Robust Model-Based Reinforcement Learning with an Adversarial Auxiliary Model

Siemen Herremans
siemen.herremans@uantwerpen.be
IDLab, University of Antwerp - imec

Ali Anwar
ali.anwar@uantwerpen.be
IDLab, University of Antwerp - imec

Siegfried Mercelis
siegfried.mercelis@uantwerpen.be
IDLab, University of Antwerp - imec

Abstract

Reinforcement learning has demonstrated impressive performance in various challenging problems such as robotics, board games, and classical arcade games. However, its 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 nearly identical MDPs. 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 work 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. We further explore the learned deviation between the proposed auxiliary world model and the nominal model, to examine how pessimism is achieved. By learning a pessimistic world model and demonstrating its role in improving policy robustness, our research contributes towards making (model-based) RL more robust.

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. Therefore, there is a need for policies that are robust enough to perform well in environments that slightly 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, the goal is to maximize the worst-case performance of the RL agent, commonly formalized as a *Robust Markov decision processes* (RMDP). This formalism defines an *uncertainty set* of multiple MDPs, where the agent is oblivious to which MDP it is acting in. The objective in an RMDP then becomes to maximize the return in the worst (lowest cumulative reward)

MDP of the uncertainty set. In previous research, methods that work within the RMDP formals 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. This paper follows the RMDP formulation and proposes a novel algorithm that improves the robustness of a learned policy, without placing any additional requirements on the training environment. 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 model's objective then acts in 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 model-based RL algorithm to improve robustness in an online setting. This is achieved by adding an auxiliary model to MBPO which learns a pessimistic world model via adversarial updates. Secondly *(ii)*, we evaluate the empirical performance of our algorithm on high-dimensional Gym MuJoCo control benchmarks [1]. Thirdly *(iii)*, we analyze how the predictions of the learned robust model differ from the nominal model. The remainder of this work will first describe the relevant background to our approach. Then, 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 model-based RL (MBRL) within the broader context of Markov decision processes (MDP). 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

Model-Based Reinforcement Learning (MBRL) (Moerland et al., 2023) operates within the framework of a Markov decision process (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-eval`

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 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 Kullback-Leibler (KL) uncertainty set, denoted as $\mathcal{P}_{KL}$. The KL uncertainty set is defined as:

$$
\mathcal{P}_{KL} = \left\{ P \in \mathcal{P}_{feasible} \mid D_{KL}(P||\bar{P}) \leq \epsilon \right\}, \tag{3}
$$

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, $\mathcal{P}_{feasible}$ contains every MDP that can be represented by that model. $D_{KL}(P||\bar{P})$ is the KL divergence between the model P and the nominal model $\bar{P}$, and ϵ is a predefined threshold. In this definition, the KL uncertainty set $\mathcal{P}_{KL}$ consists of all models that are within a KL divergence of ϵ from the nominal model $\bar{P}$. This set is the uncertainty set that will be approximated in our work.

3 Auxiliary Adversarial Model

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, this section first (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). Finally, we propose **Robust MBPO (RMBPO)**, an algorithm that incorporates the auxiliary model to improve the robustness of the learned policy.

3.1 Auxiliary Model

Since we work within the context of Model-Based RL, we have direct access to a parameterized approximation, $p_\theta(s', r|s, a)$, of the nominal transition model $\bar{P}(.)$. 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 simulated transition probabilities, $\bar{P}(s', r|s, a)$, 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.

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

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 under the current policy while remaining within the desired uncertainty set $\mathcal{P}_{KL}$. To incentivize both these objectives, the auxiliary model is trained by minimizing a sum of two loss functions, displayed in Eq. 5. The first term is the Kullback-Leibler (KL) divergence between the normal 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 the KL uncertainty set $\mathcal{P}_{KL}$. The second term of the loss function minimizes the return of the current state transition, as proposed by Rigter et al. (2022). By lowering the (log) likelihood of transitions with a high return and heightening the likelihood of transitions with a low return, the auxiliary model will become a pessimistic approximation of p_θ and $\bar{P}$.

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

In Eq. 5, $\mathbf{sg()}$ defines the *stop_grad* operator, no gradient is computed for the value, only for g_ψ (Rigter et al., 2022). $V(.)$ 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, $p_\theta \approx g_\psi$ 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 A. Formal guarantees that the auxiliary model remains in the uncertainty set are left for future work.

3.3 Proposed Algorithm

To improve policy robustness, we combine the auxiliary model with MBPO (Janner et al., 2019) to become RMBPO. MBPO approximates the training environment by maximizing the 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. This algorithm 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 B.

Algorithm 1 RMBPO (Additions in blue)

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

---

4 Results

The following section aims to answer two main research questions: *"Does the auxiliary model learn pessimistic state transitions?"* and *"Can the auxiliary model make a learned policy more robust?"*. The first question is investigated in Section 4.2, where we perform a case study on the Hopper-v3 environment to examine which changes are made by the auxiliary model. The second question is investigated in Section 4.1, where we plot the performance of RMBPO compared to MBPO under distorted evaluation environments.

4.1 Main Results

The main goal of this section is to evaluate the hypothesis that our proposed auxiliary model aids MBRL algorithms in being more robust. To achieve our results, we train both MBPO and RMBPO in the nominal environment using 3 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 means (IQM), we report the average performance. The outlier rejection associated with IQM can yield overly optimistic results, which makes it a flawed metric when evaluating robustness. The results are presented in Fig. 1, which compares the trained agents in various environments under distortion. I.e. one of the simulation parameters is modified to different values than in the nominal (training) environment. Following Pinto et al. (2017), the pendulum mass is distorted in InvertedPendulum-v2, while the torso mass and friction coefficient are distorted in Hopper-v3 and Walker2d-v3. The results show that RMBPO performs better than MBPO in most of our experiments. This is the case for both the mean performance and the lower limit of the confidence interval. These results support our claim that the auxiliary model makes the policy more robust. Our results

often lie between that of non-robust RL algorithms and RARL (Pinto et al., 2017). This is an expected result since RMBPO cannot distort the simulator during training, in contrast to RARL. Additionally, in Fig. 2, we examine the robustness of RMBPO, compared to MBPO when the standard deviation of the action noise is modified. A noise scale of 1 indicates an MDP that is identical to the training environment (the nominal model), other noise scales denote multiplication factors of the standard deviation. In our results, RMBPO succeeds in being more robust to modifications of the action noise scale during evaluation.

4.2 What is the model learning?

A natural question that arises is how g_ψ modifies the state transitions compared to the approximated nominal model $p_\theta \approx \bar{P}$. Also, the reader might wonder about the relationship between η and the magnitude of these modifications. Therefore, we perform a case study on the MuJoCo Hopper-v3 environment. The observation space of this MDP 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 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 three largest modifications (on state dimensions) that are made by the auxiliary model in Fig. 3. The modifications for all other state dimensions can be found in Appendix D. It can be seen in Fig. 3 that increasing η consistently increases the distance of the robust predictions from the predictions of the nominal model. The three state variables that are the most influenced by the adversarial updates are the angular velocity of the torso, the thigh hinge, and the leg hinge. More importantly, it is shown that the robust model increases the angular velocity of the torso, whilst it decreases the other two. This is expected since higher mobility of the torso makes the Hopper harder to control and therefore increases the probability of it falling. The results also show that the angular velocity of the actuated parts (such as the leg and thigh) is lowered. Since these limbs are used to control the robot, this makes the system harder to control. Furthermore, Fig. 4a shows that the adversarial model lowers the x-velocity of Hopper-v3, which is associated with a lower value since the robot gets rewarded for moving forward rapidly. These results suggest that the auxiliary model makes the state transitions more pessimistic, however, further research is needed to confirm this claim.

5 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. As many works exist in this field, the reader is referred to Moos et al. (2022) for more information on this 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. (Schulman et al., 2015). In contrast to our work, the adversary in their methodology modifies 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. Rigter et al. (2022) propose an approach similar to ours, but to be 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. Furthermore, Queeney & Benosman (2024) consider model-free robust RL to improve safety.
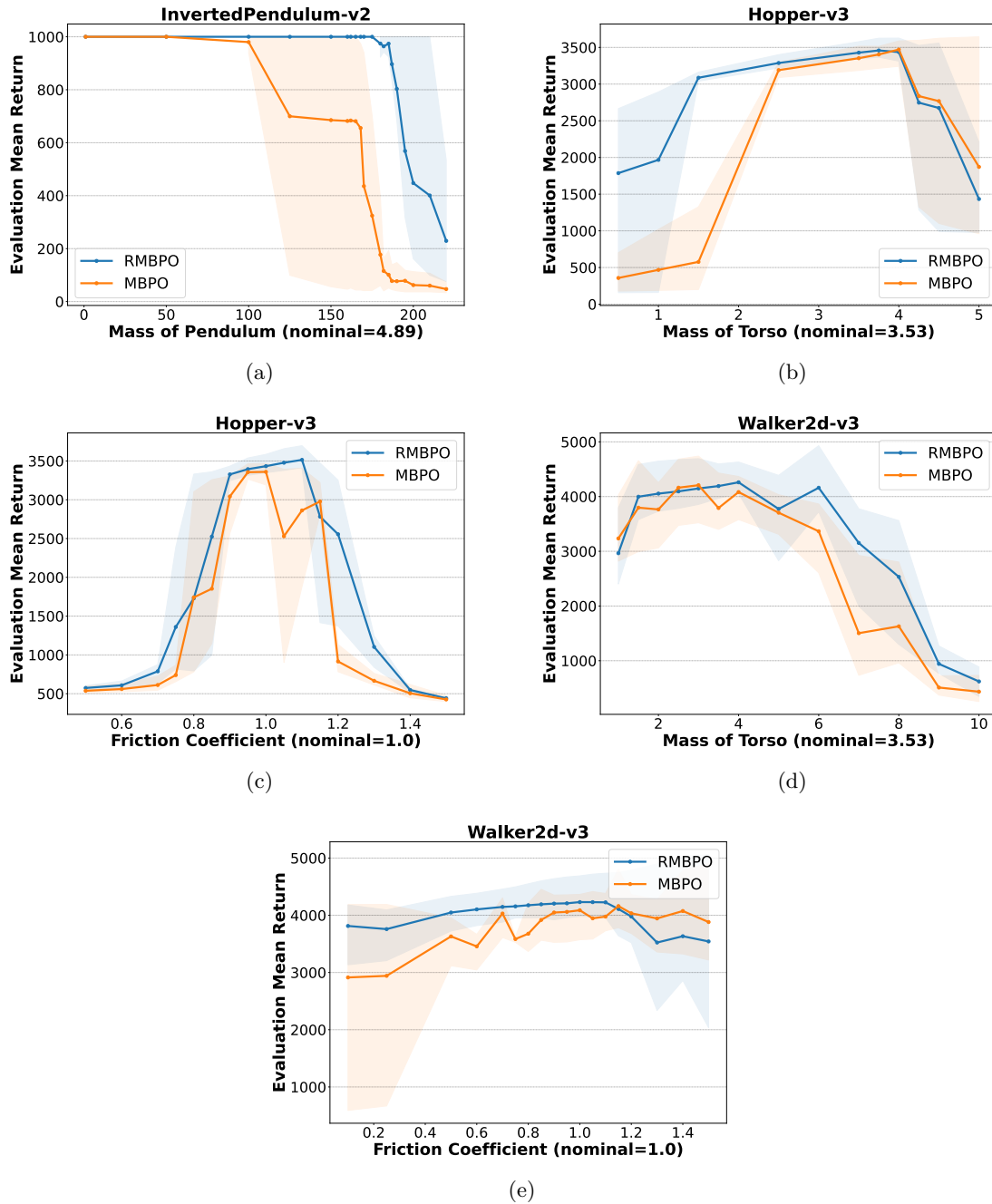
Figure 1: Comparison of robustness between MBPO and RMBPO (ours). Mean return (y-axis) is plotted over the distortion of a simulation parameter during evaluation (x-axis). Results display mean over 3 seeds with the shaded region denoting the bootstrapped 95% confidence interval between seeds.
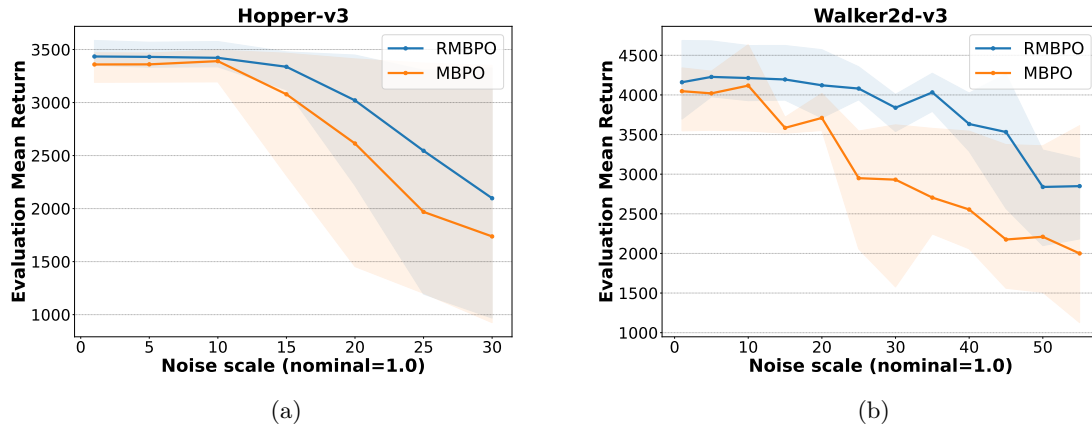
(a)   (b)

Figure 2: Comparison of robustness between MBPO and RMBPO (ours). Mean return (y-axis) is plotted over the distortion of the action noise standard deviation (x-axis). Results display mean over 3 seeds with the shaded region denoting the bootstrapped 95% confidence interval between seeds.
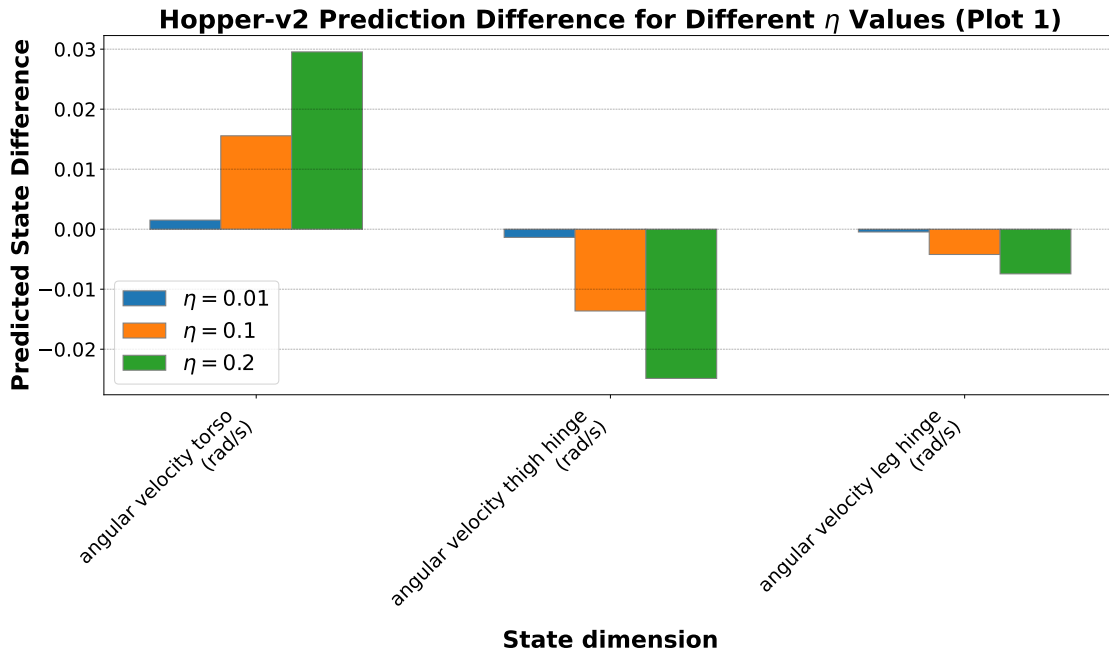

Figure 3: The difference between the nominal model and the auxiliary model. The additional state dimensions can be found in Appendix D.

6   Conclusions and Future work

We have proposed a novel approach for robust adversarial reinforcement learning in an online, high-dimensional setting. We have motivated the use of an auxiliary model to tackle the inner-loop optimization problem of the robust MDP formulation. This auxiliary model was then used in a practical algorithm as a modification to the model-based RL algorithm MBPO. Our experiments demonstrate the potential of the auxiliary model to improve the robustness of MBPO. Although the authors believe that this approach will work for a larger set of algorithms and environments, we leave further experimentation for future work. Other interesting areas for future work could include the addition of a secondary policy that is updated in a non-robust manner. That could ensure that the robust policy does not hinder exploration toward the optimal policy. The authors plan the extension of this work towards latent space models such as PlaNet (Hafner et al., 2019) and its more recent extensions such as DreamerV3 (Hafner et al., 2023).

Acknowledgments

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.

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, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019.

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.

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.

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.

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.

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.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1889–1897, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/schulman15.html.

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.

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   Hyperparamters

Hyperparameter	InvertedPendulum-v2	Hopper-v3	Walker2d-v3
η	0.3	0.004	0.001
λ_a	1e-4	1e-4	1e-4
Total environment steps	15e3	125e3	230e3

Table 1: Hyperparameters

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 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 InvertedPendulum and Hopper, but we use a lower amount in Walker2d (230k compared to 300k), this helped to significantly reduce experiment duration and computational cost. In future work, we aim to evaluate our methodology on more environments for a longer amount of steps.

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

B   Implementation details

Following related work (Zhou et al., 2024), we slightly modify the mentioned MuJoCo environments by adding Gaussian noise to the action: $a_t \leftarrow a_t + \mathcal{N}(0, 5e - 3)$. Only in Pendulum-v2, we use a significantly higher noise variance of 0.2. 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 environments. As proposed in appendix A.1 by 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 does not employ a value network directly, but this does not pose an issue since on-policy samples from the Q-network will provide the same expected gradient, given a large enough minibatch size.

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. All hyperparameters of MBPO remain identical to Janner et al. (2019). 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 v100 GPU, two CPU cores, and 10GB of memory.

C   Reproducibility

To improve reproducibility, we provide all implementation details that are not mentioned in the main body of the paper in Appendix B. Furthermore, we provide the trained weights of the learned policies as supplementary materials, together with the modified environments and an evaluation script [2]. This allows for a clear comparison with our research. We chose to evaluate by distorting

---

[2]https://github.com/rmbpo-eval/rmbpo-eval

the same model parameters as Pinto et al. (2017) for two reasons: a) to add perspective to the results and ease future benchmarking in the community, and b) to avoid cherry-picking the best conditions for RMBPO. To improve the trustworthiness of our results, we share the same 3 seeds throughout all of our experiments, to avoid picking the best seeds per environment. The authors are not able to release source code 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.

D   Additional state difference graphs Hopper-v3
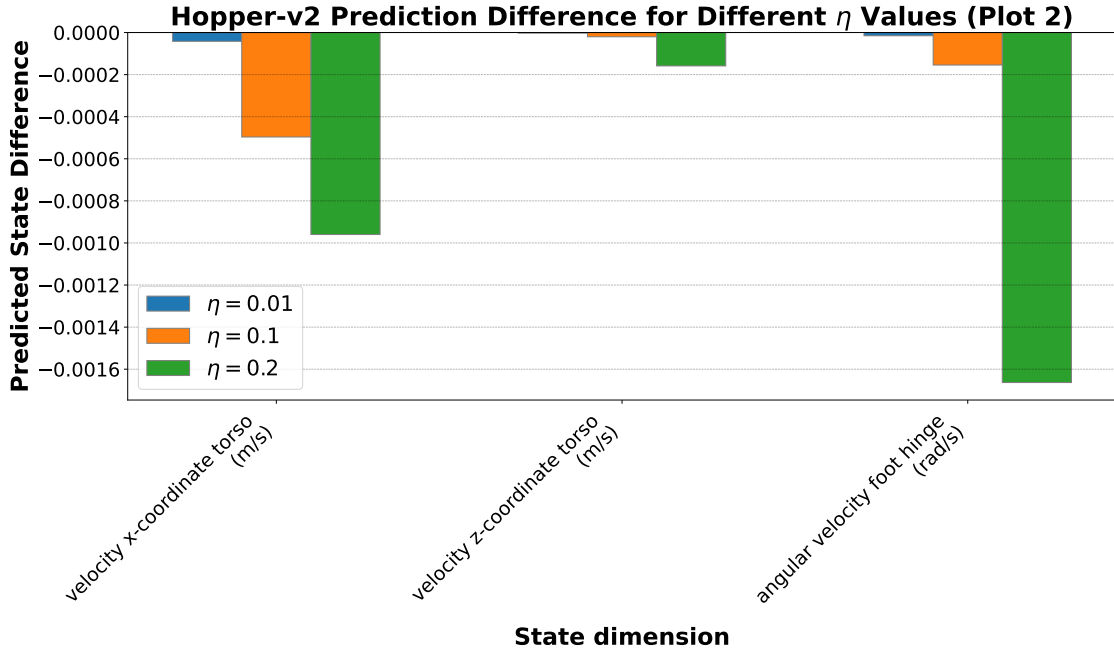
(a) Medium state differences
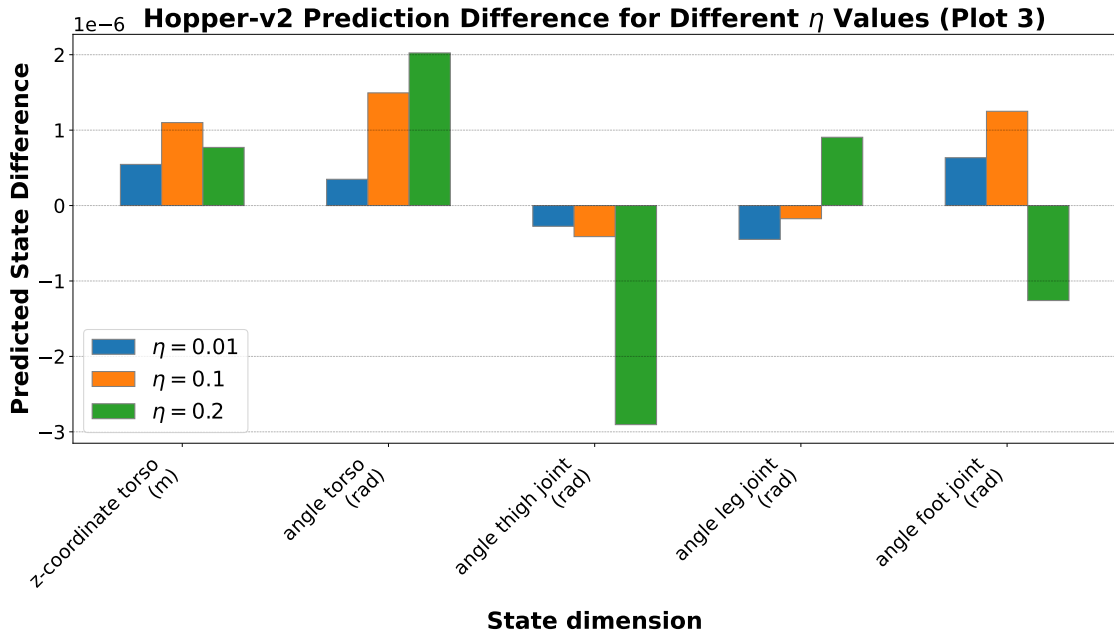
(b) Small state differences

Figure 4: Additional Hopper-v3 state differences.