fraction $f \in [0; 1]$, de ne the f-interpolant MDP M_f as the MDP on the same state-space, action-space and with the same discount as the MDP with dynamics: $P_{M_f} := fP_{M_1} + (1 \quad f)P_{M_2}$ and reward function: $r_{M_f} := fr_{M_1} + (1 \quad f)r_{M_2}$. Then, given any auxiliary MDP, M, the return of any policy in M_f, $J(; M_f)$, also denoted by $J(M_1; M_2; f; \quad)$, lies in the interval:

$$J(; M) \quad ; \; J(; M) + \quad ; \qquad \text{where is given by:}$$

$$= \frac{2 (1 \quad f)}{(1 \quad)^2} R_{max} D(P_{M_2}; P_M) + \frac{f}{1} \; E_{d_M} \quad P_M \quad P_{M_1} \; Q_M \quad :$$
$$+ \frac{f}{1} E_{s;a \; d_M} \; [jr_{M_1}(s; a) \quad r_M(s; a)j] + \frac{1 \quad f}{1} E_{s;a \; d_M} \; [jr_{M_2}(s; a) \quad r_M(s; a)j]: \tag{16}$$

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

$$J(P; r_{M_f}) = J(P; fr_{M_1} + (1 \quad f)r_{M_2}) = fJ(P; r_{M_1}) + (1 \quad f)J(P; r_{M_2}):$$

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

$$\begin{aligned} J(P; r_{M_f}) &= J(P; fr_{M_1} + (1 \quad f)r_{M_2}) = J(P; r + f(r_{M_1} \quad r) + (1 \quad f)(r_{M_2} \quad r)) \\ &= J(P; r) + fJ(P; r_{M_1} \quad r) + (1 \quad f)J(P; r_{M_2} \quad r) \\ &= J(P; r) + \frac{f}{1} E_{s;a \; d_M(s) \; (ajs)} [r_{M_1}(s; a) \quad r(s; a)] \\ &\qquad + \frac{1 \quad f}{1} E_{s;a \; d_M(s) \; (ajs)} [r_{M_2}(s; a) \quad r(s; a)]: \end{aligned}$$

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

$$\begin{aligned} &J(P_{M_f}; r) = J(fP_{M_1} + (1 \quad f)P_{M_2}; r) \\ &= J(P_M + f(P_{M_1} \quad P_M) + (1 \quad f)(P_{M_2} \quad P_M); r) \\ &= J(P_M; r) \quad \frac{(1 \quad f)}{1} E_{s;a \; d_M(s) \; (ajs)} \quad P_{M_2} \quad P_M \quad Q_M \qquad \frac{f}{1} E_{s;a \; d_M(s) \; (ajs)} \quad P_M \quad P_{M_1} \; Q_M \\ &\; 2 \quad J(P_M; r) \qquad \frac{f}{(1 \quad)} \; E_{s;a \; d_M(s) \; (ajs)} \quad P_M \quad P_{M_1} \; Q_M \quad + \frac{2 (1 \quad f)R_{max}}{(1 \quad)^2} D(P_{M_2}; P_M) \quad : \end{aligned}$$

To observe the third equality, we utilize the result on the difference between returns of a policy π on two different MDPs, $P_{\overline{M}_1}$ and $P_{\overline{M}_f}$ from Agarwal et al. (2019) (Chapter 2, Lemma 2.2, Simulation Lemma), and additionally incorporate the auxiliary MDP $\overline{M}$ 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 $D(P_{\overline{M}_2}, P_{\overline{M}})$ and the other term corresponding to the empirical MDP $\overline{M}$ 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:

$$J(\overline{M}_1, \overline{M}_2, f, \pi) := J(P_{\overline{M}_f}, f r_{\overline{M}_1} + (1-f) r_{\overline{M}_2}) = J(f P_{\overline{M}_1} + (1-f) P_{\overline{M}_2}, r_{\overline{M}_f})$$

$$\in \left[J(P_{\overline{M}_f}, r_{M}) \pm \underbrace{\frac{f}{1-\gamma} \mathbb{E}_{s,a \sim d_{\overline{M}}} \left[|r_{\overline{M}_1}(s,a) - r_{M}(s,a)| \right] + \frac{1-f}{1-\gamma} \mathbb{E}_{s,a \sim d_{\overline{M}}} \left[|r_{\overline{M}_2}(s,a) - r_{M}(s,a)| \right]}_{:=\Delta_R} \right],$$

where the second step holds via linear decomposition of the return of π in $\overline{M}_f$ 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 $J(P_{\overline{M}_f}, r_M)$ in terms of the return on the actual MDP, $J(P_M, r_M)$, using the inequality proved above that provides intervals for mixture dynamics but a fixed reward function. Thus, the overall bound is given by $J(\pi, \overline{M}_f) \in [J(\pi, M) - \alpha, J(\pi, M) + \alpha]$, where α is given by:

$$\alpha = \frac{2\gamma(1-f)}{(1-\gamma)^2} R_{\max} D(P_{\overline{M}_2}, P_{M}) + \frac{\gamma f}{1-\gamma} \left| \mathbb{E}_{d^{\pi}_{M}} \left[\left(P^{\pi}_{M} - P^{\pi}_{\overline{M}_1} \right) Q^{\pi}_{M} \right] \right| + \Delta_R. \quad (17)$$

This concluded the proof of this lemma. □

Finally, we prove Theorem 4 that shows how policy optimization with respect to $\hat{J}(f, \pi)$ affects the performance in the actual MDP by using Equation 15 and building on the analysis of pure model-free algorithms from Kumar et al. (2020). We restate a more complete statement of the theorem below and present the constants at the end of the proof.

Theorem 10 (Formal version of Theorem 4). *Let $\hat{\pi}_{\text{out}}(a|s)$ be the policy obtained by COMBO. Then, the policy $\pi_{\text{out}}(a|s)$ is a ζ-safe policy improvement over π_β in the actual MDP M, i.e., $J(\pi_{\text{out}}, M) \geq J(\pi_\beta, M) - \zeta$, with probability at least $1-\delta$, where ζ is given by (where $\rho(s,a) := d^{\pi}_{\widehat{M}}(s,a)$):*

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

$$+ O\left(\frac{\gamma(1-f)}{(1-\gamma)^2}\right) D_{TV}(P_M, P_{\widehat{M}}) - \beta \frac{\nu(\rho^{\pi_{\text{out}}}, f) - \nu(\rho^{\beta}, f)}{(1-\gamma)}.$$

Proof. We first note that since policy improvement is not being performed in the same MDP, M as the f-interpolant MDP, 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 $J(\pi, M)$ and $J(\pi, M_f) := J(\overline{M}, \widehat{M}, f, \pi)$ 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 9 stated and proved previously, we obtain the following upper and lower-bounds on the return of a policy π:

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

where α is shown in Equation 16. 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 17), and (2) terms that depend on the dynamics (the other two terms in Equation 17).

To bound $_R$, we sipmply appeal to concentration inequalities on reward (Assumption A1), and bound $_R$ as:

$$_R := \frac{f}{1}E_{s;a\ \ d_M}\ [|r_{M\ 1}(s;a)\quad r_M\ (s;a)|] + \frac{1\quad f}{1}E_{s;a\ \ d_M}\ [|r_{M\ 2}(s;a)\quad r_M\ (s;a)|]$$

$$\frac{C_{r;}}{1}E_{s;a\ \ d_M}\ \left[\frac{1}{\sqrt{|D(s;a)|}}\right] + \frac{1}{1}||R_M\quad R_{\hat{M}}\ || := \ ^u_R:$$

Note that both of these terms are of the order of $O(1=(1\quad))$ and hence they don't gure in the informal bound in Theorem 4 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 , we utilize a result directly from Kumar et al. (2020) for the empirical MDP, $\overline{M}$, which holds for any policy $(a|s)$, as shown below.

$$\frac{}{(1\quad)}E_{s;a\ \ d_M\ (s;a)}\quad P_M\quad P_{M\ 1}\quad Q_M\qquad \frac{2\ R_{\ max}C_{P;}}{(1\quad)^2}E_{s\ \ d_{\overline{M}}(s)}\left[\frac{\sqrt{|A|}}{\sqrt{|D(s)|}}\quad\sqrt{D_{CQL}(\ ;\quad)(s)+1}\right]:$$

Step 2: Incorporate policy improvement in the f-inrerpolant MDP. Now we incorporate the improvement of policy $_{out}$ over the policy on a weighted mixture of $\hat{M}$ and $\overline{M}$. In what follows, we derive a lower-bound on this improvement by using the fact that policy $_{out}$ is obtained by maximizing $\hat{J}(f;\quad)$ from Equation 15. As a direct consequence of Equation 15, we note that

$$\hat{J}(f;\ \ _{out}) = J(\overline{M};\hat{M};f;\ \ _{out})\quad \frac{(\quad;f)}{1}\quad \hat{J}(f;\quad) = J(\overline{M};\hat{M};f;\quad)\quad \frac{(\quad;f)}{1}\qquad(18)$$

Following Step 1, we will use the upper bound on $J(\overline{M};\hat{M};f;\quad)$ for policy $=\ _{out}$ and a lower-bound on $J(\overline{M};\hat{M};f;\quad)$ for policy $=$ and obtain the following inequality:

$$J(\ _{out};M)\quad \frac{(\quad;f)}{1}\quad \Big\{ J(\quad;M)\quad \frac{(\quad;f)}{1}\quad \frac{4\ (1\quad f)R_{max}}{(1\quad)^2}D(P_M\ ;P_{\hat{M}})$$

$$\underbrace{\frac{2\ f}{(1\quad)}\ E_{d_M^{out}}\Big[P_M^{out}\quad P_{\overline{M}}^{out}\quad Q_M^{out}\Big]}_{:=(\ \ )}\quad \underbrace{\frac{4\ R_{\ max}C_{P;}\ \ f}{(1\quad)^2}E_{s\ \ d_M}\left[\sqrt{\frac{|A|}{|D(s)|}}\right]}_{:=(\ \hat{}\ )}\quad ^u_R\Big\}:$$

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:

$$(\ \ )\quad \frac{4\ fC_{\ P;}\ \ R_{max}}{(1\quad)^2}E_{s\ \ d_M^{out}(s)}\left[\frac{\sqrt{|A|}}{\sqrt{|D(s)|}}\quad\sqrt{D_{CQL}(\ _{out};\quad)(s)+1}\right]:\qquad(19)$$

Finally, using Equation 19, we can lower-bound the policy return difference as:

$$J(\ _{out};M)\quad J(\quad;M)\qquad \frac{(\quad;f)}{1}\qquad \frac{(\quad;f)}{1}\qquad \frac{4\ (1\quad f)R_{max}}{(1\quad)^2}D(P_M\ ;P_{\hat{M}})\quad(\ \ )\qquad ^u_R:$$

Plugging the bounds for terms (a), (b) and (c) in the expression for where $J(\ _{out};M)\quad J(\quad;M)$, we obtain:

$$=\quad \frac{4f\ R_{\ max}C_{P;}}{(1\quad)^2}\quad E_{s\ \ d_M^{out}(s)}\left[\frac{\sqrt{|A|}}{\sqrt{|D(s)|}}\quad\sqrt{D_{CQL}(\ _{out};\quad)(s)+1}\right] + (\ \hat{}\ )\qquad ^u_R$$

$$+\ \frac{4(1\quad f)\ R_{\ max}}{(1\quad)^2}D(P_M\ ;P_{\hat{M}})\qquad \frac{(\quad;f)}{1} + \frac{(\quad;f)}{1}:\qquad(20)$$

□

Remark 11 (Interpretation of Theorem 4). Now we will interpret the theoretical expression for in Equation 20, and discuss the scenarios when it is negative When the expression for is negative, the policy $_{out}$ is an improvement over in the original MDP, M.

- First note that we have never used the fact that the learned models $P_{\hat{M}}$ close to the actual MDP, P_M on the states visited by the behavior policy in our analysis. We will use this fact now: in practical scenarios, $(\;;f)$ is expected to be smaller than $(\;;f)$, since $(\;;f)$ is directly controlled by the difference and density ratio of $(s;a)$ and $d(s;a)$: $(\;;f) \quad (\;;f = 1) = \sum_{s;a} d_{\hat{M}} \; (s;a) \;\; d_{\hat{M}} \; (s;a) = d_{\overline{M}} \; (s;a) \quad 1^{2}$ by Lemma 1 which is expected to be small for the behavior policy in cases when the behavior policy marginal in the empirical MDP, $d_{\overline{M}}(s;a)$, is broad. This is a direct consequence of the fact that the learned dynamics integrated with the policy under the learned model: $P_{\hat{M}}$ is closer to its counterpart in the empirical MDP: $P_{\overline{M}}$ for . 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, 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 19, or Lemma D.4.1 in Kumar et al. (2020)). Thus, in practice, we argue that we are interested in situations where $(\;;f) > (\;;f)$, in which case by increasing , we can make the expression for in Equation 20 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 $jD(s)j$), if we can learn a good model, i.e., $D(P_M \;; P_{\hat{M}})$ 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.

D EXPERIMENTAL DETAILS

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

D.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:

$$\hat{T} (s_{t+1}; r j s; a) = N (\;\; (s_t; a_t); \;\; (s_t; a_t)):$$

We train an ensemble of 7 such dynamics models following Janner et al. (2019) 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 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 4.2, we additionally use a two-head architecture to output the mean and variance after the last hidden layer following Yu et al. (2020c).

In the image-based setting, we follow Rafailov et al. (2020) and use a variational model with the following components:

$$\begin{array}{ll} \text{Image encoder:} & h_t = E \; (o_t) \\ \text{Inference model:} & s_t \quad q \; (s_t j h_t; s_{t \; 1}; a_{t \; 1}) \\ \text{Latent transition model:} & s_t \quad \hat{T} \; (s_t j s_{t \; 1}; a_{t \; 1}) \\ \text{Reward predictor:} & r_t \quad p \; (r_t j s_t) \\ \text{Image decoder:} & o_t \quad D \; (o_t j s_t): \end{array} \quad (21)$$

We train the model using the evidence lower bound:

$$\max \sum_{=0}^{T-1} \left[E_q [\log D (o_{+1} | s_{+1})] \quad E_q \left[D_{KL} [q (o_{+1} ; s_{+1} | s ; a) \| \hat{T} (s_{+1} ; a_{+1})] \right] \right]$$

At each step we sample a latent forward model $\hat{T}$ from a fixed set of K models $[\hat{T}_1 ; : : : ; \hat{T}_K]$. For the encoder E 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 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. (2019) 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 D and another 50% from D_{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 4. We represent the Q-networks and policy as 3-layer feedforward neural networks with 256 hidden units.

For the choice of $(s; a)$ in Equation 4, we can obtain the Q-values that lower-bound the true value of the learned policy by setting $(s; a) = d_{\hat{M}} (s) \; (a|s)$. However, as discussed in Kumar et al. (2020), computing by alternating the full off-policy evaluation for the policy $\hat{}^k$ at each iteration k and one step of policy improvement is computationally expensive. Instead, following Kumar et al. (2020), we pick a particular distribution $(a|s)$ that approximates the the policy that maximizes the Q-function at the current iteration and set $(s; a) = d_{\hat{M}} (s) \; (a|s)$. We formulate the new objective as follows:

$$\hat{Q}^{k+1} \quad \arg\min_{Q} \; E_{s \; d_{\hat{M}}(s); a \; (a|s)} [Q(s; a)] \quad E_{s; a \; D} [Q(s; a)] + \frac{1}{2} E_{s; a; s^0 \; d_f} \left[\left(Q(s; a) \quad \hat{B} \hat{Q}^k (s; a) \right)^2 \right] + R( ); \tag{22}$$

where $R( )$ is a regularizer on . In practice, we pick $R( )$ to be the $D_{KL}(\; (a|s) \| \text{Unif}(a))$ and under such a regularization, the first term in Equation 22 corresponds to computing softmax of the Q-values at any state s as follows:

$$\hat{Q}^{k+1} \quad \arg\min_{Q} \max \; E_{s \; d_{\hat{M}}(s)} \left[\log \sum_a Q(s; a) \right] \quad E_{s; a \; D} [Q(s; a)] + \frac{1}{2} E_{s; a; s^0 \; d_f} \left[\left(Q(s; a) \quad \hat{B} \hat{Q}^k (s; a) \right)^2 \right] : \tag{23}$$

We estimate the log-sum-exp term in Equation 23 by sampling 10 actions at every state s in the batch from a uniform policy $\text{Unif}(a)$ and the current learned policy $(a|s)$ with importance sampling following Kumar et al. (2020).

D.2 Workflow with COMBO and Hyperparameter Selection

In this section, we discuss the hyperparameters associated with COMBO along with guidelines to choose hyperparameters for offline RL tasks with a lot of variety in terms of dataset-diversity, input modality, etc. 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 (Fu et al., 2020) 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.

We now list the additional hyperparameters and our decisions behind these (which can serve as guidelines/rules for hyperparameter selection when COMBO is used on a new task) as follows.

- Rollout length h. We perform a short-horizon model rollouts in COMBO similar to Yu et al. (2020c) and Rafailov et al. (2020). 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. (2020).
- Q-function and policy learning rates. According to Kumar et al. (2020), we used smaller learning rates for the policy as compared to the Q-function and we found that 3e 4 for the Q-function learning rate (also used previously in Kumar et al. (2020)) and 1e 4 for the policy learning rate (also recommended previously in Kumar et al. (2020) 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. In the image-based domains, we followed the defaults from prior work (Rafailov et al., 2020) and used 3e 4 for both the policy and Q-function.
- Conservative coef cient . As noted in our theoretical results in Lemma 1, the amount of conservatism depends on the choice of fraction f and $(s; a)$. In principle, we only need to control one of these factors, f, , to obtain the right degree of conservatism. Since we do not alter f and $(s; a)$ for different quality datasets (see Appendix D.1 for our choice of f; was chosen based on model-prediction error as discussed next) we instead choose values of for different dataset types. We xed to take one of three values from a default set $f 0{:}5; 1{:}0; 5{:}0g$, which correspond to low conservatism, medium conservatism and high conservatism. A larger 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 is desirable with diverse dataset distributions. On the D4RL experiments, we found that $= 0{:}5$ 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 $= 5{:}0$ whereas more "diverse" dataset distributions: random and medium-replay datasets work best with smaller ($= 0{:}5$ for walker2d and $= 1{:}0$ for hopper) which is consistent with the intuition. On generalization experiments, $= 1{:}0$ works best for all environments. In the image-domains we used $= 0{:}5$ for the medium-replay walker-walk task and and $= 1{:}0$ for all other domains, which again is in accordance with the impact of on performance.
- Choice of $(s; a)$. We rst decouple $(s; a) = (s) (ajs)$ for convenience. As discussed in Appendix D.1, we use (ajs) as the soft-maximum of the Q-values and estimated with log-sum-exp. For (s), we used $d_{\hat{M}}$ for the hopper task in D4RL and d_f for the rest of the environments. As consistent with Lemma 1, we found that COMBO works well with $(s) = d_{\hat{M}}$ in settings where the learned model is accurate (as measured by model prediction error on a held-out validation dataset), which is the case in hopper. For the remaining domains, we used $(s) = d_f$.
- Choice of (ajs). For the rollout policy , we use $(ajs) = \text{Unif}(a)$ for the hopper task in D4RL since the dynamics model is most accurate on this task (as measured by model-prediction error on a held-out validation subset of the data) which enables augmenting the dataset with uniform-at-random actions without introducing much model bias into learning, and also in the ant-angle generalization experiment. For the remaining state-based environments, we use the default $(ajs) = (ajs)$ which is consistent with our theory (Lemma 1). In the image-based domain, we used $(ajs) = \text{Unif}(a)$ in the walker-walk domain and $(ajs) = (ajs)$ for the sawyer-door environment. We found that $(ajs) = \text{Unif}(a)$ behaves less conservatively and is suitable to tasks where dynamics models can be learned fairly precisely.
- Choice of Backup. Following CQL (Kumar et al., 2020), we use the standard deterministic backup for COMBO.

D.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 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.

D.4 DETAILS OF IMAGE-BASED ENVIRONMENTS

We use the standard `walker-walk` environment from Tassa et al. (2018) with 64×64 pixel observations and an action repeat of 2. Datasets were constructed the same way as Fu et al. (2020) 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. (2020b). 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.

D.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.

E EMPIRICAL EVIDENCE ON CHALLENGES OF UNCERTAINTY QUANTIFICATION

In this section, we perform empirical evaluations to show that 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 (Kidambi et al., 2020) and MOPO (Yu et al., 2020c). In our evaluations, we consider two uncertainty quantification methods, maximum learned variance over the ensemble (denoted as **Max Var**) $\max_{i=1,\dots,N} \|\Sigma^i(\mathbf{s},\mathbf{a})\|_\text{F}$ (used in MOPO) and the variance of the model prediction over the ensemble (denoted as **Ens. Var**) $\max_{i=1,\dots,N} \|\mu^i(\mathbf{s},\mathbf{a}) - \frac{1}{N}\sum_{j=1}^{N}\mu^j(\mathbf{s},\mathbf{a})\|_2$ (used in MOPO and MOReL) where we use an ensemble of N probabilistic dynamics models $\{\widehat{T}^i(\mathbf{s}_{t+1}, r|\mathbf{s},\mathbf{a}) = \mathcal{N}(\mu^i(\mathbf{s}_t,\mathbf{a}_t), \Sigma^i(\mathbf{s}_t,\mathbf{a}_t))\}_{i=1}^N$.

As shown in Table 1, MOPO performs underwhelmingly on medium datasets in the D4RL datasets where the dataset is collected with a single policy and hence with relatively narrow data coverage of the whole state space. To empirically analyze the poor performance of MOPO on those datasets, we visualize the correlation between the true model error and two uncertainty quantification methods **Max Var** and **Ens. Var**. We normalize both the model error and the uncertainty estimates to be within scale $[0, 1]$. As shown in Figure 3, on all three medium datasets, **Max Var** tends to be overly conservative and **Ens. Var** behaves too optimistic to correctly quantify the true model error, suggesting that uncertainty estimation used by MOPO is not accurate and might be the major factor that results in its poor performance. Meanwhile, COMBO circumvents challenging uncertainty quantification problem and achieves much better performances on those medium datasets, indicating the effectiveness and the robustness of the method.