RAMBO-RL: Robust Adversarial Model-Based Offline Reinforcement Learning

Marc Rigter, Bruno Lacerda, Nick Hawes
Oxford Robotics Institute
University of Oxford
{mrigter, bruno, nickh}@robots.ox.ac.uk

Abstract

Offline reinforcement learning (RL) aims to find performant policies from logged data without further environment interaction. Model-based algorithms, which learn a model of the environment from the dataset and perform conservative policy optimisation within that model, have emerged as a promising approach to this problem. In this work, we present Robust Adversarial Model-Based Offline RL (RAMBO), a novel approach to model-based offline RL. We formulate the problem as a two-player zero sum game against an adversarial environment model. The model is trained to minimise the value function while still accurately predicting the transitions in the dataset, forcing the policy to act conservatively in areas not covered by the dataset. To approximately solve the two-player game, we alternate between optimising the policy and adversarially optimising the model. The problem formulation that we address is theoretically grounded, resulting in a probably approximately correct (PAC) performance guarantee and a pessimistic value function which lower bounds the value function in the true environment. We evaluate our approach on widely studied offline RL benchmarks, and demonstrate that it outperforms existing state-of-the-art baselines.

1 Introduction

Reinforcement learning (RL) [61] has achieved state-of-the-art performance on many sequential decision-making problems [40, 45, 59]. However, the need for extensive exploration prohibits the application of RL to many real world domains where such exploration is costly or dangerous. Offline RL [33, 35] overcomes this limitation by learning policies from static, pre-recorded datasets.

Online RL algorithms perform poorly in the offline setting due to the distributional shift between the state-action pairs in the dataset and those taken by the learnt policy. Thus, an important aspect of offline RL is to introduce conservatism to prevent the learnt policy from executing state-action pairs which are out of distribution. Model-free offline RL algorithms [15, 23, 27, 29, 31, 72] train a policy from only the data present in the fixed dataset, and incorporate conservatism either into the value function or by directly constraining the policy.

On the other hand, model-based offline RL algorithms [76, 75, 25, 63, 39] use the dataset to learn a model of the environment, and train a policy using additional synthetic data generated from that model. By training on additional synthetic data, model-based algorithms can potentially generalise better to states not present in the dataset, or to solving new tasks. Previous approaches to model-based offline RL incorporate conservatism by estimating the uncertainty in the model and applying reward penalties for state-action pairs that have high uncertainty [76, 25]. However, uncertainty estimation can be unreliable for neural network models [75, 37]. Like recent work [75], we propose an approach for offline model-based RL which *does not require uncertainty estimation*.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

In this work we present Robust Adversarial Model-Based Offline (RAMBO) RL, a new algorithm for model-based offline RL. RAMBO incorporates conservatism by modifying the *transition dynamics* of the learnt Markov decision process (MDP) model in an adversarial manner. We formulate the problem of offline RL as a zero-sum game against an adversarial environment. To solve the resulting maximin optimisation problem, we alternate between optimising the agent and optimising the adversary in the style of Robust Adversarial RL (RARL) [50]. Unlike existing RARL approaches, our model-based approach forgoes the need to define and train an adversary policy, and instead only learns an adversarial model of the MDP. We train the agent policy with an actor-critic algorithm using synthetic data generated from the model in addition to data sampled from the dataset, similar to Dyna [60] and a number of recent methods [22, 76, 25, 75]. We update the environment model so that it reduces the value function for the agent policy, while still accurately predicting the transitions in the dataset. As a result, our approach introduces conservatism by generating *pessimistic synthetic transitions* for state-action pairs which are out-of-distribution. The theoretical formulation of offline RL that our algorithm addresses yields a PAC bound for the performance gap with respect to any policy covered by the dataset, and a pessimistic value function that lower bounds the value function in the true environment.

In summary, the main contributions of this work are:

- RAMBO, a novel and theoretically-grounded model-based offline RL algorithm which enforces conservatism by training an adversarial dynamics model.
- Adapting the Robust Adversarial RL approach to model-based offline RL by proposing a new formulation of RARL, where instead of defining and training an adversary policy, we directly train the model adversarially.

In our experiments we demonstrate that RAMBO outperforms current state-of-the-art algorithms on the D4RL benchmarks [14]. Furthermore, we provide ablation results which show that training the model adversarially is crucial to the strong performance of RAMBO.

2 Related Work

Offline RL: Offline RL addresses the problem of learning policies from fixed datasets, and has been applied to domains such as healthcare [42, 57], natural language processing [24, 23], and robotics [30, 38, 51]. Model-free offline RL algorithms do not require a learnt model. Approaches for model-free offline RL include importance sampling algorithms [36, 41], constraining the learnt policy to be similar to the behaviour policy [16, 28, 72, 23, 58, 15], incorporating conservatism into the value function during training [9, 27, 31, 73], using uncertainty quantification to generate more robust value estimates [1, 2, 29], or applying only a single iteration of policy iteration [7, 49]. In contrast, model-based approaches learn a model of the environment and generate synthetic data from that model [60] to optimise a policy using either planning [4] or RL algorithms [25, 76, 75]. By training a policy on additional synthetic data, model-based approaches have the potential for broader generalisation and for solving new tasks [6, 76]. A simple approach to ensuring conservatism is to constrain the policy to be similar to the behaviour policy in the same fashion as some model-free approaches [8, 39, 63]. Another approach is to apply reward penalties for executing state-action pairs with high uncertainty in the environment model [25, 74, 76]. However, this requires explicit uncertainty estimates which may be unreliable for neural network models [17, 37, 46, 75]. COMBO [75] obviates the need for uncertainty estimation in model-based offline RL by adapting model-free techniques [31] to regularise the value function for out-of-distribution samples. Like COMBO, our approach does not require uncertainty estimation.

Most approaches to model-based offline RL use maximum likelihood estimates (MLE) of the MDP trained using standard supervised learning [4, 39, 63, 76, 75]. However, other methods have been proposed to learn models which are more suitable for offline policy optimisation. One approach is to reweight the loss function to ensure the model is accurate under the state-action distribution generated by the policy [34, 53, 20]. In contrast, our approach produces pessimistic synthetic transitions when out-of-distribution.

Most related to our work is a recent paper [66] which introduces the maximin formulation of offline RL that we address. This existing work motivates our approach theoretically by showing that the problem formulation obtains probably approximately correct (PAC) guarantees. However, [66]

only addresses the theoretical aspects of the problem formulation and does not propose a practical algorithm. In this work, we propose a practical RL algorithm to solve the maximin formulation of model-based offline RL.

Robust RL: Algorithms for Robust MDPs [5, 12, 21, 43, 54, 64, 70] find the policy with the best worst-case performance over a set of possible MDPs. Typically, it is assumed that the uncertainty set of MDPs is specified a priori. To eliminate the need to specify the set of possible MDPs, model-free approaches to Robust MDPs [68, 55] instead assume that samples can be drawn from a misspecified MDP which is similar to the true MDP. As our work addresses offline RL, we assume that we have a fixed dataset from the true MDP.

Our approach is conceptually similar to Robust Adversarial RL (RARL) [50], a method proposed to improve the robustness of RL policies in the *online* setting. RARL is posed as a two-player zero-sum game where the agent plays against an adversary which perturbs the environment. Formulations of model-free RARL differ in how they define the action space of the adversary. Options include allowing the adversary to apply perturbation forces to the simulator [50], add noise to the agent's actions [65], or periodically take over control [48]. A model-based approach to RARL is proposed in [13], which learns an optimistic and pessimistic model to encourage online exploration. However, this existing approach requires uncertainty estimation as well as an adversarial policy to be learnt in *addition to the model*. Our work follows the paradigm of RARL and alternates between agent and adversarial updates in a maximin formulation. We adapt model-based RARL to the offline setting and propose an alternative formulation: we eliminate the need to learn an adversary policy and instead *directly modify the MDP model adversarially*.

3 Preliminaries

MDPs and Offline RL: An MDP is defined by the tuple, $M = (S, A, T, R, \mu_0, \gamma)$. S and A denote the state and action spaces respectively, $R(s, a)$ is the reward function, $T(s'|s, a)$ is the transition function, μ_0 is the initial state distribution, and $\gamma \in (0, 1)$ is the discount factor. In this work we consider Markovian policies, $\pi \in \Pi$, which map from each state to a distribution over actions. We denote the (improper) discounted state visitation distribution of a policy by $d^\pi_M(s) := \sum_{t=0}^{\infty} \gamma^t \Pr(s_t = s|\pi, M)$, where $\Pr(s_t = s|\pi, M)$ is the probability of reaching state s at time t by executing policy π in M. The improper state-action visitation distribution is $d^\pi_M(s, a) = \pi(a|s) \cdot d^\pi_M(s)$. We also denote the normalised state-action visitation distribution by $\tilde{d}^\pi_M(s, a) = (1 - \gamma) \cdot d^\pi_M(s, a)$.

The value function, $V^\pi_M(s)$, represents the expected discounted return from executing π from state s in M: $V^\pi_M(s) = \mathbb{E}_{\pi,M}\big[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\big]$. We write V^π_M to indicate the value function under the initial state distribution, i.e. $V^\pi_M = \sum_{s\in S} \mu_0(s) V^\pi_M(s)$. The standard objective for MDPs is to find the policy which maximises V^π_M. The state-action value function, $Q^\pi_M(s, a)$, is the expected discounted cumulative reward from taking action a at state s and then executing π thereafter.

In offline RL we only have access to a fixed dataset of transitions from the MDP, $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^{|\mathcal{D}|}$. The goal of offline RL is to find the best possible policy using the fixed dataset.

Model-Based Offline RL Algorithms: Model-based approaches to offline RL use a model of the MDP to help train a policy. The dataset is used to learn a dynamics model, $\widehat{T}$, which is typically trained via maximum likelihood estimation: $\min_{\widehat{T}} \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\big[-\log \widehat{T}(s'|s, a)\big]$. A model of the reward function, $\widehat{R}(s, a)$, can also be learnt if it is unknown. The estimated MDP, $\widehat{M} = (S, A, \widehat{T}, \widehat{R}, \mu_0, \gamma)$, has the same state and action space as the true MDP but uses the learnt transition and reward functions. Thereafter, any planning or RL algorithm can be used to recover optimal policy in the learnt model, $\widehat{\pi} = \arg\max_{\pi\in\Pi} V^\pi_{\widehat{M}}$.

Unfortunately, directly applying this approach to the offline RL setting does not perform well due to distributional shift. In particular, if the dataset does not cover the entire state-action space, the model will inevitably be inaccurate for some state-action pairs. Thus, naive policy optimisation on a learnt model in the offline setting can result in *model exploitation* [22, 32, 53]. To mitigate this issue, we propose the novel approach of enforcing conservatism by adversarially modifying the transition dynamics of $\widehat{M}$.

In line with existing works [76, 75, 8], we use model-based policy optimisation (MBPO) [22] to learn the optimal policy for $\widehat{M}$. MBPO utilises a standard actor-critic RL algorithm. However, the value function is trained using an augmented dataset $\mathcal{D} \cup \mathcal{D}_{\widehat{M}}$, where $\mathcal{D}_{\widehat{M}}$ is synthetic data generated by simulating rollouts in the learnt model. To generate the synthetic data, MBPO performs k-step rollouts in $\widehat{M}$ starting from states $s \in \mathcal{D}$, and adds this data to $\mathcal{D}_{\widehat{M}}$. To train the policy, minibatches of data are drawn from $\mathcal{D} \cup \mathcal{D}_{\widehat{M}}$, where each datapoint is sampled from the real data, $\mathcal{D}$, with probability f, and from $\mathcal{D}_{\widehat{M}}$ with probability $1 - f$.

Robust Adversarial Reinforcement Learning: RARL addresses the problem of finding a robust agent policy, π, in the online RL setting by posing the problem as a two-player zero sum game against adversary policy, $\bar{\pi}$:

$$\pi = \arg\max_{\pi \in \Pi} \min_{\bar{\pi} \in \bar{\Pi}} V_M^{\pi, \bar{\pi}} \tag{1}$$

where $V_M^{\pi, \bar{\pi}}$ is the expected value from executing π and $\bar{\pi}$ in environment M. Different approaches define the action space for $\bar{\pi}$ in different ways, as discussed in Section 2. For a scalable approximation to the optimisation problem in Equation 1, algorithms for RARL alternate between applying steps of stochastic gradient ascent to the agent's policy to increase the expected value, and stochastic gradient descent to the adversary's policy to decrease the expected value. In our work, we follow the RARL paradigm of alternating between agent and adversarial updates and adapt it to the model-based offline setting. Instead of defining a separate adversary policy, we treat the *model itself* as the policy to be adversarially trained.

4 Problem Formulation

For the sake of generality, we assume that both the transition function and reward function are unknown. Hereafter, we write $\widehat{T}$ to denote both the learnt dynamics and reward function, where $\widehat{T}(s', r|s, a)$ is the probability of receiving reward r and transitioning to s' after executing (s, a). We address the maximin formulation of offline RL recently proposed by [66]:

Problem 1. *For some dataset, $\mathcal{D}$, and some fixed constant $\xi > 0$, find the policy π defined by*

$$\pi = \arg\max_{\pi \in \Pi} \min_{\widehat{T} \in \mathcal{M}_{\mathcal{D}}} V_{\widehat{T}}^{\pi}, \text{ where} \tag{2}$$

$$\mathcal{M}_{\mathcal{D}} = \Big\{ \widehat{T} \mid \mathbb{E}_{\mathcal{D}} \big[\mathrm{TV}(\widehat{T}_{\mathrm{MLE}}(\cdot|s,a), \widehat{T}(\cdot|s,a))^2 \big] \leq \xi \Big\}, \tag{3}$$

where $\mathrm{TV}(P_1, P_2)$ is the total variation distance between distributions P_1 and P_2, and $\widehat{T}_{\mathrm{MLE}}$ denotes the maximum likelihood estimate of the MDP given the offline dataset, $\mathcal{D}$.

Thus, the set defined in Equation 3 contains MDPs which are similar to the maximum likelihood estimate under state-action pairs in $\mathcal{D}$. However, because the expectation in Equation 3 is taken under $\mathcal{D}$, there is no restriction on $\widehat{T}$ for regions of the state-action space not covered by $\mathcal{D}$. We present a brief overview of the theoretical guarantees from [66] in the following subsection.

Remark 1. Note that Problem 1 differs from the pessimistic MDP formulations introduced by MOPO [76] and MOReL [25]. Problem 1 considers the worst-case transition dynamics, while the pessimistic MDPs constructed by MOPO and MOReL only modify the reward function by applying reward penalties for state-action pairs with high uncertainty.

4.1 Theoretical Motivation

The theoretical analysis from [66] shows that solving Problem 1 outputs a policy that with high probability is approximately as good as any policy with a state-action distribution that is covered by the dataset. This is formally stated in the following theorem.

Theorem 1 (PAC guarantee from [66], Theorem 5)**.** *Denote the true MDP transition function by T, and let $\mathcal{M}$ denote a hypothesis class of MDP models such that $T \in \mathcal{M}$. Let π denote the solution to Problem 1 for dataset $\mathcal{D}$. Then with probability $1 - \delta$ for any policy, $\pi^* \in \Pi$, we have*

$$V_T^{\pi^*} - V_T^{\pi} \leq (1-\gamma)^{-2} c_1 \sqrt{C_{\pi^*}} \sqrt{G_{\mathcal{M}_1} + G_{\mathcal{M}_2} + \xi_n^2 + \frac{\ln(c/\delta)}{|\mathcal{D}|}}, \text{ where}$$

$$C_{\pi^*} = \max_{T' \in \mathcal{M}} \frac{\mathbb{E}_{(s,a)\sim \tilde{d}_T^{\pi^*}}\left[\mathrm{TV}(T'(\cdot|s,a), T(\cdot|s,a))^2\right]}{\mathbb{E}_{(s,a)\sim\rho}\left[\mathrm{TV}(T'(\cdot|s,a), T(\cdot|s,a))^2\right]},$$

where ρ is the state-action distribution from which $\mathcal{D}$ was sampled, and c and c_1 are universal constants. We refer the reader to Appendix A of [66] for the definitions of $G_{\mathcal{M}_1}$, $G_{\mathcal{M}_2}$, and ξ_n.

The quantity C_{π^*} is upper bounded by the maximum density ratio between the comparator policy, π^*, and the offline distribution, i.e. $C_{\pi^*} \leq \max_{(s,a)} \tilde{d}_T^{\pi^*}(s,a)/\rho(s,a)$. It represents the discrepancy between the distribution of data in the dataset compared to the visitation distribution of policy π^*. Theorem 1 shows that if we find a policy by solving Problem 1, the performance gap of that policy is bounded with respect to any other policy π^* that has a state-action distribution which is covered by the dataset.

Furthermore, the value function under the worst-case model in the set defined by Problem 1 is a lower bound on the value function in the true environment, as stated by Proposition 1.

Proposition 1 (Pessimistic value function)**.** *Let T denote the true transition function for some MDP, and let $\mathcal{M}_\mathcal{D}$ be the set of MDP models defined in Equation 3. Then for any policy π, with probability $1-\delta$ we have that*

$$\min_{\widehat{T} \in \mathcal{M}_\mathcal{D}} V_{\widehat{T}}^\pi \leq V_T^\pi.$$

Proposition 1 follows from the fact that $T \in \mathcal{M}_\mathcal{D}$ with high probability, which is proven in [66] (Appendix E.2). Proposition 1 shows that we can expect the performance of any policy in the true MDP to be at least as good as the value in the worst-case model defined in Problem 1.

While [66] provides the theoretical motivation for solving Problem 1, it does not propose a practical algorithm. In this work, we focus on developing a practical approach to solving Problem 1.

5 RAMBO-RL

In this section, we present Robust Adversarial Model-Based Offline RL (RAMBO), our algorithm for solving Problem 1. The main difficulty with solving Problem 1 is that it is unclear how to find the worst-case MDP in the set defined in Equation 3. To arrive at a scalable solution, we propose a novel approach which is in the spirit of RARL. We alternate between optimising the agent policy to increase the expected value, and adversarially optimising the model to decrease the expected value. In this section, we first describe how we compute the gradient to adversarially train the model. Then, we discuss how to ensure that the model remains approximately within the constraint set defined in Problem 1. Finally, we present our overall algorithm.

5.1 Model Gradient

We propose a policy gradient-inspired approach to adversarially optimise the model. Typically, policy gradient algorithms are used to modify the distribution over actions taken by a policy at each state [62, 71]. In contrast, the update that we propose modifies the likelihood of the successor states and rewards in the MDP model.

We assume that the MDP model is defined by parameters, ϕ, and we write $\widehat{T}_\phi$ to indicate this. We denote by V_ϕ^π the value function for policy π in model $\widehat{T}_\phi$. To approximately find $\min_{\widehat{T}_\phi \in \mathcal{M}_\mathcal{D}} V_\phi^\pi$ as required by Problem 1 via gradient descent, we wish to compute the gradient of the model parameters that reduces the value of the policy within the model, i.e. $\nabla_\phi V_\phi^\pi$.

Proposition 2 (Model Gradient)**.** *Let ϕ denote the parameters of a parametric MDP model $\widehat{T}_\phi$, and let V_ϕ^π denote the value function for policy π in $\widehat{T}_\phi$. Then:*

$$\nabla_\phi V_\phi^\pi = \mathbb{E}_{s\sim d_\phi^\pi, a\sim\pi, (s',r)\sim\widehat{T}_\phi}\left[(r + \gamma V_\phi^\pi(s')) \cdot \nabla_\phi \log \widehat{T}_\phi(s', r|s,a)\right] \tag{4}$$

The proof of Proposition 2 is given in Appendix A. We can subtract the baseline $Q_\phi^\pi(s,a)$ without biasing the gradient estimate (see Appendix A.1 for details):

$$\nabla_\phi V_\phi^\pi = \mathbb{E}_{s\sim d_\phi^\pi, a\sim\pi, (s',r)\sim\widehat{T}_\phi}\left[(r + \gamma V_\phi^\pi(s') - Q_\phi^\pi(s,a)) \cdot \nabla_\phi \log \widehat{T}_\phi(s', r|s,a)\right] \tag{5}$$

The Model Gradient differs from the standard policy gradient in that a) it is used to update the likelihood of successor states in the model, rather than actions in a policy, and b) the "advantage" term $r + \gamma V_\phi^\pi(s') - Q_\phi^\pi(s,a)$ compares the utility of receiving reward r and transitioning to s' to the expected value of state action pair (s,a). In contrast, the standard advantage term, $Q(s,a) - V(s)$ [56], compares the value of executing state-action pair (s,a) to the value at s. To estimate V_ϕ^π and Q_ϕ^π, we use the critic learnt by the actor-critic algorithm used for policy optimisation. Thus, we use the critic *both* for training the policy and adversarially training the model.

Remark 2. The Model Gradient can be thought of as a specific instantiation of the policy gradient if we view $\widehat{T}_\phi$ as a adversarial "policy" on an augmented MDP, M^+, i.e. $\widehat{T}_\phi : S^+ \to \mathrm{Dist}(A^+)$. The augmented state space, $S^+ = S \times A$ includes the state in the original MDP augmented by the action taken by the agent. The augmented action space consists of the reward applied and the successor state, $A^+ = S \times [R_{\text{min}}, R_{\text{max}}]$. Thus, we can think of this approach as an instantiation of RARL in which the adversary policy ($\bar{\pi}$ in Equation 1) that we train is the *model itself*.

5.2 Adversarial Model Training

If we were to update the model using Equation 5 alone, this would allow the model to be modified arbitrarily such that the value function in the model is reduced. However, the set of plausible MDPs given by Equation 3 states that over the dataset, $\mathcal{D}$, the model $\widehat{T}_\phi$ should be close to the maximum likelihood estimate, $\widehat{T}_{\text{MLE}}$. Specifically, in the inner optimisation of Problem 1 we wish to find a solution to the constrained optimisation problem

$$\min_{\widehat{T}_\phi} V_\phi^\pi, \quad s.t. \;\; \mathbb{E}_{\mathcal{D}}\big[\text{TV}(\widehat{T}_{\text{MLE}}(\cdot|s,a), \widehat{T}_\phi(\cdot|s,a))^2\big] \leq \xi. \tag{6}$$

The Lagrangian relaxation leads to the unconstrained problem

$$\max_{\lambda \geq 0} \min_{\widehat{T}_\phi} \Big(L(\hat{T}, \lambda) := V_\phi^\pi + \lambda\big(\mathbb{E}_{\mathcal{D}}\big[\text{TV}(\widehat{T}_{\text{MLE}}(\cdot|s,a), \widehat{T}_\phi(\cdot|s,a))^2\big] - \xi\big)\Big), \tag{7}$$

where λ is the Lagrange multiplier. Rather than optimising the Lagrange multiplier, we find that in practice fixing λ to apply a constant weighting between the two terms works well with minimal tuning. To facilitate easier tuning of the learning rate, in our implementation we apply the weighting constant to the value function term rather than the model term, which is equivalent up to a scaling factor. This leads to

$$\min_{\widehat{T}_\phi} \Big(\lambda V_\phi^\pi + \mathbb{E}_{\mathcal{D}}\big[\text{TV}(\widehat{T}_{\text{MLE}}(\cdot|s,a), \widehat{T}_\phi(\cdot|s,a))^2\big]\Big). \tag{8}$$

We aim to make our algorithm efficient and simple to implement. Therefore, rather than minimising the TV distance between the model and MLE model as prescribed by Equation 8, we directly optimise the standard MLE loss. This leads to the final loss function:

$$\mathcal{L}_\phi = \lambda V_\phi^\pi - \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}}\big[\log \widehat{T}_\phi(s', r|s,a)\big]. \tag{9}$$

Thus, the loss function for the model in Equation 9 simply adds the adversarial term to the standard MLE loss. By minimising the loss function in Equation 9, the model is trained to a) predict the transitions within the dataset, and b) reduce the value function of the policy, with λ determining the tradeoff between these two objectives. Choosing λ to be small ensures that the MLE term dominates for transitions within $\mathcal{D}$, ensuring that the model fits the dataset accurately. Because the MLE term is only computed over $\mathcal{D}$, the adversarial term dominates outside of the dataset meaning that the model is modified adversarially for transitions outside of the dataset.

To estimate the gradient of the loss function in Equation 9 for stochastic gradient descent, we sample a minibatch of transitions from $\mathcal{D}$ to estimate the MLE term. The gradient for the value function term is computed using the Model Gradient. The transitions used to compute the Model Gradient term must be sampled under the current policy and model (Equation 5). Therefore, to estimate the Model Gradient term, we generate a minibatch of transitions by simulating the current policy in $\widehat{T}_\phi$.

Like previous works [10, 76, 75, 8], we represent the dynamics model using an ensemble of neural networks. Each neural network produces a Gaussian distribution over the next state and reward:

$\widehat{T}_\phi(s', r|s, a) = \mathcal{N}(\mu_\phi(s, a), \Sigma_\phi(s, a))$. A visualisation of the result of adversarially training the dynamics model can be found in Appendix C.3.

Normalisation The composite loss function in Equation 9 comprises two terms which may have different magnitudes across domains depending on the scale of the states and rewards. To enable easier tuning of the adversarial loss weighting, λ, across different domains we perform the following normalisation procedure. Prior to training, we normalise the states in $\mathcal{D}$ in the manner proposed in [15], by subtracting the mean and dividing by the standard deviation of each state dimension in the dataset. Additionally, when computing the gradient in Equation 5 we normalise the advantage terms, $r + \gamma V^\pi_\phi(s') - Q^\pi_\phi(s, a)$, according to the mean and standard deviation across each minibatch. Advantage normalisation is common practice in policy gradient RL implementations [3, 52].

5.3 Algorithm

We are now ready to present our overall approach in Algorithm 1. The first step of RAMBO is to pretrain the environment dynamics model using standard MLE (Line 1). Thereafter, the algorithm follows the format of RARL. At each iteration, we apply gradient updates to the agent to increase the expected value, followed by gradient updates to the model to decrease the expected value.

Prior to each agent update, we generate synthetic k-step rollouts starting from states in $\mathcal{D}$ by simulating rollouts in the current MDP model $\widehat{T}_\phi$. This data is added to the synthetic dataset $\mathcal{D}_{\widehat{T}_\phi}$ (Line 3). Following previous approaches [76, 75, 22] we only store data from recent iterations in $\mathcal{D}_{\widehat{T}_\phi}$. The agent's policy and value functions are trained with an off-policy actor-critic algorithm using samples from $\mathcal{D} \cup \mathcal{D}_{\widehat{T}_\phi}$ (Line 4). In our implementation, we use soft actor-critic (SAC) [19] for agent training. To update the model to minimise the loss in Equation 9 (Line 5), we sample data from $\mathcal{D}$ to estimate the gradient for the MLE component. To compute the adversarial component we generate samples by simulating the current policy and model, and utilise the value function learnt by the agent to compute the gradient according to Equation 5.

Algorithm 1 RAMBO-RL

Require: Normalised dataset, $\mathcal{D}$;

1: $\widehat{T}_\phi \leftarrow$ MLE dynamics model.
2: **for** $i = 1, 2, \ldots, n_{\text{iter}}$ **do**
3: Generate synthetic k-step rollouts. Add transition data to $\mathcal{D}_{\widehat{T}_\phi}$.
4: *Agent update*: Update π and Q^π_ϕ with an actor critic algorithm, using samples from $\mathcal{D} \cup \mathcal{D}_{\widehat{T}_\phi}$.
5: *Adversarial model update*: Update $\widehat{T}_\phi$ according to Eq. 9, using samples from $\mathcal{D}$ for the MLE component, and the current critic Q^π_ϕ and synthetic data sampled from π and $\widehat{T}_\phi$ for the adversarial component.

6 Experiments

In our experiments, we aim to: a) evaluate how well RAMBO performs compared to state-of-the-art baselines, b) examine whether RAMBO can be tuned offline, c) determine the impact of adversarial training on the performance of the algorithm, and d) investigate the difference between RAMBO and COMBO, the most similar prior algorithm. The code for our experiments is available at github.com/marc-rigter/rambo. We evaluate our approach on the following domains.

MuJoCo There are three different environments representing different robots (*HalfCheetah, Hopper, Walker2D*), each with 4 datasets (*Random, Medium, Medium-Replay, Medium-Expert*). *Random* contains transitions collected by a random policy. *Medium* contains transitions collected by an early-stopped SAC policy. *Medium-Replay* consists of the replay buffer generated while training the *Medium* policy. The *Medium-Expert* dataset contains a mixture of suboptimal and expert data.

AntMaze The agent controls a robot and navigates to reach a goal, receiving a sparse reward only if the goal is reached. There are three different layouts of maze (*Umaze, Medium, Large*), and different dataset types (*Fixed, Play, Diverse*) which differ in terms of the variety of start and goal locations used to collect the dataset. The MuJoCo and AntMaze benchmarks are from D4RL [14].

Hyperparameter Details The base hyperparameters that we use for RAMBO mostly follow those used in SAC [19] and COMBO [75]. We find that the performance of RAMBO is sensitive to the choice of rollout length, k, consistent with findings in previous works [22, 37]. The other critical parameter for RAMBO is the choice of the adversarial weighting, λ.

For each dataset, we choose the rollout length and the adversarial weighting from one of three possible configurations: $(k, \lambda) \in \{(2, 3\text{e-}4), (5, 3\text{e-}4), (5, 0)\}$. We included $(k, \lambda) = (5, 0)$ as we found that an adversarial weighting of 0 worked well for some datasets. For the MuJoCo datasets we performed model rollouts using the current policy, and initialised the policy using behaviour cloning (BC) which is a common practice in offline RL [25, 74]. Ablation results in Appendix C.4 indicate that the BC initialisation results in a small improvement. For the AntMaze datasets we used a random rollout policy and a randomly initialised policy as we found that this performed better. Further details about the hyperparameters are in Appendix B.

Evaluation We present two different evaluations of our approach: RAMBO and RAMBO^{OFF}. For RAMBO, we ran each of the three hyperparameter configurations for five seeds each, and report the best performance across the three configurations. Thus, our evaluation of RAMBO utilises limited online tuning which is the most common practice among existing model-based offline RL algorithms [25, 37, 39, 76]. The performance obtained for each of the hyperparameter configurations is included in Appendix C.2.

Offline hyperparameter selection is an important topic in offline RL [47, 77]. Therefore, we present additional results for RAMBO^{OFF} where we select between the three choices of hyperparameters offline using a simple heuristic (details in Appendix B.5) based on the magnitude and stability of the Q-values during offline training. We first select the hyperparameters using the heuristic, and then rerun each dataset for 5 seeds to generate the results for RAMBO^{OFF}.

Baselines We compare RAMBO against state-of-the-art model-based (COMBO [75], RepB-SDE [34], MOReL [25], and MOPO [76]) and model-free (CQL [31], IQL [28], and TD3+BC [15]) offline RL algorithms. We provide results for all algorithms for the MuJoCo-v2 D4RL datasets and the AntMaze-v0 datasets (details in Appendix B.7).

6.1 Results

D4RL Performance The results in Table 1 show that RAMBO achieves the best total score for the MuJoCo locomotion domains, outperforming existing state-of-the-art methods. Furthermore, RAMBO obtains the best overall score on both the Medium and Medium-Replay dataset types. For the random datasets, RAMBO is outperformed only by MOReL. This shows that RAMBO performs very well for datasets that are either noisy or consist of suboptimal data.

For the Medium-Expert datasets, RAMBO is outperformed by most of the baseline algorithms, suggesting that RAMBO is less suitable for high-quality datasets. However, for the Medium-Expert datasets, simpler approaches such as performing behaviour cloning on the best 10% of trajectories can be used to achieve stronger performance than offline RL methods [28]. Therefore, the suboptimal performance of RAMBO on the Medium-Expert datasets is less of a concern, as applying offline RL algorithms may not be the most suitable approach for these high-quality datasets.

For AntMaze, the model-based algorithms perform considerably less well than the model-free approaches. Unlike the other model-based approaches, RAMBO at least scores greater than zero for most of the datasets. Our results echo previous findings that model-based approaches struggle to perform well in the AntMaze domains, potentially because model-based algorithms are too aggressive and collide with walls [67]. Recent work [67] showed that *reverse* rollouts can lead to stronger performance for model-based methods in these domains. In future work, we wish to investigate whether combining RAMBO with reverse rollouts improves the performance for AntMaze.

Offline Tuning In Table 1, we also present results for RAMBO^{OFF} where the final hyperparameters are chosen using the heuristic described in Appendix B.5. We see that there is a slight degradation in the performance relative to RAMBO, which uses online tuning. However, RAMBO^{OFF} still achieves comparable performance to the best existing approaches on the MuJoCo datasets. This suggests that suitable hyperparameters for RAMBO can reliably be chosen using our offline heuristic.

Table 1: Results for the D4RL benchmark using the normalisation procedure proposed by [14]. We report the normalised performance during the last 10 iterations of training averaged over 5 seeds. $\pm$ captures the standard deviation over seeds. Highlighted numbers indicate results within 2% of the most performant algorithm. * indicates the total without random datasets.

		Ours		Model-based baselines				Model-free baselines			
		RAMBO	**RAMBO^{OFF}**	**RepB-SDE**	**COMBO**	**MOPO**	**MOReL**	**CQL**	**IQL**	**TD3+BC**	**BC**
Random	HalfCheetah	40.0 ± 2.3	33.5 ± 2.6	32.9	38.8	35.4	25.6	19.6	-	11.0	2.1
	Hopper	21.6 ± 8.0	15.5 ± 9.4	8.6	17.9	4.1	53.6	6.7	-	8.5	9.8
	Walker2D	11.5 ± 10.5	0.2 ± 0.6	21.1	7.0	4.2	37.3	2.4	-	1.6	1.6
Medium	HalfCheetah	77.6 ± 1.5	71.0 ± 3.0	49.1	54.2	69.5	42.1	49.0	47.4	48.3	36.1
	Hopper	92.8 ± 6.0	91.2 ± 16.3	34.0	94.9	48.0	95.4	66.6	66.3	59.3	29.0
	Walker2D	86.9 ± 2.7	89.1 ± 2.7	72.1	75.5	-0.2	77.8	83.8	78.3	83.7	6.6
Medium Replay	HalfCheetah	68.9 ± 2.3	67.0 ± 1.5	57.5	55.1	68.2	40.2	47.1	44.2	44.6	38.4
	Hopper	96.6 ± 7.0	97.6 ± 3.4	62.2	73.1	39.1	93.6	97.0	94.7	60.9	11.8
	Walker2D	85.0 ± 15.0	88.5 ± 4.0	49.8	56.0	69.4	49.8	88.2	73.9	81.8	11.3
Medium Expert	HalfCheetah	93.7 ± 10.5	79.3 ± 2.9	55.4	90.0	72.7	53.3	90.8	86.7	90.7	35.8
	Hopper	83.3 ± 9.1	89.5 ± 11.1	82.6	111.1	3.3	108.7	106.8	91.5	98.0	111.9
	Walker2D	68.3 ± 20.6	63.1 ± 31.3	88.8	96.1	-0.3	95.6	109.4	109.6	110.1	6.4
MuJoCo-v2 Total:		826.2 ± 33.8	785.5 ± 40.4	614.1	769.7	413.4	773.0	767.4	692.6*	698.5	300.8
AntMaze	Umaze	25.0 ± 12.0	23.8 ± 15.0	0.0	80.3	0.0	0.0	74.0	87.5	78.6	65.0
	Medium-Play	16.4 ± 17.9	5.6 ± 10.9	0.0	0.0	0.0	0.0	61.2	71.2	3.0	0.0
	Large-Play	0.0 ± 0.0	0.0 ± 0.0	0.0	0.0	0.0	0.0	15.8	39.6	0.0	0.0
	Umaze-Diverse	0.0 ± 0.0	0.0 ± 0.0	0.0	57.3	0.0	0.0	84.0	62.2	71.4	55.0
	Medium-Diverse	23.2 ± 14.2	8.4 ± 9.9	0.0	0.0	0.0	0.0	53.7	70.0	10.6	0.0
	Large-Diverse	2.4 ± 3.3	0.0 ± 0.0	0.0	0.0	0.0	0.0	14.9	47.5	0.2	0.0
AntMaze-v0 Total:		67.0 ± 14.9	37.8 ± 12.4	0.0	137.6	0.0	0.0	303.6	378.0	163.8	120.0

Table 2: Ablation of the adversarial updates for RAMBO. These results use the same rollout length for each dataset as RAMBO but with no adversarial updates (i.e. $\lambda = 0$). The scores are averaged over 5 seeds.

RAMBO (No Adversarial Training)	
MuJoCo-v2 Total: 694.4 ± 56.5	**AntMaze-v0 Total:** 45.8 ± 21.8

Table 3: Comparison between RAMBO and COMBO for the Single Transition Example. We use 20 seeds and $\pm$ captures the standard deviation over seeds. RAMBO outperforms COMBO (p = 0.005).

RAMBO: $\mathbf{1.49 \pm 0.05}$	**COMBO:** 1.41 ± 0.12

Ablation of Adversarial Training In Table 2 we present results for RAMBO with no adversarial updates. These results demonstrate that overall performance degrades if the adversarial training is removed. This parallels previous findings that mitigating the issue of model exploitation is crucial to obtaining strong performance in model-based offline RL. Interestingly however, for some specific datasets we obtain the best performance with no adversarial training (Appendix C.2). This suggests that a potential direction for future work could be trying to identify which types of problems do not require regularisation for a successful policy to be trained offline with model-based RL.

Comparison to COMBO We focus especially on comparing our approach to COMBO, as it is the most similar existing algorithm. We compare RAMBO and COMBO on the Single Transition toy example which is described in detail in Appendix C.1. This domain has a one-dimensional state and action space, and several distinct regions of the action space are covered by the dataset. We use this domain to investigate whether the policies optimised by RAMBO and COMBO tend to become stuck in local optima.

Table 3 compares the performance of RAMBO and COMBO on the Single Transition Example. Further analysis in Appendix C.1 shows that for this problem, the pessimistic value function updates used by COMBO create local maxima in the Q-function which are present throughout training. Policy optimisation can become stuck in these local maxima. On the other hand, the value function produced by RAMBO is initially optimistic, and pessimism is introduced into the value function *gradually* as the transition function is modified adversarially. Adversarial modification of the transition function is visualised in Appendix C.3. As a result of this gradual introduction of pessimism, we observe that the

policy produced by RAMBO is less likely to become stuck in poor local maxima, and better overall performance is obtained in Table 3. This observation may help to explain why RAMBO is able to achieve consistently strong performance relative to existing algorithms in the MuJoCo domains. Gradually increasing the level of pessimism could be a useful modification for existing offline RL algorithms to be investigated in future work.

7 Conclusion and Future Directions

RAMBO is a promising new approach to offline RL which imposes conservatism by adversarially modifying the transition dynamics of a learnt model. Our approach is theoretically justified, and achieves state-of-the-art performance on standard benchmarks.

There are a number of possible extensions to RAMBO, some of which we have already discussed. In addition, we would like to apply RAMBO to image-space domains by using deep latent variable models to compress the state space [18, 51] and adversarially perturbing the transition dynamics in the latent space representation. Another direction that we would like to investigate is the use of adversarially trained models to aid interpretability in deep RL [44] by generating imagined worst-case trajectories. Finally, we wish to investigate applying the ideas developed in this work to the online RL setting to improve robustness.

Acknowledgements

This work was supported by a Programme Grant from the Engineering and Physical Sciences Research Council (EP/V000748/1), the Clarendon Fund at the University of Oxford, and a gift from Amazon Web Services. Additionally, this project made use of time on Tier 2 HPC facility JADE2, funded by the Engineering and Physical Sciences Research Council (EP/T022205/1).

The authors would like to thank Raunak Bhattacharyya, Paul Duckworth, and Matthew Budd for their feedback on an earlier draft of this work.

References

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

[2] Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified Q-ensemble. *Advances in Neural Information Processing Systems*, 34:7436–7447, 2021.

[3] Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Leonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters for on-policy deep actor-critic methods? A large-scale study. In *International Conference on Learning Representations*, 2021.

[4] Arthur Argenson and Gabriel Dulac-Arnold. Model-based offline planning. In *International Conference on Learning Representations*, 2021.

[5] J Andrew Bagnell, Andrew Y Ng, and Jeff G Schneider. Solving uncertain Markov decision processes. Technical report, 2001.

[6] Philip J Ball, Cong Lu, Jack Parker-Holder, and Stephen Roberts. Augmented world models facilitate zero-shot dynamics generalization from a single offline environment. In *International Conference on Machine Learning*, pages 619–629. PMLR, 2021.

[7] David Brandfonbrener, Will Whitney, Rajesh Ranganath, and Joan Bruna. Offline RL without off-policy evaluation. *Advances in Neural Information Processing Systems*, 34:4933–4946, 2021.

[8] Catherine Cang, Aravind Rajeswaran, Pieter Abbeel, and Michael Laskin. Behavioral priors and dynamics models: Improving performance and domain transfer in offline RL. In *Deep RL Workshop NeurIPS 2021*, 2021.

[9] Ching-An Cheng, Tengyang Xie, Nan Jiang, and Alekh Agarwal. Adversarially trained actor critic for offline reinforcement learning. 2022.

[10] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in Neural Information Processing Systems*, 31, 2018.

[11] Ignasi Clavera, Violet Fu, and Pieter Abbeel. Model-augmented actor-critic: Backpropagating through paths. *International Conference on Learning Representations*, 2020.

[12] Michael K Cohen and Marcus Hutter. Pessimism about unknown unknowns inspires conservatism. In *Conference on Learning Theory*, pages 1344–1373. PMLR, 2020.

[13] Sebastian Curi, Ilija Bogunovic, and Andreas Krause. Combining pessimism with optimism for robust and efficient model-based deep reinforcement learning. In *International Conference on Machine Learning*, pages 2254–2264. PMLR, 2021.

[14] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[15] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[16] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062. PMLR, 2019.

[17] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. A survey of uncertainty in deep neural networks. *arXiv preprint arXiv:2107.03342*, 2021.

[18] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in Neural Information Processing Systems*, 31, 2018.

[19] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018.

[20] Toru Hishinuma and Kei Senda. Weighted model estimation for offline model-based reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[21] Garud N Iyengar. Robust dynamic programming. *Mathematics of Operations Research*, 30(2):257–280, 2005.

[22] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

[23] 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.

[24] Kirthevasan Kandasamy, Yoram Bachrach, Ryota Tomioka, Daniel Tarlow, and David Carter. Batch policy gradient methods for improving neural conversation models. *International Conference on Learning Representations*, 2017.

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

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

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

[28] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations*, 2022.

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

[30] Aviral Kumar, Anikait Singh, Stephen Tian, Chelsea Finn, and Sergey Levine. A workflow for offline model-free robotic reinforcement learning. *Conference on Robot Learning*, 2021.

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

[32] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. In *International Conference on Learning Representations*, 2018.

[33] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer, 2012.

[34] Byung-Jun Lee, Jongmin Lee, and Kee-Eung Kim. Representation balancing offline model-based reinforcement learning. In *International Conference on Learning Representations*, 2020.

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

[36] Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Off-policy policy gradient with state distribution correction. *International Conference on Machine Learning RL4RealLife Workshop*, 2019.

[37] Cong Lu, Philip Ball, Jack Parker-Holder, Michael Osborne, and Stephen J Roberts. Revisiting design choices in offline model based reinforcement learning. In *International Conference on Learning Representations*, 2022.

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

[39] Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. Deployment-efficient reinforcement learning via model-based offline optimization. In *International Conference on Learning Representations*, 2021.

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

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

[42] Xinkun Nie, Emma Brunskill, and Stefan Wager. Learning when-to-treat policies. *Journal of the American Statistical Association*, 116(533):392–409, 2021.

[43] Arnab Nilim and Laurent El Ghaoui. Robust control of Markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.

[44] Michael Oberst and David Sontag. Counterfactual off-policy evaluation with gumbel-max structural causal models. In *International Conference on Machine Learning*, pages 4881–4890. PMLR, 2019.

[45] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand. *arXiv preprint*, 2019.

[46] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. *Advances in Neural Information Processing Systems*, 32, 2019.

[47] Tom Le Paine, Cosmin Paduraru, Andrea Michi, Caglar Gulcehre, Konrad Zolna, Alexander Novikov, Ziyu Wang, and Nando de Freitas. Hyperparameter selection for offline reinforcement learning. *arXiv preprint arXiv:2007.09055*, 2020.

[48] Xinlei Pan, Daniel Seita, Yang Gao, and John Canny. Risk averse robust adversarial reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8522–8528. IEEE, 2019.

[49] 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.

[50] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, pages 2817–2826. PMLR, 2017.

[51] Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Offline reinforcement learning from images with latent space models. In *Learning for Dynamics and Control*, pages 1154–1168. PMLR, 2021.

[52] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 2021.

[53] Aravind Rajeswaran, Igor Mordatch, and Vikash Kumar. A game theoretic framework for model based reinforcement learning. In *International Conference on Machine Learning*, pages 7953–7963. PMLR, 2020.

[54] Marc Rigter, Bruno Lacerda, and Nick Hawes. Minimax regret optimisation for robust planning in uncertain Markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11930–11938. Association for the Advancement of Artificial Intelligence, 2021.

[55] Aurko Roy, Huan Xu, and Sebastian Pokutta. Reinforcement learning under model mismatch. *Advances in Neural Information Processing Systems*, 30, 2017.

[56] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *International Conference on Learning Representations*, 2016.

[57] 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):109–136, 2011.

[58] Noah Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavior modelling priors for offline reinforcement learning. In *International Conference on Learning Representations*, 2020.

[59] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

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

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

[62] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 12, 1999.

[63] Phillip Swazinna, Steffen Udluft, and Thomas Runkler. Overcoming model bias for robust offline deep reinforcement learning. *Engineering Applications of Artificial Intelligence*, 104:104366, 2021.

[64] Aviv Tamar, Shie Mannor, and Huan Xu. Scaling up robust MDPs using function approximation. In *International Conference on Machine Learning*, pages 181–189. PMLR, 2014.

[65] Chen Tessler, Yonathan Efroni, and Shie Mannor. Action robust reinforcement learning and applications in continuous control. In *International Conference on Machine Learning*, pages 6215–6224. PMLR, 2019.

[66] Masatoshi Uehara and Wen Sun. Pessimistic model-based offline reinforcement learning under partial coverage. *International Conference on Learning Representations*, 2022.

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

[68] Yue Wang and Shaofeng Zou. Online robust reinforcement learning with model uncertainty. *Advances in Neural Information Processing Systems*, 34, 2021.

[69] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[70] Wolfram Wiesemann, Daniel Kuhn, and Berç Rustem. Robust Markov decision processes. *Mathematics of Operations Research*, 38(1):153–183, 2013.

[71] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

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

[73] Tengyang Xie, Ching-An Cheng, Nan Jiang, Paul Mineiro, and Alekh Agarwal. Bellman-consistent pessimism for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[74] Yijun Yang, Jing Jiang, Tianyi Zhou, Jie Ma, and Yuhui Shi. Pareto policy pool for model-based offline reinforcement learning. In *International Conference on Learning Representations*, 2021.

[75] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. COMBO: Conservative offline model-based policy optimization. *Advances in Neural Information Processing Systems*, 34, 2021.

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

[77] Siyuan Zhang and Nan Jiang. Towards hyperparameter-free policy selection for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We included results for the AntMaze datasets where our approach performs less well than model-free methods.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] We include a complete proof of Proposition 2. For the theoretical results from [66], we provide references to the appropriate parts of that paper.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] A URL to the code is provided.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Detailed information about implementation details and hyperparameters is in the appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Standard deviation with respect to seeds is reported for all results.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Details of compute used per run and total compute used for full evaluation is in Appendix B.8.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cite the D4RL benchmark datasets [14].
   (b) Did you mention the license of the assets? [Yes] In Appendix B.7.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Code for our work is in included via a URL.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendices

A Proof of Proposition 2

Proposition 2 (Model Gradient)**.** *Let ϕ denote the parameters of a parametric MDP model $\widehat{T}_\phi$, and let V_ϕ^π denote the value function for policy π in $\widehat{T}_\phi$. Then:*

$$\nabla_\phi V_\phi^\pi = \mathbb{E}_{s\sim d_\phi^\pi, a\sim\pi, (s',r)\sim\widehat{T}_\phi}\big[(r+\gamma V_\phi^\pi(s'))\cdot\nabla_\phi \log \widehat{T}_\phi(s',r|s,a)\big] \tag{4}$$

Proof: The proof is analogous to the proof of the policy gradient theorem [61, 62]. We start by expressing the value function using Bellman's equation, for some state s:

$$\begin{aligned} V_\phi^\pi(s) &= \sum_a \pi(a|s) Q_\pi(s,a) \\ &= \sum_a \pi(a|s) \sum_{s',r} (r+\gamma V_\phi^\pi(s'))\cdot \widehat{T}_\phi(s',r|s,a). \end{aligned} \tag{10}$$

Applying the product rule:

$$\begin{aligned} \nabla_\phi V_\phi^\pi(s) &= \sum_a \pi(a|s) \sum_{s',r} \Big[(r+\gamma V_\phi^\pi(s'))\cdot\nabla_\phi \widehat{T}_\phi(s',r|s,a) + \widehat{T}_\phi(s',r|s,a)\cdot\nabla_\phi(r+\gamma V_\phi^\pi(s'))\Big] \\ &= \sum_a \pi(a|s) \sum_{s',r} (r+\gamma V_\phi^\pi(s'))\cdot\nabla_\phi \widehat{T}_\phi(s',r|s,a) + \gamma \sum_a \pi(a|s) \sum_{s'} \widehat{T}_\phi(s'|s,a)\cdot\nabla_\phi V_\phi^\pi(s') \end{aligned} \tag{11}$$

Define $\psi(s) := \sum_a \pi(a|s) \sum_{s',r} (r+\gamma V_\phi^\pi(s'))\cdot\nabla_\phi \widehat{T}_\phi(s',r|s,a)$, and additionally define $\rho_\phi^\pi(s\to x, n)$ as the probability of transitioning from state s to x by executing π for n steps in the MDP model defined by $\widehat{T}_\phi$. Using these definitions, we can rewrite the above equation as:

$$\begin{aligned} \nabla_\phi V_\phi^\pi(s) &= \psi(s) + \gamma \sum_{s'} \rho_\phi^\pi(s\to s',1)\cdot\nabla_\phi V_\phi^\pi(s') \\ &= \psi(s) + \gamma \sum_{s'} \rho_\phi^\pi(s\to s',1)\big[\psi(s') + \gamma \sum_{s''} \rho_\phi^\pi(s'\to s'',1)\cdot\nabla_\phi V_\phi^\pi(s'')\big] \\ &= \psi(s) + \gamma \sum_{s'} \rho_\phi^\pi(s\to s',1)\psi(s') + \gamma^2 \sum_{s''} \rho_\phi^\pi(s\to s'',2)\cdot\nabla_\phi V_\phi^\pi(s'') \\ &= \sum_{s'} \sum_{t=0}^{\infty} \gamma^t \rho_\phi^\pi(s\to s',t)\psi(s') \qquad \text{(continuing to unroll)} \end{aligned} \tag{12}$$

We have that $V_\phi^\pi = \sum_s \mu_0(s)\cdot V_\phi^\pi(s)$ by definition. Therefore,

$$\begin{aligned} \nabla_\phi V_\phi^\pi &= \nabla_\phi\big(\sum_s \mu_0(s)\cdot V_\phi^\pi(s)\big) \\ &= \sum_s \mu_0(s)\cdot\nabla_\phi V_\phi^\pi(s) \\ &= \sum_s \mu_0(s) \sum_{s'} \sum_{t=0}^{\infty} \gamma^t \rho_\phi^\pi(s\to s',t)\psi(s') \\ &= \sum_s d_\phi^\pi(s)\psi(s) \end{aligned} \tag{13}$$

where $d_\phi^\pi(s)$ is the (improper) discounted state visitation distribution of the policy π in the MDP model $\widehat{T}_\phi$, under initial state distribution μ_0. Finally, we apply the log-derivative trick to arrive at the final result:

$$\begin{aligned}
\nabla_\phi V_\phi^\pi &= \sum_s d_\phi^\pi(s) \sum_a \pi(a|s) \sum_{s',r} (r + \gamma V_\phi^\pi(s')) \cdot \nabla_\phi \widehat{T}_\phi(s', r|s, a) \\
&= \sum_s d_\phi^\pi(s) \sum_a \pi(a|s) \sum_{s',r} \widehat{T}_\phi(s', r|s, a)(r + \gamma V_\phi^\pi(s')) \frac{\nabla_\phi \widehat{T}_\phi(s', r|s, a)}{\widehat{T}_\phi(s', r|s, a)} \\
&= \mathbb{E}_{s\sim d_\phi^\pi, a\sim\pi, (s',r)\sim\widehat{T}_\phi} \left[(r + \gamma V_\phi^\pi(s')) \cdot \nabla_\phi \log \widehat{T}_\phi(s', r|s, a) \right] \quad \square
\end{aligned} \tag{14}$$

A.1 Model Gradient with State-Action Value Baseline

We can subtract $Q_\phi^\pi(s, a)$ as a baseline without biasing the gradient estimate:

$$\nabla_\phi V_\phi^\pi = \mathbb{E}_{s\sim d_\phi^\pi, a\sim\pi, (s',r)\sim\widehat{T}_\phi} \left[(r + \gamma V_\phi^\pi(s') - Q_\phi^\pi(s, a)) \cdot \nabla_\phi \log \widehat{T}_\phi(s', r|s, a) \right] \tag{15}$$

Proof: We can subtract any quantity which does not depend on s' or r without changing the value of the expression because the subtracted quantity is zero. Specifically, for the baseline of $Q_\phi^\pi(s, a)$ we have that:

$$\begin{aligned}
&\mathbb{E}_{s\sim d_\phi^\pi, a\sim\pi, (s',r)\sim\widehat{T}_\phi} \left[Q_\phi^\pi(s, a) \cdot \nabla_\phi \log \widehat{T}_\phi(s', r|s, a) \right] \\
&\quad = \sum_s d_\phi^\pi(s) \sum_a \pi(a|s) \sum_{s',r} \widehat{T}_\phi(s', r|s, a) \cdot Q(s, a) \cdot \nabla_\phi \log \widehat{T}_\phi(s', r|s, a) \\
&\quad = \sum_s d_\phi^\pi(s) \sum_a \pi(a|s) \cdot Q(s, a) \cdot \sum_{s',r} \nabla_\phi \widehat{T}_\phi(s', r|s, a) \\
&\quad = \sum_s d_\phi^\pi(s) \sum_a \pi(a|s) \cdot Q(s, a) \cdot \nabla_\phi \sum_{s',r} \widehat{T}_\phi(s', r|s, a) \\
&\quad = \sum_s d_\phi^\pi(s) \sum_a \pi(a|s) \cdot Q(s, a) \cdot \nabla_\phi(1) \\
&\quad = 0
\end{aligned} \tag{16}$$

B Implementation Details

The code for RAMBO is available at github.com/marc-rigter/rambo. Our implementation builds upon the official MOPO codebase.

B.1 Model Training

We represent the model as an ensemble of neural networks that output a Gaussian distribution over the next state and reward given the current state and action:

$$\widehat{T}_\phi(s', r|s, a) = \mathcal{N}(\mu_\phi(s, a), \Sigma_\phi(s, a)).$$

Following previous works [75, 76], during the initial maximum likelihood model training (Line 1 of Algorithm 1) we train an ensemble of 7 such dynamics models and pick the best 5 models based on the validation error on a held-out test set of 1000 transitions from the dataset $\mathcal{D}$. Each model in the ensemble is a 4-layer feedforward neural network with 200 hidden units per layer.

After the maximum likelihood training, RAMBO updates all of these 5 models adversarially according to Algorithm 1. For each model rollout, we randomly pick one of the 5 dynamics models to simulate

Table 4: Base hyperparameter configuration which is shared across all runs of RAMBO.

	Hyperparameter	Value
SAC	critic learning rate	3e-4
	actor learning rate	1e-4
	discount factor (γ)	0.99
	soft update parameter (τ)	5e-3
	target entropy	-dim(A)
	batch size	256
MBPO	no. of model networks	7
	no. of elites	5
	model learning rate	3e-4
	ratio of real data (f)	0.5
	model training batch size	256
	number of iterations (n_{iter})	2000

the rollout. The learning rate for the model training is 3e-4 for both the MLE pretraining, and the adversarial training. The model is trained using the Adam optimiser [26].

For the MLE pretraining, the batch size is 256. For each adversarial update in Equation 9 we sample a batch of 256 transitions from $\mathcal{D}$ to compute the maximum likelihood term, and 256 synthetic transitions for the model gradient term. The hyperparameters used for model training are summarised in Table 4.

B.2 Policy Training

We sample a batch of 256 transitions to train the policy and value function using soft actor-critic (SAC) [19]. We set the ratio of real data to $f = 0.5$ for all datasets, meaning that we sample 50% of the batch transitions from $\mathcal{D}$ and the remaining 50% from $\mathcal{D}_{\widehat{T}_\phi}$. We chose this value because it was used for most datasets by COMBO [75] and we found that it worked well. We represent the Q-networks and the policy as three layer neural networks with 256 hidden units per layer.

For SAC [19] we use automatic entropy tuning, where the entropy target is set to the standard heuristic of $-\text{dim}(A)$. The only hyperparameter that we modify from the standard implementation of SAC is the learning rate for the actor, which we set to 1e-4 as this was reported to work better in the CQL paper [31] which also utilised SAC. For reference, the hyperparameters used for SAC are included in Table 4.

B.3 RAMBO Implementation Details

For each iteration of RAMBO we perform 1000 gradient updates to the actor-critic algorithm, and 1000 adversarial gradient updates to the model. For each run, we perform 2000 iterations. The base hyperparameter configuration for RAMBO that is shared across all runs is shown in Table 4.

The only parameters that we vary between datasets are the length of the synthetic rollouts, k, the weighting of the adversarial term in the model update, λ, and the choice of rollout policy. Details are included in the Hyperparameter Tuning section below.

B.4 Hyperparameter Tuning

We found that the hyperparameters that have the largest influence on the performance of RAMBO are the length of the synthetic rollouts (k), and the weighting of the adversarial term in the model update (λ). For the rollout length, we found that a value of either 2 or 5 worked well across most datasets. This is a slight modification to the values of $k \in \{1, 5\}$ that are typically used in previous model-based offline RL algorithms [8, 75, 76].

To arrive at a suitable value for the adversarial weighting, we used the intuition that we must have $\lambda \ll 1$. This is necessary so that the MLE term dominates in the loss function in Equation 9 for in-distribution samples. This ensures that the model still fits the dataset accurately despite the adversarial

Table 5: Hyperparameters used by RAMBO and RAMBO^{OFF} for each dataset. k is the rollout length, and λ is the adversary loss weighting. The hyperparameters for RAMBO are those with the best performance from the three configurations tested (see Table 6). The hyperparameters for RAMBO^{OFF} are those which obtained the lowest value of the offline heuristic in Table 6. As indicated, a random rollout policy was used for some of the AntMaze domains as we found that this performed better.

		RAMBO		RAMBO^{OFF}		
		k	λ	k	λ	Rollout Policy
Random	HalfCheetah	5	0	2	3e-4	Current Policy
	Hopper	2	3e-4	5	0	
	Walker2D	5	0	5	3e-4	
Medium	HalfCheetah	5	3e-4	2	3e-4	
	Hopper	5	3e-4	5	3e-4	
	Walker2D	5	0	5	0	
Medium Replay	HalfCheetah	5	3e-4	5	0	
	Hopper	2	3e-4	2	3e-4	
	Walker2D	5	0	5	0	
Medium Expert	HalfCheetah	5	3e-4	2	3e-4	
	Hopper	5	3e-4	5	3e-4	
	Walker2D	2	3e-4	2	3e-4	
AntMaze	Umaze	5	3e-4	5	3e-4	Current Policy
	Medium-Play	5	0	5	3e-4	Uniform Random
	Large-Play	5	3e-4	5	3e-4	
	Umaze-Diverse	5	0	5	0	
	Medium-Diverse	5	0	2	3e-4	
	Large-Diverse	5	0	5	0	

training. We observed that if λ is set too high, the training can be unstable and the Q-function can become extremely negative. If λ is too small, the adversarial training has no effect as the model is updated too slowly. We found that a value of 3e-4 generally obtained good performance.

For all MuJoCo datasets we performed model rollouts using the current policy, but for some AntMaze datasets we used a random rollout policy as we found this performed better. The rollout policies used are listed in Table 5. For the MuJoCo datasets, we initialised the policy using behaviour cloning (BC) which is common practice in offline RL [25, 74]. Ablation results without the BC initialisation are in Appendix C.4. We did not use the BC initialisation for the AntMaze problems.

For the rollout length and the adversarial weighting, we tested the following three configurations on each dataset: $(k, \lambda) \in \{$(2, 3e-4), (5, 3e-4), (5, 0)$\}$. We included $(k, \lambda) = (5,\ 0)$ as we found that an adversarial weighting of 0 worked well for some datasets. As described in Section 6, to evaluate RAMBO we ran each of the three hyperparameter configurations for five seeds each, and reported the best performance across the three configurations. The results for the best configuration are in Table 1, and the corresponding final hyperparameters chosen can be found in Table 5. The results for each of the three configurations can be found in Table 6.

Offline selection of hyperparameters is an important topic in offline RL [47, 77]. This is because in some applications the selection of hyperparameters based on online performance may be infeasible. Therefore, we present additional results in Table 1 where we select between the three choices of hyperparameters offline using a simple heuristic based on the magnitude and stability of the Q-values during training. After selecting the hyperparameters using the heuristic, we ran a further five seeds using these parameters to produce the final result. We refer to this approach as RAMBO^{OFF}. The hyperparameters used by RAMBO^{OFF} are also included in Table 5. Details of the heuristic used to select the hyperparameter configuration for RAMBO^{OFF} can be found in the following subsection.

B.5 Offline Hyperparameter Selection Method

Deep RL algorithms are highly sensitive to the choice of hyperparameters [3]. There is no consensus on how parameters should be selected for offline RL [47]. Some possibilities include a) selection based on online evaluation [8, 9, 25, 39, 72, 76], b) selection based on an offline heuristic [47, 75], and c) methods based on off-policy evaluation [47, 77].

To address applications in which online evaluation is possible, we evaluated RAMBO using approach a) which is the most common practice among model-based offline RL algorithms [8, 25, 37, 39, 76]. To consider situations where online tuning is not possible, we also evaluate using approach b) to select the hyperparameters for each dataset using a simple heuristic in a similar vein to [75]. We refer to this second approach as RAMBO^{OFF}.

To arrive at a heuristic that can be evaluated offline, we use the intuition that the value function should be regularised (i.e. low) as well as stable during training. Therefore, our heuristic makes the final hyperparameter selection from the set of candidate parameters according to: $\min\left(Q_{\text{avg}} + Q_{\text{var}}\right)$, where Q_{avg} is the average Q-value at the end of training, and Q_{var} is the variance of the average Q-value over the final 100 iterations of training.

The values of this heuristic for each of the three configurations that we test are the values in the brackets provided in Table 6. The bolded values in the table indicate the lowest value for the heuristic, which is the parameter configuration chosen for RAMBO^{OFF}. After choosing the hyperparameters in this manner, we *rerun* the algorithm for a further five seeds per dataset. The performance for these additional runs are the results reported in Table 1 for RAMBO^{OFF}.

The results in Table 1 indicate that RAMBO^{OFF} obtains strong performance compared to existing baselines on the MuJoCo domains. This indicates that it is possible to obtain solid performance with RAMBO by choosing the final hyperparameters according to the magnitude and stability of the Q-values. Unsurprisingly however, RAMBO performs slightly better than RAMBO^{OFF}. This shows that better performance is obtained using online hyperparameter tuning.

B.6 Evaluation Procedure

We report the average undiscounted normalized return averaged over the last 10 iterations of training, with 10 evaluation episodes per iteration. We evaluated the SAC policy by deterministically taking the mean action. The normalization procedure is that proposed by [14], where 100 represents an expert policy and 0 represents a random policy.

B.7 Baselines and Domains

We compare RAMBO against state of the art model-based (COMBO [75], MOReL [25], RepB-SDE [34] and MOPO [76]) and model-free (CQL [31], IQL [28], and TD3+BC [15]) offline RL algorithms on the D4RL benchmarks [14]. The D4RL benchmarks are released with the Apache License 2.0. To facilitate a fair comparison, we provide results for all algorithms for the MuJoCo-v2 D4RL datasets which contain more performant data than v0 [37].

Because the original COMBO, MOPO, and CQL papers use the MuJoCo-v0 datasets we report the results for these algorithms on the MuJoCo-v2 datasets from [67]. The MOReL, IQL, RepB-SDE, and TD3+BC papers include evaluations on the MuJoCo-v2 datasets, so for these algorithms we include the results from the original papers.

For AntMaze-v0, we report the results from the original papers for CQL, IQL, and TD3+BC. For COMBO, MOPO, RepB-SDE, and MOReL, the results are taken from [67]. Following the IQL paper [28], we subtract 1 from the rewards of the AntMaze datasets for our experiments.

B.8 Computational Resources

During our experiments, each run of RAMBO has access to 2 CPUs of an Intel Xeon Platinum 8259CL processor at 3.1GHz, and half of an Nvidia T4 GPU. With this hardware, each run of RAMBO takes 24-30 hours.

For our main evaluation of RAMBO, we ran five seeds for each of three parameter configurations for 18 tasks resulting in a total of 270 runs. This required approximately 150 GPU-days of compute.

C Additional Results

C.1 Single Transition Example

Description In this domain, the state and action spaces are one-dimensional. The agent executes a single action, $a \in [-1, 1]$, from initial state s_0. After executing the action, the agent transitions to s' and receives a reward equal to the successor state, i.e. $r(s') = s'$, and the episode terminates.

The actions in the dataset are sampled uniformly from $a \in [-0.75, 0.7] \cup [-0.15, -0.1] \cup [0.1, 0.15] \cup [0.7, 0.75]$. In the MDP for this domain, the transition distribution for successor states is as follows:

- $s' \sim \mathcal{N}(\mu = 1,\ \sigma = 0.2)$, for $a \in [-0.8,\ -0.65]$.
- $s' \sim \mathcal{N}(\mu = 0.5,\ \sigma = 0.2)$, for $a \in [-0.2,\ -0.05]$.
- $s' \sim \mathcal{N}(\mu = 1.25,\ \sigma = 0.2)$, for $a \in [0.05,\ 0.2]$.
- $s' \sim \mathcal{N}(\mu = 1.5,\ \sigma = 0.2)$, for $a \in [0.65,\ 0.8]$.
- $s' = 0.5$, for all other actions.

The transitions to successor states from the actions in the dataset are illustrated in Figure 1.

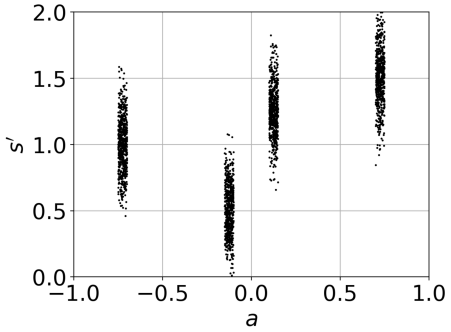


Figure 1: Transition data for the Single Transition Example after executing action a from s_0.

Comparison between RAMBO and COMBO To generate the comparison, we run both algorithms for 50 iterations. To choose the regularisation parameter for COMBO, we sweep over $\beta \in \{0.1, 0.2, 0.5, 1, 5\}$ and choose the parameter with the best performance. Note that 0.5, 1, and 5 are the values used in the original COMBO paper [75], so we consider lower values of regularisation than in the original paper. For RAMBO we use λ =3e-2. We used the implementation of COMBO from github.com/takuseno/d3rlpy.

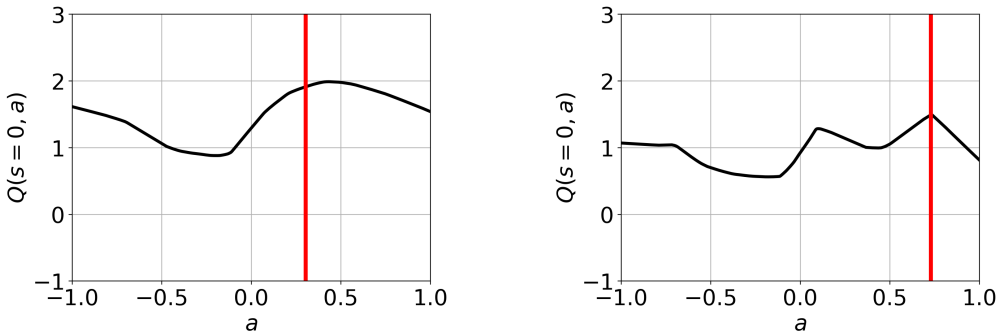
Figure 2 shows the Q-values produced by RAMBO throughout training. We can see that after 5 iterations, the Q-values for actions which are outside of the dataset are initially *optimistic*, and over-estimate the true values. However, after 50 iterations the Q-values for out of distribution actions have been regularised by RAMBO. The action selected by the policy after 50 iterations is the optimal action in the dataset, $a \in [0.7, 0.75]$. Thus, we see that RAMBO introduces pessimism *gradually* as the adversary is trained to modify the transitions to successor states.

For COMBO, the best performance averaged over 20 seeds was obtained for $\beta = 0.2$, and therefore we report results for this value of the regularisation parameter. Figure 3 illustrates the Q-values produced by COMBO throughout training (with $\beta = 0.2$). We see that at both 5 iterations and 50 iterations, the value estimates for actions outside of the dataset are highly pessimistic. In the run illustrated for COMBO, the action selected by the policy after 50 iterations is $a \in [0.1, 0.15]$, which is not the optimal action in the dataset. Figure 3 shows that the failure to find an optimal action is due to the gradient-based policy optimisation becoming stuck in a local maxima of the Q-function.

These results highlight a difference in the behaviour of RAMBO compared to COMBO. For RAMBO, pessimism is introduced gradually as the adversary is trained to modify the transitions to successor states. For COMBO, pessimism is introduced at the outset as it is part of the value function update

throughout the entirety of training. Additionally, the regularisation of the Q-values for out of distribution actions appears to be less aggressive for RAMBO than for COMBO.

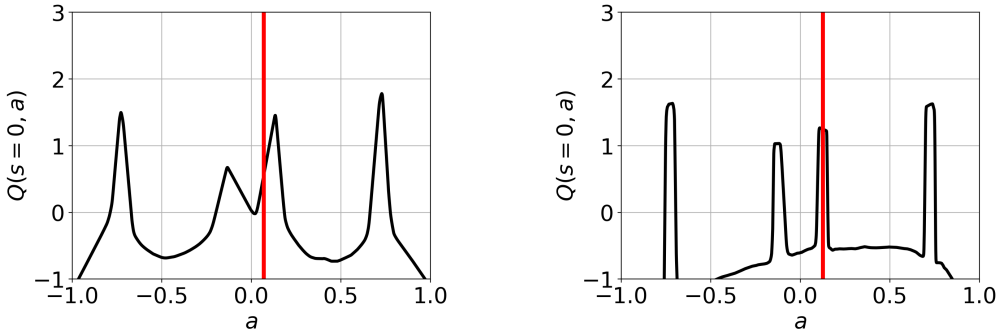
The results averaged over 20 seeds in Table 3 show that RAMBO consistently performs better than COMBO for this problem. This suggests that the gradual introduction of pessimism produced by RAMBO means that the policy optimisation procedure is less likely to get stuck in poor local maxima for this example. The downside of this behaviour is that it may take more iterations for RAMBO to find a performant policy. Modifying other algorithms to gradually introduce pessimism may be an interesting direction for future research.



(a) 5 iterations

(b) 50 iterations

Figure 2: Q-values at the initial state during the training of RAMBO on the Single Transition Example. The red line indicates the mean action of the policy.



(a) 5 iterations

(b) 50 iterations

Figure 3: Q-values at the initial state during the training of COMBO on the Single Transition Example ($\beta = 0.2$). The red line indicates the mean action of the policy.

C.2 Results for each Hyperparameter Configuration

In Table 6, we report the results for each of the three parameter configurations that we test. The highlighted values indicate the best performance for each dataset, and are the results reported for RAMBO in Table 1. In Table 6 we also report the values of the offline hyperparameter selection heuristic in brackets. The bold values indicate the lowest value of the heuristic, which is the hyperparameter configuration selected for RAMBO^{OFF}.

Table 6: Performance of RAMBO across each of the three configurations of the rollout length, k, and the adversarial weighting, λ, that we tested. Values indicate the average normalized return at the end of training. Each configuration was run for 5 seeds. $\pm$ captures the standard deviation over seeds. Highlighted values indicate the best performance for each dataset. The values in brackets indicate the value of the offline tuning heuristic described in Appendix B.5. "$-$" indicates that the value function diverged.

		$k = 2, \lambda = 3e-4$	$k = 5, \lambda = 3e-4$	$k = 5, \lambda = 0$
Random	HalfCheetah	34.5 ± 3.7 (**348.1**)	38.2 ± 3.9 (408.6)	40.0 ± 2.3 (377.6)
	Hopper	21.6 ± 8.0 (1179.7)	21.3 ± 9.2 (1293.2)	15.1 ± 9.4 (**188.9**)
	Walker2D	$-$	0.2 ± 0.5 (**1.2e8**)	11.5 ± 10.5 (2.5e9)
Medium	HalfCheetah	72.5 ± 4.4 (**671.1**)	77.6 ± 1.5 (717.4)	75.5 ± 6.0 (720.6)
	Hopper	$-$	92.8 ± 6.0 (**300.4**)	47.5 ± 36.0 (314.1)
	Walker2D	82.2 ± 11.4 (523.6)	76.8 ± 4.8 (2124.2)	86.9 ± 2.7 (**339.4**)
Medium Replay	HalfCheetah	65.4 ± 3.7 (623.5)	68.9 ± 2.3 (647.2)	64.7 ± 3.2 (**608.4**)
	Hopper	96.6 ± 7.0 (**262.4**)	93.5 ± 10.3 (309.1)	36.7 ± 9.5 (263.4)
	Walker2D	6.7 ± 2.5 (2.3e7)	62.1 ± 33.5 (4.3e6)	85.0 ± 15.0 (**259.0**)
Medium Expert	HalfCheetah	78.1 ± 6.6 (**987.1**)	93.7 ± 10.5 (1008.9)	92.9 ± 11.9 (1013.2)
	Hopper	36.4 ± 16.7 (344.1)	83.3 ± 9.1 (**342.8**)	77.6 ± 14.3 (344.4)
	Walker2D	68.3 ± 20.6 (**371.9**)	31.7 ± 49.8 (2.0e7)	50.8 ± 57.8 (6.9e9)
AntMaze	Umaze	8.8 ± 8.2 (-48.3)	25.0 ± 12.0 (**-54.2**)	3.8 ± 5.8 (-51.5)
	Medium-Play	5.8 ± 6.6 (1421)	8.4 ± 13.5 (**-70.77**)	16.4 ± 17.9 (-68.10)
	Large-Play	0.0 ± 0.0 (-77.9)	0.0 ± 0.0 (**-78.1**)	0.0 ± 0.0 (-77.9)
	Umaze-Diverse	0.0 ± 0.0 (790.8)	0.0 ± 0.0 (68.2)	0.0 ± 0.0 (**-65.4**)
	Medium-Diverse	3.8 ± 1.7 (**-72.4**)	1.6 ± 1.8 (-72.2)	23.2 ± 14.2 (-71.2)
	Large-Diverse	$-$	0.0 ± 0.0 (6.0e8)	2.4 ± 3.3 (**3.5e6**)

C.3 Visualisation of Adversarially Trained Dynamics Model

To visualise the influence of training the transition dynamics model in an adversarial manner as proposed by RAMBO, we consider the following simple example MDP. In the example, the state space (S) and action space (A) are 1-dimensional with, $A = [-1, 1]$. For this example, we assume that the reward function is known and is equal to the current state, i.e. $R(s, a) = s$, meaning that greater values of s have greater expected value. The true transition dynamics to the next state s' depend on the action but are the same regardless of the initial state, s.

In Figure 4 we plot the data present in the offline dataset for this MDP. Note that the actions in the dataset are sampled from a subset of the action space: $a \in [-0.3, 0.3]$. Because greater values of s' correspond to greater expected value, the transition data indicates that the optimal action covered by the dataset is $a = 0.3$.

In Figure 5 we plot the output of running naïve model-based policy optimisation (MBPO) on this illustrative offline RL example. Figure 5a illustrates the MLE transition function used by MBPO. The transition function fits the dataset for $a \in [-0.3, 0.3]$. Outside of the dataset, it predicts that an action of $a \approx 1$ transitions to the best successor state. Figure 5b shows that applying a reinforcement learning algorithm (SAC) to this model results in the value function being over-estimated, and $a \approx 1$ being predicted as the best action. This illustrates that the policy learns to exploit the inaccuracy in the model and choose an out-of-distribution action.

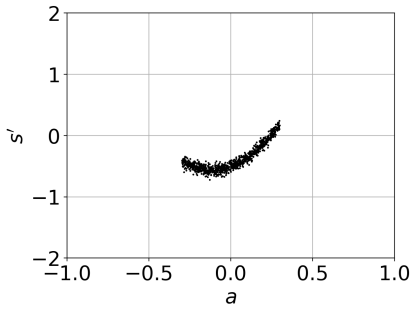
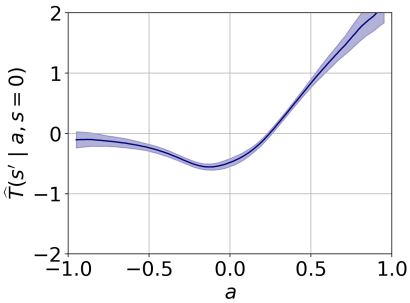
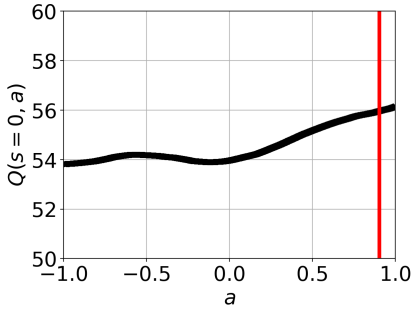


Figure 4: Transition data for illustrative example. The transition function is the same regardless of the initial state. The actions in the dataset are sampled from $a \in [-0.3, 0.3]$.

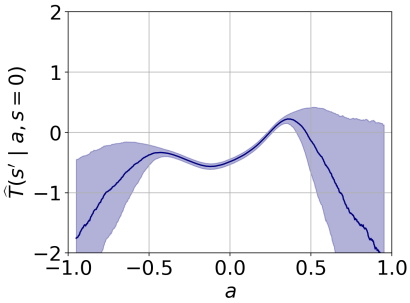


(a) Maximum likelihood estimate of the transition function, which is used by MBPO. Shaded area indicates ± 1 SD of samples generated by the ensemble.

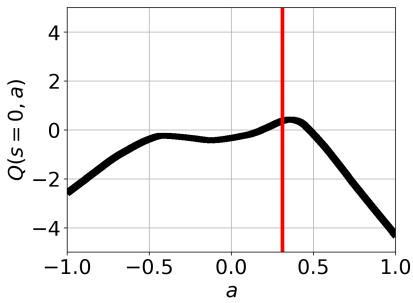


(b) Q-values after running SAC for 15 iterations. The values are significantly over-estimated. The red line indicates the mean action taken by the SAC policy.

Figure 5: Plots generated by running naïve MBPO on the illustrative MDP example.



(a) Model which is adversarially trained using RAMBO. Shaded area indicates ± 1 SD of samples generated by the ensemble. The transition function accurately predicts the dataset for in-distribution actions, but predicts transitions to low value states for actions which are out of distribution.



(b) Q-values from SAC critic. The red line indicates the mean action taken by the policy trained using SAC, which is $a \approx 0.3$ (the best in-distribution action). Training on pessimistic synthetic transitions regularises the Q-values for out-of-distribution actions.

Figure 6: Output generated by running RAMBO for 15 iterations on the illustrative MDP example, using an adversary weighting of λ =3e-2.

In Figure 6 we show plots generated after running RAMBO on this example for 15 iterations, using an adversary weighting of $\lambda = 3\text{e-}2$. Figure 6a shows that the transition function still accurately predicts the transitions within the dataset. This is because choosing $\lambda \ll 1$ means that the MLE loss dominates for state-action pairs within the dataset. Due to the adversarial training, for actions $a \notin [-0.3, 0.3]$ which are out of the dataset distribution, the transition function generates pessimistic transitions to low values of s, which have low expected value.

As a result, low values are predicted for out-of-distribution actions, and the SAC agent illustrated in 6b learns to take action $a \approx 0.3$, which is the best action which is within the distribution of the dataset. This demonstrates that by generating pessimistic synthetic transitions for out-of-distribution actions, RAMBO enforces conservatism and prevents the learnt policy from taking state-action pairs which have not been observed in the dataset.

C.4 Ablation of Behaviour Cloning Initialisation

For the MuJoCo locomotion experiments we initialised the policy using behaviour cloning, as noted in Section 6 and Appendix B.4. This is a common initialisation step used in previous works [25, 74]. In Table 7 we present ablation results where the behaviour cloning initialisation is removed, and the policy is initialised randomly. Table 7 indicates that the behaviour cloning initialisation results in a small improvement in the performance of RAMBO.

Table 7: Ablation of RAMBO with no behaviour cloning initialisation on the D4RL benchmark [14]. We report the normalised performance during the last 10 iterations of training averaged over 5 seeds.

		RAMBO	**RAMBO, No BC**
Random	HalfCheetah	40.0 ± 2.3	39.5 ± 3.5
	Hopper	21.6 ± 8.0	25.4 ± 7.5
	Walker2D	11.5 ± 10.5	0.0 ± 0.3
Medium	HalfCheetah	77.6 ± 1.5	77.9 ± 4.0
	Hopper	92.8 ± 6.0	87.0 ± 15.4
	Walker2D	86.9 ± 2.7	84.9 ± 2.6
Medium Replay	HalfCheetah	68.9 ± 2.3	68.7 ± 5.3
	Hopper	96.6 ± 7.0	99.5 ± 4.8
	Walker2D	85.0 ± 15.0	89.2 ± 6.7
Medium Expert	HalfCheetah	93.7 ± 10.5	95.4 ± 5.4
	Hopper	83.3 ± 9.1	88.2 ± 20.5
	Walker2D	68.3 ± 20.6	56.7 ± 39.0
MuJoCo-v2 Total:		826.2 ± 33.8	812.4 ± 47.8