Robust Reinforcement Learning in a Sample-Efficient Setting

Anonymous authors
Paper under double-blind review

Abstract

The performance of reinforcement learning (RL) in real-world applications can be hindered by the absence of robustness and safety in the learned policies. More specifically, an RL agent that trains in a certain Markov decision process (MDP) often struggles to perform well in MDPs that slightly deviate. To address this issue, we employ the framework of Robust MDPs (RMDPs) in a model-based setting and introduce a novel learned transition model. Our method specifically incorporates an auxiliary pessimistic model, updated adversarially, to estimate the worst-case MDP within a Kullback-Leibler uncertainty set. In comparison to several existing works, our method does not impose any additional conditions on the training environment, such as the need for a parametric simulator. To test the effectiveness of the proposed pessimistic model in enhancing policy robustness, we integrate it into a practical RL algorithm, called Robust Model-Based Policy Optimization (RMBPO). Our experimental results indicate a notable improvement in policy robustness on high-dimensional MuJoCo control tasks, with the auxiliary model enhancing the performance of the learned policy in distorted MDPs, while maintaining the data-efficiency of the base algorithm. Our methodology is also compared against other robust RL approaches. We further examine how pessimism is achieved by exploring the learned deviation between the proposed auxiliary world model and the nominal model. By introducing a pessimistic world model and demonstrating its role in improving policy robustness, our research presents a general methodology for robust reinforcement learning in a model-based setting.

1 Introduction

Reinforcement learning (RL) has been shown to perform well in many environments. However, the performance of a trained RL agent can rapidly decrease when the agent is evaluated in a slightly altered environment (Christiano et al., 2016; Rusu et al., 2017). This is one of the issues that has limited the adoption of RL in real-world scenarios, more specifically due to the simulation-to-reality (sim2real) gap and inherent variability in real control systems. These control systems could for example be robots, where the friction changes over time due to the oil in the joints. Therefore, there is a need for policies that are robust enough to perform well in environments that differ from the training environment. Due to this necessity, various approaches tackle the sim2real issue, often using different problem formulations (Zhao et al., 2020). Some of these approaches include domain randomization or transfer learning. In our work, however, we aim to maximize the worst-case performance of the RL agent under bounds on the uncertainty, commonly formalized as a robust Markov decision process (RMDP). This formalism defines an *uncertainty set* of multiple Markov decision processes (MDPs), where the agent is oblivious to which MDP of the set it is acting in. The objective in an RMDP then becomes to maximize the return in the worst (i.e., lowest cumulative reward) MDP of the uncertainty set. In previous research, methods that work within the RMDP formalism have demonstrated enhanced robustness against perturbations between the train and test environment (Gadot et al., 2024; Pinto et al., 2017). However, these works often impose extra requirements on the training environment, such as the ability to re-sample a transition multiple times or to have access to a parametric environment during training. A second challenge for RL in some real-world applications is the sample efficiency, since it is often slow to perform exploration (e.g., on a physical robot). Model-based reinforcement learning (MBRL) is an approach

that has demonstrated significant progress in sample efficiency, such as the work of Janner et al. (2019) for simulated robotics or the more general work by Hafner et al. (2023) that allows visual state representations.

This paper adopts the RMDP setting and proposes a novel algorithm that improves the robustness of a learned policy, without placing any additional requirements on the training environment. Importantly, we work within the MBRL framework and aim to maintain the sample efficiency of these methods. Inspired by the ideas of Rigter et al. (2022) and Pinto et al. (2017), our approach introduces an auxiliary model that acts as an adversary to minimize the cumulative reward under the current policy. This auxiliary objective then defines a two-player Markov game with the policy optimization objective. By sequentially optimizing these two competing objectives, our algorithm can optimize towards a more robust policy. Our **main contributions** are firstly *(i)*, proposing a novel robust MBRL algorithm to improve robustness in an online setting, while remaining sample efficient. This is achieved by adding an auxiliary model to model-based policy optimization (MBPO) which learns a pessimistic world model via adversarial updates. Secondly *(ii)*, we evaluate the empirical performance of our algorithm on high-dimensional Multiple Joint Control (MuJoCo) benchmarks under both single and double parameter distortions [1] [2]. Thirdly *(iii)*, we interpret and quantify how the predictions of the learned robust model differ from the nominal model, demonstrating how the agent achieves robustness. The remainder of this work will first describe current robust reinforcement learning research. Then, we will highlight relevant background to our approach. Furthermore, the methodology is described in detail. Subsequently, the results demonstrate the improvement in robustness that our method provides to MBPO (Janner et al., 2019) in multiple MuJoCo (Todorov et al., 2012) control environments. Finally, we draw conclusions and outline future research directions.

2 Background

In this section, we first introduce MBRL within the broader context of MDPs. Secondly, RMDPs are described and an adversarial framework to tackle them is highlighted. Finally, the Kullback-Leibler (KL) uncertainty set is defined.

2.1 Model-Based Reinforcement Learning

MBRL (Moerland et al., 2023) operates within the framework of an MDP, defined by the tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma, \rho_0)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $P(s'|s,a)$ is the distribution that defines the probability of ending up in next state s' when taking action a in state s. Next, $r(s,a,s')$ defines the reward function, γ is the discount factor, and $\rho_0(s)$ is the initial state distribution. The objective in RL is to identify an optimal policy π^* that maximizes the expected sum of discounted rewards:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi,P,\rho_0}\left[\sum_{t=0}^{H} \gamma^t r(s_t, a_t, s_{t+1}) \mid s_0 \sim \rho_0\right] \tag{1}$$

In addition, we denote the state visitation distribution of the MDP as d_P^π, which defines the likelihood of being in a state certain state when following policy π. In MBRL, the agent learns a model of the environment's dynamics, represented by $p_\theta(s'|s,a)$, from the data collected through its interactions with the MDP. This model is then used to simulate future states and rewards, reducing the number of interactions with the real environment. The expected reward function, $r(s,a)$, is also learned from data. In most MBRL algorithms, the agent's policy is updated based on both real experiences and simulated experiences from the learned model, balancing between exploration for model learning and exploitation of the learned model for policy improvement. For notational simplicity, we will use s, a and s' to denote s_t, a_t, s_{t+1} respectively, when it is clear from context.

[1] Evaluation code and weights available at `https://github.com/rmbpo-eval/rmbpo-tmlr`
[2] Recorded examples available at `https://sites.google.com/view/rmbpo`

2.2 Robust Markov Decision Processes

In a traditional MDP, the agent optimizes its policy in a static transition model P. However, in some real-world problems, the transition model can change over time. Hence, we can define a Robust MDP (Wiesemann et al., 2013) where the agent acts in an unknown MDP $P \in \mathcal{P}$ that is a sample from an uncertainty set $\mathcal{P}$. The robust objective $J_{\mathcal{P},\pi}$ can now be defined to maximize an objective function in the worst-case MDP of a given uncertainty set. This objective is formally stated in Eq. 2.

$$J_{\mathcal{P},\pi} = \max_{\pi \in \Pi} \min_{P \in \mathcal{P}} \mathbb{E}_{P,\pi} \left[\sum_{t=0}^{H} \gamma^t r(s_t, a_t, s_{t+1}) \mid s_0 \sim \rho_0 \right] \tag{2}$$

The optimal robust policy ($\pi^*_{\mathcal{P}}$) now becomes the policy that maximizes $J_{\mathcal{P},\pi}$ (over the set of achievable policies Π), this is called the outer-loop problem. Additionally, the algorithm is dependent on knowing the worst-case MDP at every time step, we call this the inner-loop problem. For a small uncertainty set, the inner-loop problem can be solved by just evaluating a certain transition in each MDP $P \in \mathcal{P}$. However, when the uncertainty set becomes very large or continuous, the inner-loop problem can be challenging. We will follow related works by considering this combined optimization objective as a two-player zero-sum Markov game (Rigter et al., 2022; Pinto et al., 2017). In this game, one player optimizes the policy, to maximize the return, whilst the other player tries to find $P^* \in \mathcal{P}$, which minimizes the return. Both these players are updated in an alternating manner.

2.3 KL Uncertainty set

Since the "true" uncertainty set is often not known or ill-defined, a common choice is the KL uncertainty set, denoted as $\mathcal{P}_{KL}$ (Hu & Hong, 2013; Gadot et al., 2024; Shi & Chi, 2024). The KL uncertainty set is defined as:

$$\mathcal{P}_{KL} = \left\{ P \in \Delta_{\mathcal{S}} \mid D_{KL}(P(.|s,a)||\bar{P}(.|s,a)) \leq \epsilon_{s,a} \right\}, \tag{3}$$

where $\bar{P}$ is the nominal kernel, i.e. the environment with which the agent interacts during training. $\mathcal{P}_{feasible}$ denotes the set containing all MDPs under consideration, in the case of a parametric model, $\Delta_{\mathcal{S}}$ denotes the probability simplex over $\mathcal{S}$. Furthermore, $D_{KL}(P||\bar{P})$ is the KL divergence between the model P and the nominal model $\bar{P}$, and ϵ is a predefined threshold. We assume $\epsilon_{s,a} = \epsilon$ for all s, a. In this definition, the KL uncertainty set $\mathcal{P}_{KL}$ consists of all models of which every transition is within a KL divergence of ϵ from the nominal model $\bar{P}$. A limitation of this uncertainty set is the dependence on a stochastic transition model, since it would require the (ill-defined) KL-divergence between two Dirac functions in the deterministic setting. However, this limitation can be circumvented without loss of generality by adding action noise between the agent and the MDP (Gadot et al., 2024; Zhou et al., 2024).

3 Auxiliary Model Learning

The goal of this section is to tackle the inner-loop problem of the robust objective, as defined by the minimization problem in Eq. 2, i.e. approximating the worst-case MDP, denoted as $P^* \in \mathcal{P}$, where we choose $\mathcal{P}$ to be the KL uncertainty set centered around the nominal model $\bar{P}$. This choice of uncertainty set follows a common choice in literature (Gadot et al., 2024; Hu & Hong, 2013). To describe our methodology, Section 3.1 introduces the auxiliary adversarial model as an addition to traditional world model learning (e.g. via maximum likelihood estimation (Janner et al., 2019)). The auxiliary model has a well-defined KL divergence with the approximated nominal model. Secondly (Section 3.2), we introduce the loss function to train the auxiliary model to maintain a low KL divergence with the nominal transition model, whilst also learning to be pessimistic (i.e., minimizing the return of the transition).

3.1 Auxiliary Model

Since we work within the context of MBRL, we have direct access to a parameterized approximation, $p_\theta(s', r|s, a)$, of the nominal transition model $\bar{P}(s', r|s, a)$. However, this does not directly provide us with a method to approximate $D_{KL}(p_\theta||\bar{P})$, since we do not have access to the transition probabilities of the training environment $\bar{P}(.)$, needed to construct the KL uncertainty set. Hence, we propose to not directly try to approximate the pessimistic transition model, thus leaving p_θ untouched. As an alternative, we propose an auxiliary parameterized model, g_ψ, which takes as input the outputs of the learned transition model p_θ, in addition to s and a. Next states and rewards can now be sampled according to Eq. 4.

$$s', r \sim g_\psi(\cdot \mid s, a, p_\theta(s', r|s, a)) \tag{4}$$

Since both p_θ and g_ψ define probability distributions, it is possible to compute $D_{KL}(g_\psi||p_\theta)$, which we will consider as an approximation for $D_{KL}(g_\psi||\bar{P})$. In our work, both p_θ and g_ψ define the mean and covariance matrix of a diagonal multivariate Gaussian distribution, so the KL divergence can be computed closed-form. In practice, we provide the predicted mean μ_θ and covariance matrix Σ_θ as inputs to the auxiliary model g_ψ, since a Gaussian is fully defined by these two components. Strictly speaking, the addition of p_θ as an input to the auxiliary model is not necessary, however, this greatly eases the optimization of g_ψ, which will be explained in Section 3.2.

3.2 Training the Auxiliary Model

The goal of the auxiliary model is to minimize the value of each transition under the current policy while remaining within the desired uncertainty set $\mathcal{P}_{KL}$. To incentivize both of these objectives, the auxiliary model is trained by minimizing a sum of two loss functions, displayed in Eq. 5. The first term of the loss function minimizes the return of the current state transition, as proposed by Rigter et al. (2022); by decreasing the (log) likelihood of transitions with a high return and increasing the likelihood of transitions with a low return. The second term is the KL divergence between the approximated nominal model p_θ and the auxiliary model g_ψ, which ensures that the auxiliary model does not deviate too far from the (approximated) dynamics of the nominal environment. Therefore, this part of the loss function incentivizes the model to remain within a KL uncertainty set $\mathcal{P}_{KL}$. The auxiliary model will hence become a pessimistic version of p_θ and $\bar{P}$.

$$J_g(\psi) = \mathbb{E}_{(s',r)\sim g_\psi, s\sim d^\pi_{\psi,\theta}, a\sim\pi}\left[\eta \cdot \ln(g_\psi(s', r|s, a, p_\theta(\cdot|\cdot)) \cdot \mathbf{sg}(r + \gamma V^{\theta,\phi}_\psi(s')) + D_{\mathrm{KL}}(g_\psi||p_\theta)\right] \tag{5}$$

$$\nabla_\psi J_g(\psi) = \mathbb{E}_{(s',r)\sim g_\psi, s\sim d^\pi_{\psi,\theta}, a\sim\pi}\left[\eta \cdot (r + \gamma V^{\theta,\phi}_\psi(s')) \cdot \nabla_\psi \ln(g_\psi(s', r|s, a, p_\theta(\cdot|\cdot)) + \nabla_\psi D_{\mathrm{KL}}(g_\psi||p_\theta)\right] \tag{6}$$

In Eq. 5, **sg()** defines the *stop_grad* operator, no gradient is computed for the value, only for g_ψ (Rigter et al., 2022). The gradient is shown in full in Eqn. 6. $V(.)$ denotes the approximate value function, used to estimate the expected return after the transition. The hyperparameter η controls the influence of the value function: for a small η, the auxiliary model will be almost identical to the nominal model and therefore $\mathcal{P}_{KL}$ is small, for larger values of η, g_ψ will grow more pessimistic and therefore $\mathcal{P}_{KL}$ can be large. Values of η that were used in this work can be found in Appendix C. Formal guarantees that the auxiliary model remains in the uncertainty set are left for future work.

3.3 A Supervised Toy Experiment

Before moving to a RL algorithm in the next section, we set up a supervised toy problem, this will allow us to choose some hand-crafted value functions and interpret their effect on the pessimistic model learning visually. We create a dataset with samples of a standard normal distribution. This dataset represents samples of the transition model, given a single state and action. As a next step we learn nominal parameters $\theta = \{\mu_{nominal}, \sigma_{nominal}\}$ that define the approximated nominal distribution. In the final step, we follow the

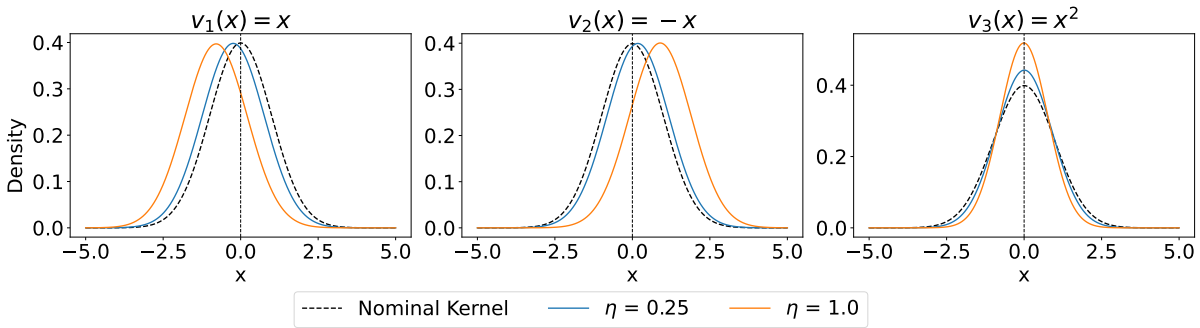


Figure 1: A toy experiment where we learn a pessimistic auxiliary model. The nominal model is a standard Gaussian, the associated value function is highlighted on each plot.

methodology of Sec. 3.2 to learn the parameters $\psi = \{\mu_{pessimistic}, \sigma_{pessimistic}\}$, which define the pessimistic auxiliary distribution. Note that it is not strictly necessary to approximate θ, and we could just provide the ground truth nominal model to compute the KL-divergence. However, we wanted to remain as close as possible to the setting of Section 4.

The following three value functions are used: $v_1(x) = x$, $v_2(x) = -x$ and $v_3(x) = x^2$. For v_1, we expect the pessimistic model to biased towards lower values of x, since there is a linear correlation between x and the value of x. For v_2 we make an analogous reasoning for a bias towards higher values of x. Lastly, we expect an unbiased, distribution for v_3, however the standard deviation is expected to be smaller. This follows from the fact that the normal distribution is already centered around the point where the value function is minimized, i.e. $x = 0$. The results of these experiments are shown in Fig. 1, which confirms that the auxiliary model is biased towards low-value points, and that this bias scales with η. We also performed experiments when learning a categorical distribution instead of a Gaussian, these can be found in Appendix B, together with a summary of the supervised algorithm that was used.

4 Robust Policy Learning

We propose robust model-based policy optimization (RMBPO), a RL algorithm that incorporates the auxiliary model to improve the robustness of the learned policy. Furthermore, we discuss the implications of RMBPO on the performance bound of MBPO, where we motivate the choice for the KL uncertainty set.

4.1 Proposed Reinforcement Learning Algorithm

To improve policy robustness, we combine the auxiliary model with MBPO (Janner et al., 2019) to create RMBPO. MBPO approximates the training environment by maximizing the log likelihood of experienced transitions under its learned model p_θ. This model is a neural network that predicts a mean and covariance matrix over the next states and rewards, conditioned on the current state and action. On-policy rollouts are then performed on the learned model. Finally, the unrolled data is used to update a policy via Soft Actor-Critic (SAC) (Haarnoja et al., 2018). We modify MBPO by training an auxiliary model in addition to the existing model, via Eq. 5. Since these two models are trained separately, the auxiliary model learning does not hinder the accuracy or precision of p_θ. During the model unroll, we pass the current state through the learned model p_θ, after which we use the output of that model $(\mu_\theta, \Sigma_\theta)$ as input to the auxiliary model. The auxiliary model will then predict a modified (μ_ψ, Σ_ψ) as an approximation to the worst-case transition model in $\mathcal{P}_{KL}$. Relating to Section 2.2, the auxiliary model tries to solve the inner-loop problem, while SAC tries to maximize the outer-loop problem. These two components act as two players in a zero-sum Markov game (see Eqn. 2). This procedure is fully described in Algorithm 1, where our additions are highlighted in blue. Following other works (Gadot et al., 2024; Zhou et al., 2024), we add a small amount of action noise to

the environment, otherwise, the uncertainty set would not be well-defined. More details on the action noise are provided in Appendix D.

Algorithm 1 RMBPO (Additions in blue)

Initialize policy π_ϕ, predictive model p_θ , auxiliary model g_ψ,
environment dataset $\mathcal{D}_{env}$, model dataset $\mathcal{D}_{model}$
for N epochs **do**
  while improving **do**
    Update model parameters θ on environment data: $\theta \leftarrow \theta - \lambda_p \hat{\nabla}_\theta J_p(\theta, \mathcal{D}_{env})$
  end while
  while improving **do**
    Update model parameters ψ according to Eq. 5: $\psi \leftarrow \psi - \lambda_a \hat{\nabla}_\psi J_g(\psi, \mathcal{D}_{env}, p_\theta, \pi_\phi)$
  end while
  for E steps **do**
    Take action in environment according to π_ϕ; add to $\mathcal{D}_{env}$
    for M model rollouts **do**
      Sample s_t uniformly from $\mathcal{D}_{env}$
      On-policy rollout according to Eq. 4 starting from s_t using policy π_ϕ; add to $\mathcal{D}_{model}$
    end for
    Perform (soft) actor-critic updates on ϕ using samples from D_{model}. x
  end for
end for

4.2 Implications on the performance bound

Letting the auxiliary model minimize the value of transitions, is a direct application of the definition of an RMDP. However, we also want to limit the difference in returns in the nominal MDP. This section will motivate that bounding the KL divergence is a way to limit this loss in episode returns. By starting from the theoretical insights, provided by Janner et al. (2019), we can see that a certain policy has a bounded difference between the returns under the real model (environment) and the learned model (p_θ). If the expected total variation (TV) distance between two transition distributions is bounded at each time step by ϵ_m and the policy divergence be bounded by ϵ_π, then the difference between the true returns and model returns is bounded, this bound is restated by Eqn. 7.

$$G[\pi] \geq \hat{G}[\pi] - \left[\frac{2\gamma r_{max}(\epsilon_m + 2\epsilon_\pi)}{(1-\gamma)^2} + \frac{4 r_{max}\epsilon_\pi}{(1-\gamma)}\right] \tag{7}$$

We can employ this insight to bound the difference in returns of the optimal policy on the learned model and the auxiliary model, as the learned model serves as the data-generating "environment" for the auxiliary model. I.e., we are interested in lower bounding the return under p_θ, given the return under g_ψ. This would mean that improving the policy under the auxiliary model also improves the policy under the nominal learned model, which provides a lower bound on the performance in the real training environment. Therefore, we define $\epsilon_{m^{aux}} = \max_t E_{s\sim\pi,t}[D_{TV}(g_\psi(s', r|s, a)||p_\theta(s', r|s, a))]$ as the expected maximum TV distance between the auxiliary model and the learned model. Furthermore, because the two models are unrolled under the same policy, we know that $\epsilon_{\pi^{aux}} = 0$. Employing Eqn. 7 in this setting provides Eqn. 8.

$$\hat{G}[\pi] \geq \hat{G}_{aux}[\pi] - \frac{2\gamma r_{max}(\epsilon_{m^{aux}})}{(1-\gamma)^2} \tag{8}$$

We can combine Eqn. 7 and Eqn. 8 to become:

$$G[\pi] \geq \hat{G}_{aux}[\pi] - \left[\frac{2\gamma r_{max}(\epsilon_m + 2\epsilon_\pi + \epsilon_{m^{aux}})}{(1-\gamma)^2} + \frac{4 r_{max}\epsilon_\pi}{(1-\gamma)}\right] \tag{9}$$

This makes intuitive sense, if g_ψ is very different from p_θ, our agent will perform poorly in the training environment. If g_ψ is (almost) identical to p_θ, the RL agent learns from a model that is identical to the nominal MBPO model, and the performance bound becomes identical. Since $\epsilon_{m^{aux}}$ denotes the TV distance, and we know from Pinkster's inequality that the KL-divergence bounds the TV distance, we know that minimizing the KL divergence will lower-bound the performance in the nominal environment (Pinsker, 1964). With a very small η, the auxiliary model will focus on minimizing the KL divergence, and hence the TV distance. The more η is increased, the less the loss function will focus on the KL divergence compared to value minimization, hence becoming more pessimistic, but losing performance in the nominal environment (and probably everywhere). This trade-off between adversarial robustness and optimality is well studied in literature. Empirical results on the relation between η and the KL divergence can be found in Appendix A.2.

5 Main Results

The following section aims to answer three main research questions: *(i) "Can the auxiliary model make a learned policy more robust?"*, *(ii) "How does RMBPO compare against other robust RL approaches?"* and *(iii) "How does the auxiliary model learn pessimistic state transitions?"*. The first two questions are investigated in Section 5.1 and Section 5.2, where we investigate the effect of the auxiliary model, after which RMBPO is compared against SAC and robust natural actor-critic (RNAC) (Zhou et al., 2024). The final question is investigated in Section 5.3, where we perform a limited case study on the Hopper-v4 environment to examine which changes are made by the auxiliary model. For all our results, each algorithm is trained five times using different initial seeds. In accordance with Agarwal et al. (2021), we employ bootstrapped 95% confidence intervals as our metric of confidence. However, in contrast to reporting the interquartile mean (IQM), we report the average performance. The outlier rejection associated with IQM can yield overly optimistic results, which could make it a flawed metric when evaluating robustness.

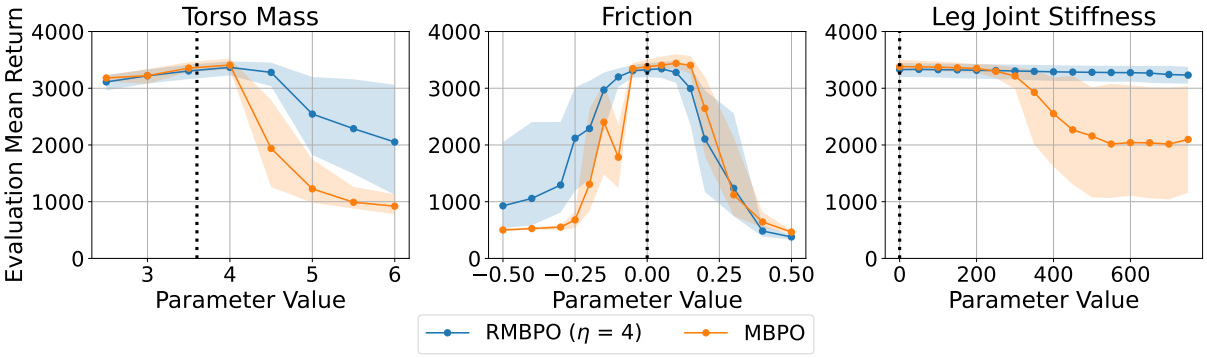
5.1 Effect of the auxiliary model

We evaluate the hypothesis that our proposed auxiliary model aids MBRL algorithms in being more robust. The results under a single distortion are presented in Fig. 2, which compares the trained agents in Hopper-v4, Walker2d-v4 and HalfCheetah-v4. The plots represent a sweep over distortions of a single simulation parameter. Following Pinto et al. (2017), the torso mass and friction are distorted in all environments. Additionaly, we follow Zhou et al. (2024) by distorting the leg joint stiffness in Hopper-v4 and the foot joint stiffness in Walker2d-v4. The plots indicate that our method can improve the robustness of MBPO, since the robustness of RMBPO is either better or matches that of MBPO everywhere.

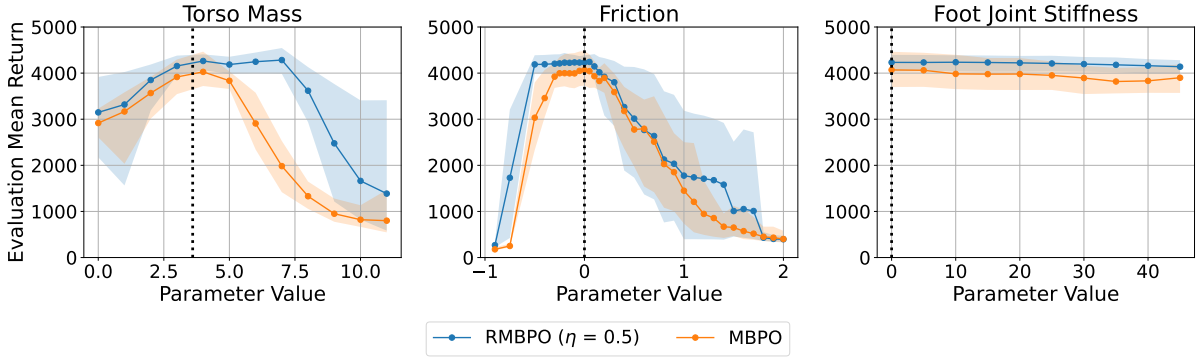
Additionally, we evaluate RMBPO under two simultaneous distortions. We again follow Pinto et al. (2017) and perform a sensitivity analysis on the combination of torso mass and friction distortions. The results are displayed in Fig. 4. In all three tested environments, it is clear that RMBPO is more robust than MBPO, confirming that the auxiliary model also aids the robustness in this setting. Using the data displayed in Fig. 4, we make a cumulative proportion plot in Fig. 5. This figure demonstrates a significant reduction in the number of distortion combinations that deliver a (very) low return. The improvement in robustness can be related to a decrease in optimality in the nominal environment, as can be seen in our experiments in HalfCheetah-v4. The trade-off between nominal optimality and robustness is controlled by the hyper-parameter η. This relates to the theory in Section 4.2 and is a well-known trade-off that is affirmed by previous work (Lee et al., 2024; Gadot et al., 2024). Additional results on the magnitude of η can be found in Appendix A.1.

5.2 Comparing with robust RL approaches

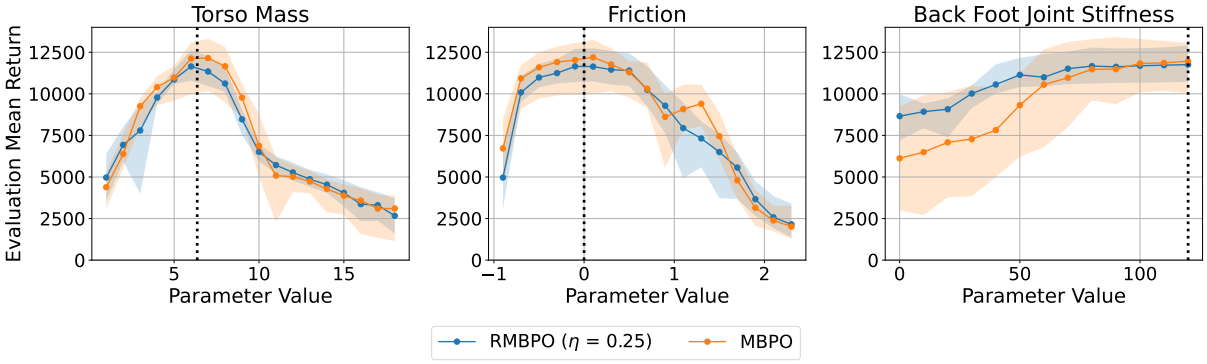
In this section, the robustness of RMBPO will be compared with two other algorithms. Importantly, we should note that SAC uses $1M$ environment samples and RNAC uses $3M$ samples in all experiments, as described in their original papers. RMBPO only uses $125k$ samples for Hopper-v4, $300k$ for Walker2d-v4 and $400k$ for HalfCheetah-v4, leaving the data-efficient setting of MBPO unaltered. For the most direct compar-



(a) Hopper-v4

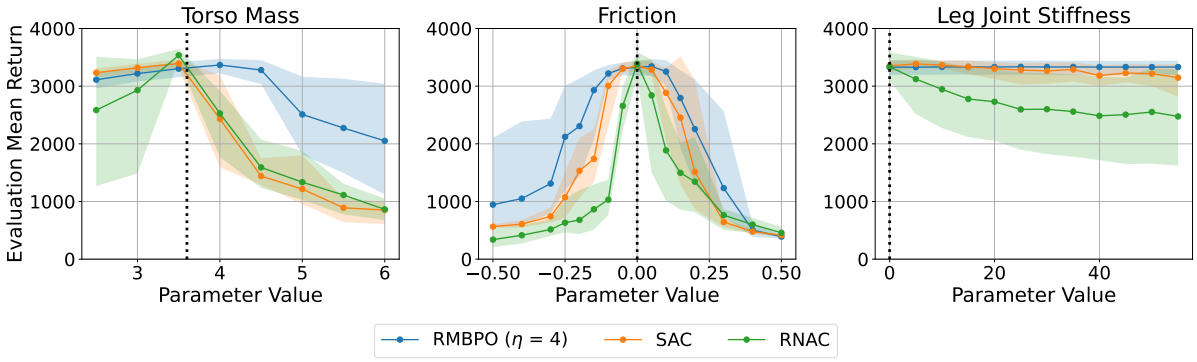


(b) Walker2d-v4

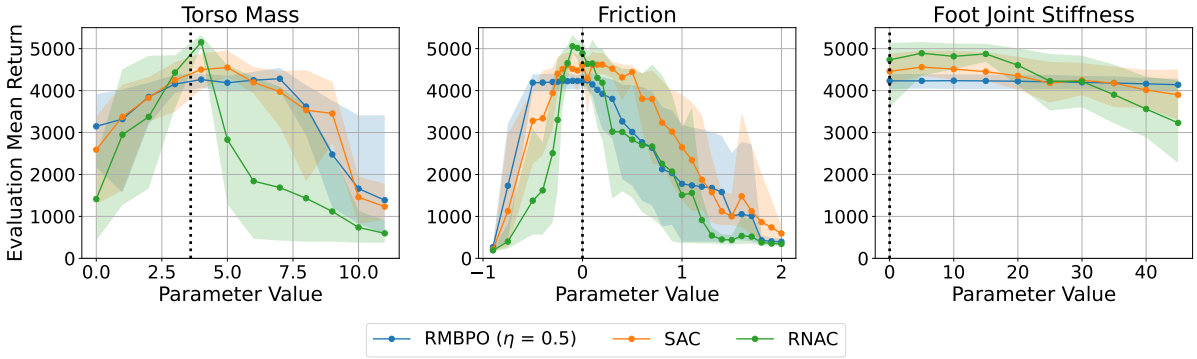


(c) HalfCheetah-v4

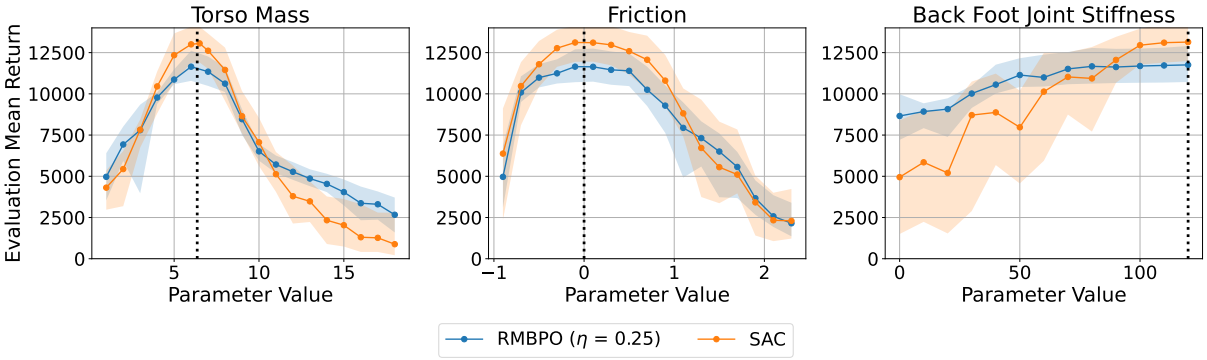
Figure 2: Influence of the auxiliary model on policy robustness under a single distortion. The vertical dotted line indicates the nominal value of the parameter, used during training.



(a) Hopper-v4

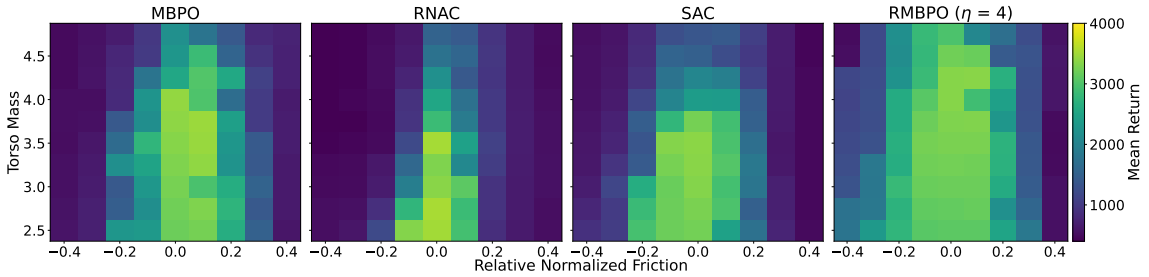


(b) Walker2d-v4

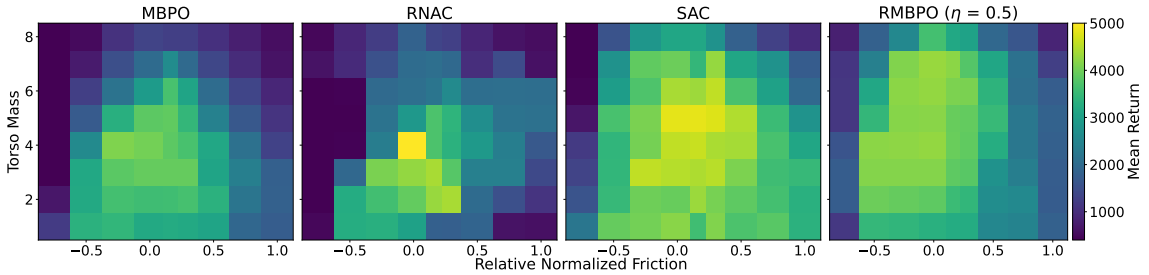


(c) HalfCheetah-v4

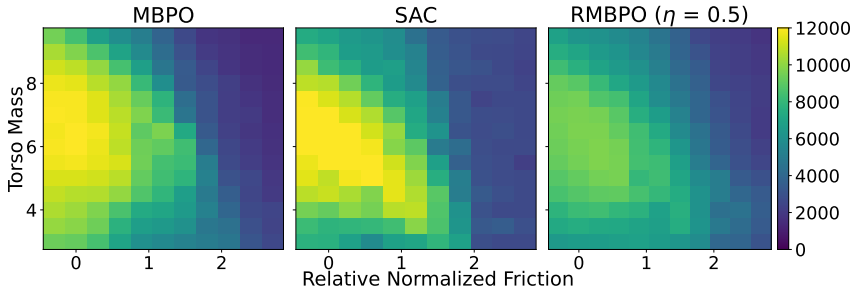
Figure 3: Comparing RNAC, SAC and RMBPO (ours) under a single distortion. The vertical dotted line indicates the nominal value of the parameter, used during training.



(a) Hopper-v4



(b) Walker2d-v4



(c) HalfCheetah-v4

Figure 4: Comparing MBPO, RNAC, SAC and RMBPO (ours) under two distortions.

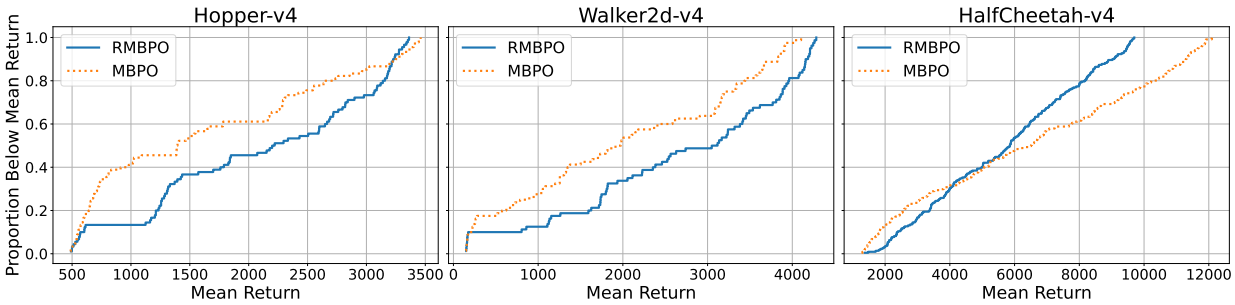


Figure 5: Cumulative proportion of samples below a certain mean return. RMBPO successfully reduces the number of low-return episodes. Samples are combined distortions, identical to Fig. 4.

Table 1: The difference in transition predictions between g_ψ and p_θ. Values indicate angular velocity (ω) or lateral velocity (v). Expressed in rad/s and m/s respectively.

	Torso (ω)	**Thigh Hinge (ω)**	**Leg Hinge (ω)**	**x-Coordinate Torso (v)**
η=0.01	2e-3	-2e-3	-0.0	-2e-5
η=0.2	1.7e-2	-1.3e-2	-4e-3	-5e4
η=0.4	2.9e-2	-2.3e-2	-8e-3	-9.6e-4

ison, we compare against the integral probability metric (IPM) version of RNAC, since this version produces the strongest results. To compare the algorithms, our first experiment evaluates the mean performance in two environments, under a range of distortions. As shown in Fig. 3, RMBPO is the most robust algorithm in Hopper-v4 and HalfCheetah-v4, under all evaluated distortions. Furthermore, RMBPO outperforms RNAC on Walker2d-v4, while achieving similar (arguably slightly worse) results to SAC in this environment.

In a second experiment, we compare RMBPO against the other approaches when dealing with a combination of two distortions. As visible in Fig. 4, the observations remain similar to the single-distortion experiments. RMBPO is again the most robust algorithm on Hopper-v4 and HalfCheetah-v4, while it slightly underperforms to SAC in Walker2d-v4. These results align with the notion that SAC is an adversarially robust algorithm for some problems, but not all (Eysenbach & Levine, 2022; Zhou et al., 2024). Furthermore, in all of our experiments, RMBPO was more robust than RNAC.

5.3 What is the model learning (in Hopper-v4)?

In addition to the quantitative results in this section, we perform a limited case study on how g_ψ modifies the state transitions compared to the approximated nominal model p_θ. In Hopper-v4, the observation space consists of 11 values describing the angles and angular velocities of the joints in the robot and the position and (angular) velocity of the top of the robot. For an exhaustive list, the reader is deferred to Todorov et al. (2012). The goal of the environment is to use three rotors (in the foot, leg, and thigh) to make the robot move forward as fast as possible, without falling. Therefore, we would expect the auxiliary model to modify the transitions in such a way that the robot moves forward more slowly and becomes more prone to falling. To examine the learned model, we display the four largest modifications that are made by the auxiliary model in Table 1. It can be seen that increasing η consistently increases the distance of the robust predictions from the predictions of the nominal model. The four state variables that are the most influenced by the adversarial updates are the angular velocity of the torso, the thigh hinge, the leg hinge and the x-velocity. More importantly, it is shown that the robust model increases the angular velocity of the torso, whilst it decreases the other two angular velocities. This aligns with the intuition of the system, since higher mobility of the torso makes the Hopper harder to control and therefore increases the probability of it falling. The results also demonstrate a lower angular velocity on the actuated parts (such as the leg and thigh). Since these limbs are used to control the robot, this makes the system harder to control. Finally, the lateral velocity of the robot is lowered, which directly reduces the step-wise reward of the environment. All these transition modifications are visually illustrated in Fig. 6.

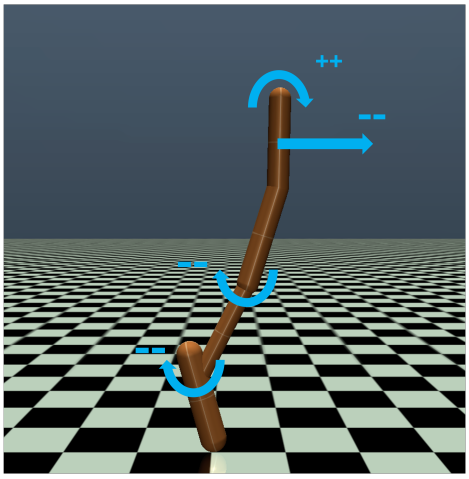


Figure 6: A render of Hopper-v4, annotated with the four largest modifications of the auxiliary model. An increase in (angular) velocity is denoted with '++', a decrease with '- -'.

6 Related Works

Many works focus on robust reinforcement learning in a tabular setting. These works include a robust policy gradient (Wang & Zou, 2022; Kumar et al., 2024) and a tractable approach to tackle non-rectangular RMDPs (Goyal & Grand-Clement, 2023). In a step towards generality, Wang & Zou (2021) and Morimoto & Doya (2005) consider robust reinforcement learning with function approximation on the inverted pendulum problem. Recently, Wang et al. (2024) provide a robust RL algorithm with sample complexity analysis. As many works exist that consider tabular robust RL, the reader is referred to Moos et al. (2022) for more information on the topic.

In the context of high-dimensional state and action spaces, Pinto et al. (2017) propose adversarial RL for robustness. They show that an adversarial approach can make RL robust towards differences between the training and evaluation environment. In contrast to our work, the adversary in their methodology has access to parameters of the simulator during training. Gadot et al. (2024) propose a methodology where multiple next states are sampled at each time step from a stochastic transition model. Subsequently, a single next state is resampled with an importance weight, based on the value of that state. Similar to this work, the KL uncertainty set is considered, however, their methodology requires a simulator where multiple next states can be sampled at any time step. Rajeswaran et al. (2017) investigate an approach that, similar to ours, makes use of MBRL with ensemble world models. However their methodology explicitly requires training randomization over the distortion parameter that is evaluated. Rigter et al. (2022) propose an approach similar to ours, with the goal of being robust to out-of-distribution data in offline RL. More recently, Zhou et al. (2024) provide a model-free alternative to our work. Improved robustness against transition dynamics is demonstrated in the MuJoCo environment, in addition to exhaustive theoretical motivation. We compare against this work in the results section. Recently, Liu et al. (2024) introduce a robust RL algorithm, with theoretical guarantees on the robustness and sample complexity of their approach. However, their work is restricted to an action robust setting. Additionally, the work of Rigter et al. (2024) demonstrates the benefit of adversarial robustness in a reward-free RL setting. Queeney et al. (2024) introduce a novel uncertainty set, called Optimal Transport Perturbations, and demonstrate its effectiveness in improving robustness and safety in a simulated robotics setting. Finally, Queeney & Benosman (2024) consider model-free robust RL to improve the safety of a learned policy.

7 Conclusion and Future Works

This work proposed a novel approach for robust adversarial RL in an online, high-dimensional setting. We have motivated the use of an auxiliary model to tackle the inner-loop optimization problem of the RMDP formulation and provided a version of this auxiliary model, based on the KL uncertainty set. This pessimistic auxiliary model was then implemented in a practical MBRL algorithmm, called RMBPO. Our experiments demonstrate that the auxiliary model improves the robustness of MBRL, while remaining in the same data-efficient setting. Secondly, our method was compared to other recent model-free robust RL approaches. RMBPO matched or outperformed the robustness of these algorithms using significantly less data. Finally, we performed a limited case study which interprets the way in which the auxiliary model helps policy robustness.

As a future work, we want to tackle the setting of very noisy nominal MDPs, such as explored in Gadot et al. (2024). Other interesting areas for future work could include policy mixing between a traditional and a robust policy, to limit the potential downside of not exploiting the environment optimally. Furthermore, it might be interesting to look at a way to formally ensure that the auxiliary model remains within the desired uncertainty set, combined with theoretical guarantees on the robustness of the policy, as we believe that this is a vital step towards RL in industrial applications.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Paul Christiano, Zain Shah, Igor Mordatch, Jonas Schneider, Trevor Blackwell, Joshua Tobin, Pieter Abbeel, and Wojciech Zaremba. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518*, 2016.

Benjamin Eysenbach and Sergey Levine. Maximum entropy RL (provably) solves some robust RL problems. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=PtSAD3caaA2`.

Xu Feng. Unstable baselines. `https://github.com/x35f/unstable_baselines`, 2021.

Uri Gadot, Kaixin Wang, Navdeep Kumar, Kfir Yehuda Levy, and Shie Mannor. Bring your own (Non-robust) algorithm to solve robust MDPs by estimating the worst kernel. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 14408–14432. PMLR, 21–27 Jul 2024. URL `https://proceedings.mlr.press/v235/gadot24a.html`.

Vineet Goyal and Julien Grand-Clement. Robust markov decision processes: Beyond rectangularity. *Mathematics of Operations Research*, 48(1):203–226, 2023.

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

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Zhaolin Hu and L Jeff Hong. Kullback-leibler divergence constrained distributionally robust optimization. *Available at Optimization Online*, 1(2):9, 2013.

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.

Navdeep Kumar, Esther Derman, Matthieu Geist, Kfir Y Levy, and Shie Mannor. Policy gradient for rectangular robust markov decision processes. *Advances in Neural Information Processing Systems*, 36, 2024.

Bruce D Lee, Thomas TCK Zhang, Hamed Hassani, and Nikolai Matni. Performance-robustness tradeoffs in adversarially robust control and estimation. *IEEE Transactions on Automatic Control*, 2024.

Guanlin Liu, Zhihan Zhou, Han Liu, and Lifeng Lai. Efficient action robust reinforcement learning with probabilistic policy execution uncertainty. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL `https://openreview.net/forum?id=9sZsjfZV3q`.

Thomas M Moerland, Joost Broekens, Aske Plaat, Catholijn M Jonker, et al. Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 16(1):1–118, 2023.

Janosch Moos, Kay Hansel, Hany Abdulsamad, Svenja Stark, Debora Clever, and Jan Peters. Robust reinforcement learning: A review of foundations and recent advances. *Machine Learning and Knowledge Extraction*, 4(1):276–315, 2022.

Jun Morimoto and Kenji Doya. Robust reinforcement learning. *Neural computation*, 17(2):335–359, 2005.

Mark S Pinsker. Information and information stability of random variables and processes. *Holden-Day*, 1964.

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

James Queeney and Mouhacine Benosman. Risk-averse model uncertainty for distributionally robust safe reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

James Queeney, Erhan Can Ozcan, Ioannis Paschalidis, and Christos Cassandras. Optimal transport perturbations for safe reinforcement learning with robustness guarantees. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=cgSXpAR4Gl.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. EPOpt: Learning robust neural network policies using model ensembles. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SyWvgP5el.

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

Marc Rigter, Minqi Jiang, and Ingmar Posner. Reward-free curricula for training robust world models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=eCGpNGDeNu.

Andrei A Rusu, Matej Večerík, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. Sim-to-real robot learning from pixels with progressive nets. In *Conference on robot learning*, pp. 262–270. PMLR, 2017.

Laixi Shi and Yuejie Chi. Distributionally robust model-based offline reinforcement learning with near-optimal sample complexity. *Journal of Machine Learning Research*, 25(200):1–91, 2024.

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

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Yudan Wang, Shaofeng Zou, and Yue Wang. Model-free robust reinforcement learning with sample complexity analysis. In *Proceedings of the Fortieth Conference on Uncertainty in Artificial Intelligence*, UAI '24, 2024.

Yue Wang and Shaofeng Zou. Online robust reinforcement learning with model uncertainty. *Advances in Neural Information Processing Systems*, 34:7193–7206, 2021.

Yue Wang and Shaofeng Zou. Policy gradient method for robust reinforcement learning. In *International conference on machine learning*, pp. 23484–23526. PMLR, 2022.

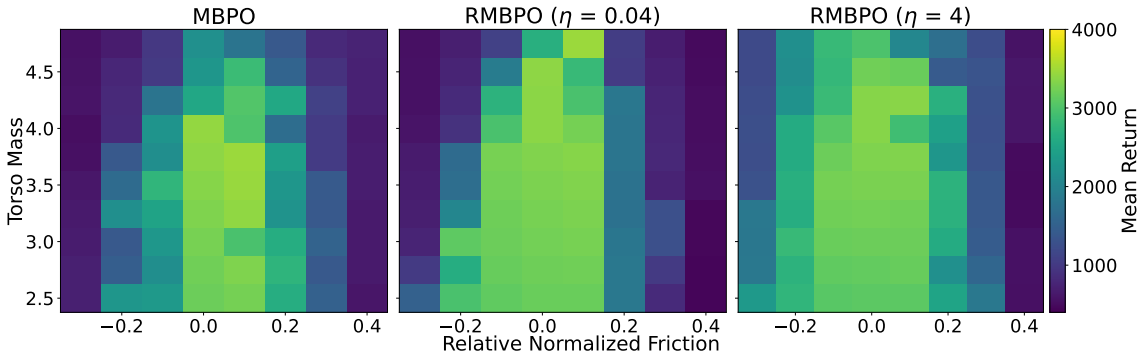
Wolfram Wiesemann, Daniel Kuhn, and Berç Rustem. Robust markov decision processes. *Mathematics of Operations Research*, 38(1):153–183, 2013.

Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE symposium series on computational intelligence (SSCI)*, pp. 737–744. IEEE, 2020.

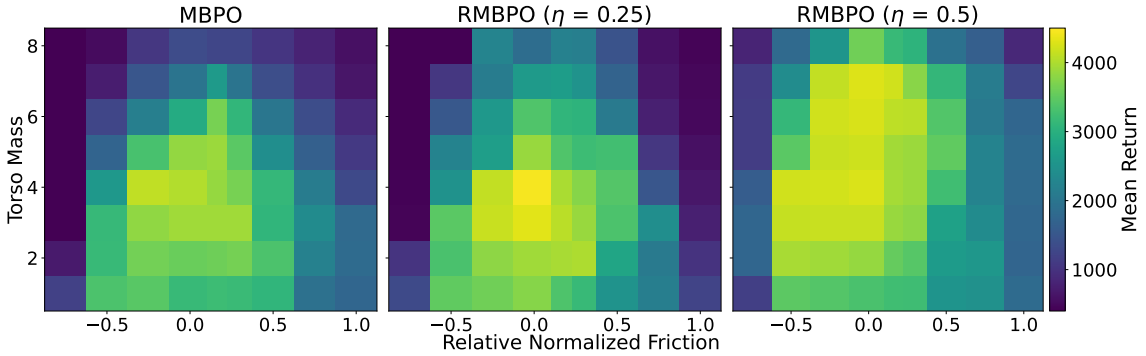
Ruida Zhou, Tao Liu, Min Cheng, Dileep Kalathil, PR Kumar, and Chao Tian. Natural actor-critic for robust reinforcement learning with function approximation. *Advances in neural information processing systems*, 36, 2024.

A Additional Results

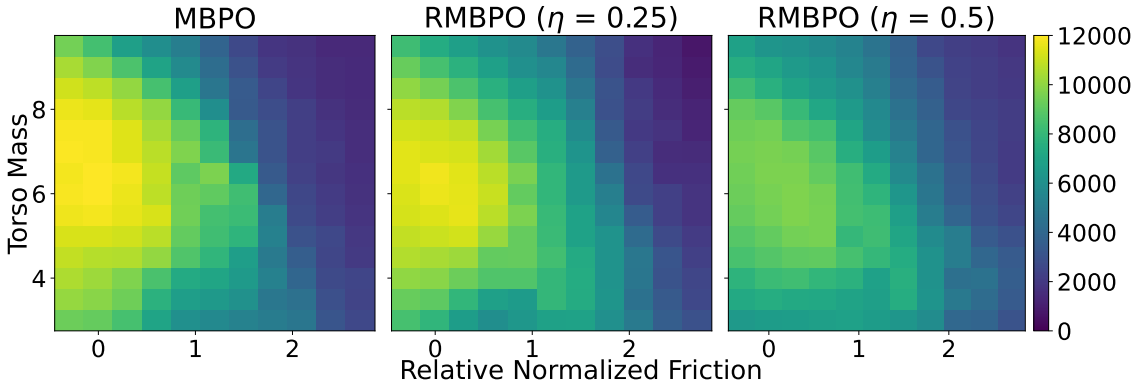
A.1 Effect of η on robustness



(a) Hopper-v4



(b) Walker2d-v4



(c) HalfCheetah-v4

Figure 7: Influence of auxiliary model on policy robustness under two distortions. All experiments demonstrate that larger η increases robustness, possibly at the cost of optimality.

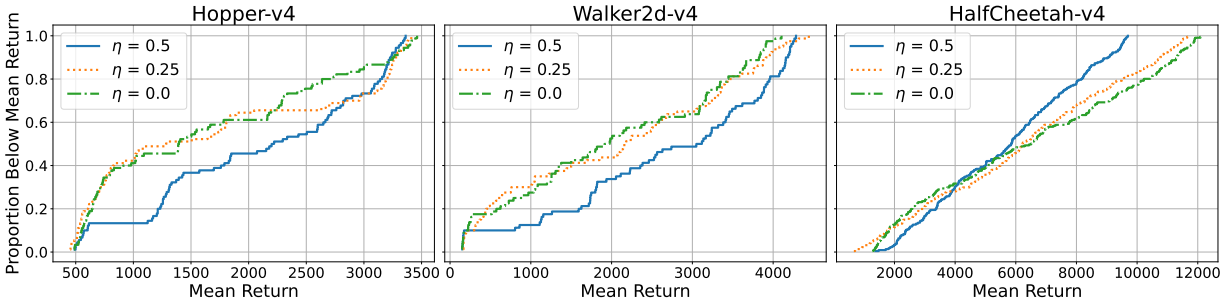
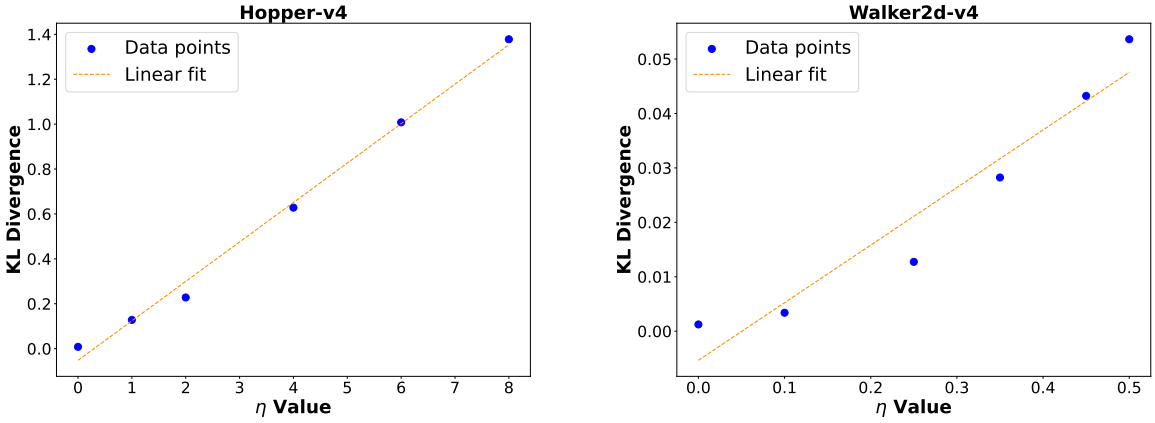


Figure 8: Cumulative proportion of samples below a certain mean return. A higher η value successfully reduces more low-return episodes. Samples are combined distortions, identical to Fig. 7.

A.2 Emperical effect of η on KL divergence

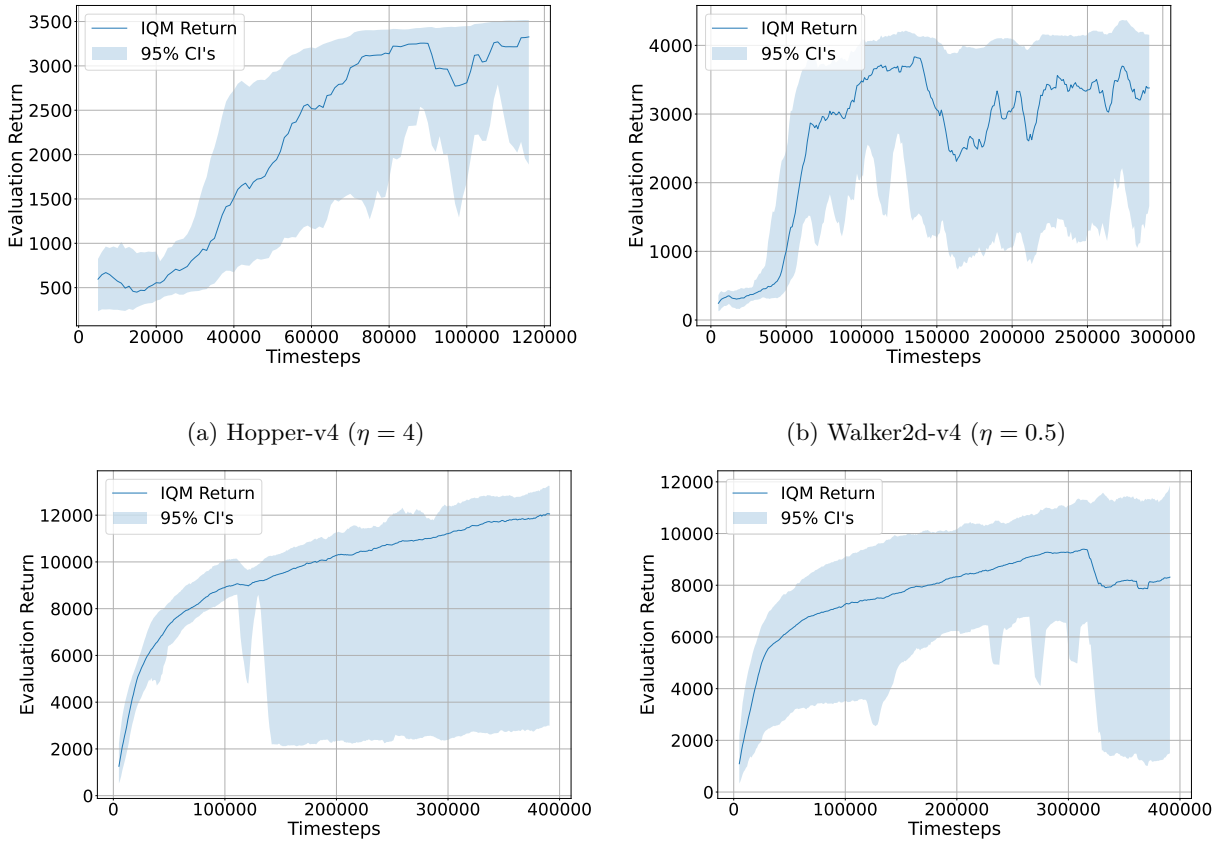


(a) Hopper-v4

(b) Walker2d-v4

Figure 9: The KL divergence between the approximated nominal model and the auxiliary model, in function of η. Linear fit included for visual reference.

A.3 Learning Curves



(a) Hopper-v4 ($\eta = 4$)

(b) Walker2d-v4 ($\eta = 0.5$)

(c) HalfCheetah-v4 ($\eta = 0.25$)

(d) HalfCheetah-v4 ($\eta = 0.5$)

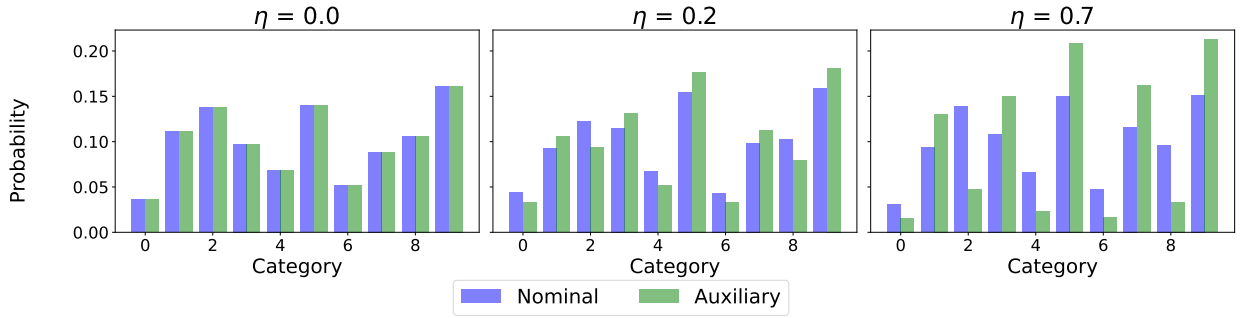
Figure 10: Evaluation curves of RMBPO. IQM with 95% confidence intervals over 5 training runs with different seeds. Timesteps denote the number of interactions with the training environment.

B Toy Experiment Details

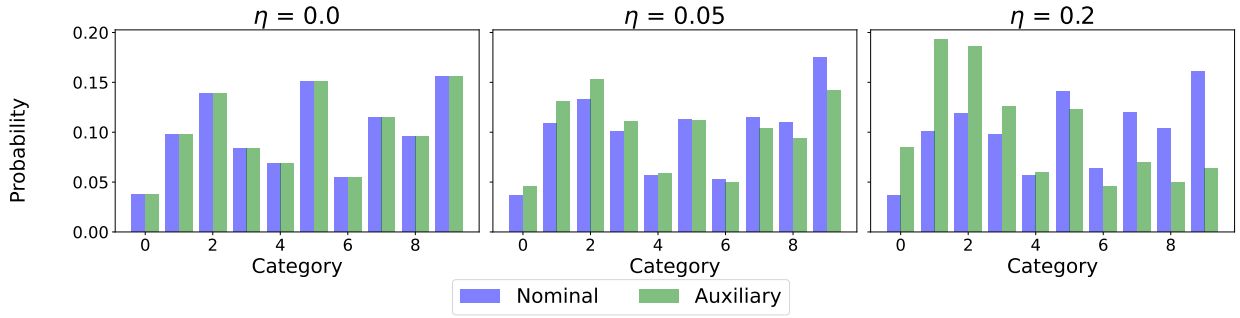
A summary of the algorithm that was used in the toy experiments is provided in Algorithm 2.

Algorithm 2 Supervised Pessimistic Distribution Learning with an Auxiliary Model

Choose an arbitrary value function $v : \mathbb{R}^n \to \mathbb{R}^n$
Initialize dataset $\mathcal{D}$ with samples from the training distribution
Initialize nominal parameters θ
Initialize pessimistic parameters ψ
while improving **do**
    Update model parameters θ on environment data: $\theta \leftarrow \theta - \lambda_p \hat{\nabla}_\theta J_p(\theta, \mathcal{D})$
end while
while improving **do**
    Update model parameters ψ according to Eq. 5: $\psi \leftarrow \psi - \lambda_a \hat{\nabla}_\psi J_g(\psi, \mathcal{D}, p_\theta, v)$
end while



(a) Value function rewards even categories (0, 2, 4, 8).



(b) Value function is equal to the label of the category (1, 2, 3, ...).

Figure 11: Difference between nominal categorical model and pessimistic categorical model.

Table 2: Hyperparameters

Hyperparameter	**Hopper-v4**	**Walker2d-v4**	**HalfCheetah-v4**
$\boldsymbol{\eta}$	4	0.5	0.25 / 0.5
$\boldsymbol{\lambda_a}$	1e-4	1e-4	1e-4
Total environment steps	125k	300k	400k

Furthermore, to demonstrate that the methodology is not exclusive to Gaussians, we also perform experiments on categorical distributions. A dataset was generated by sampling from a categorical distribution with 10 categories, with the following (randomly generated) probabilities: [0.0364, 0.1024, 0.1335, 0.1107, 0.0668, 0.1367, 0.0558, 0.1067, 0.0981, 0.1529]. Identical to the Gaussian experiments, we follow Algorithm 2 to learn a nominal and an auxiliary model from this data. Instead of parametrizing a mean and variance, we now parametrize the 10 logits. We perform experiments with two value functions. The first value function provides a value of 1 for even categories (0, 2, ...) and -1 for odd categories, the results are shown in Fig. 11a. The second value function just returns the number of the category (e.g. category 4 has a value of 4), these results are shown in Fig. 11b.

C Hyperparamters

We "tune" η by performing a sweep and taking the largest value for which we still find an adequate nominal performance. Note that this hyperparameter is actually more of a design choice, since it trades optimality for robustness. We hypothesize that the optimal value of η is related to the cardinality of the state space of the MDP, however, we leave further investigation for future work. The pessimistic model learning rate

(λ_a) is set to $\frac{1}{10}$ of the normal MBPO model learning rate, this significantly reduces variance on the return during training. Note that we use the same amount of environment steps as MBPO in all environments.

All other hyperparameters remain identical to MBPO (Janner et al., 2019), the auxiliary model g_ψ also has the same architecture as a single model of the the MBPO ensemble world model.

D Implementation details and reproducibility

Following related work (Zhou et al., 2024), we add uniform noise to the action: $a_t \leftarrow a_t + \mathcal{U}(5e-3)$. Since this action noise is invisible to the agent, it introduces stochasticity in the MDP. Inspired by the existing MBPO world model, we standardize the outputs of p_θ before providing them as inputs to g_ψ, this showed incremental stability improvements in some training runs. As proposed in appendix A.1 of Rigter et al. (2022), we subtract $V_\phi^{\theta,\psi}(s)$ as a baseline from the return in Eq. 5, this does not influence the expectation of the gradient but significantly reduces its variance. Note that MBPO/RMBPO does not employ a value network directly, however, we can approximate this with on-policy samples from the Q-value network.

Our implementation is based upon the Unstable Baselines Python library (Feng, 2021). We preferred this implementation because of its clarity, however, we experimentally verified that Unstable Baselines reached the same performance as the original open-source MBPO code. For calculating the bootstrapped confidence intervals, we used the implementation provided by SciPy (Virtanen et al., 2020). Experiments were run on a Ubuntu20.04 (Docker) machine with a single NVIDIA Quadro RTX4000 GPU, two CPU cores, and 10GB of memory. For the RNAC baseline, we used the original code, provided by the authors. For SAC, we made use of the implementation provided by the Stable Baselines 3 framework (Raffin et al., 2021).

We provide the trained weights of the learned policies as supplementary materials, together with the modified environments and an evaluation script [3]. This allows for a clear comparison with our research. We choose to distort the same model parameters as Pinto et al. (2017) and Zhou et al. (2024) to add perspective to the results and ease future benchmarking in the community, also, this avoids cherry-picking the best conditions for RMBPO. To ease implementation, we also release the source code of the toy experiment. The authors are not able to release source code of RMBPO at the time of submission of this paper, however, the reader is encouraged to contact the first author of this work with any related questions.

[3] `https://github.com/rmbpo-eval/rmbpo-tmlr`