Trust the Model Where It Trusts Itself - Model-Based Actor-Critic with Uncertainty-Aware Rollout Adaption

Bernd Frauenknecht[*], **Artur Eisele**[*], **Devdutt Subhasish, Friedrich Solowjow, Sebastian Trimpe**
Institute for Data Science in Mechanical Engineering
RWTH Aachen University
52068 Aachen, Germany
bernd.frauenknecht@dsme.rwth-aachen.de

Abstract

Dyna-style model-based reinforcement learning (MBRL) combines model-free agents with predictive transition models through model-based rollouts. This combination raises a critical question: "*When to trust your model?*"; i.e., which rollout length results in the model providing useful data? Janner et al. [2019] address this question by gradually increasing rollout lengths throughout the training. While theoretically tempting, uniform model accuracy is a fallacy that collapses at the latest when extrapolating. Instead, we propose asking the question "*Where to trust your model?*". Using inherent model uncertainty to consider local accuracy, we obtain the Model-Based Actor-Critic with Uncertainty-Aware Rollout Adaption (MACURA) algorithm. We propose an easy-to-tune rollout mechanism and demonstrate substantial improvements in data efficiency and performance compared to state-of-the-art deep MBRL methods on the MuJoCo benchmark.

A similar version of this paper has been presented at the 41^{st} *International Conference on Machine Learning (ICML).*

1 Introduction

Deep reinforcement learning (RL) has shown unprecedented results in challenging domains such as gameplay Mnih et al. [2015], OpenAI et al. [2019b] and nonlinear control OpenAI et al. [2019a], Wurman et al. [2022]. For engineering problems, however, the data inefficiency of model-free state-of-the-art approaches Schulman et al. [2017], Haarnoja et al. [2018a] remains a substantial challenge Kostrikov et al. [2023], Frauenknecht et al. [2023], prompting the need for more efficient methods Janner et al. [2019], Chen et al. [2021]. One such method, model-based reinforcement learning (MBRL), reduces the necessary degree of environment interaction by inferring information from a learned environment model. Unfortunately, model errors can lead to faulty conclusions that severely impact the agent's performance. It is therefore critical to ensure the accuracy of these models.

Model-based policy optimization (MBPO) Janner et al. [2019] represents the current state-of-the-art in Dyna-style MBRL Sutton [1991], combining a Soft Actor-Critic (SAC) agent Haarnoja et al. [2018a,b] with a Probabilistic Ensemble (PE) model Lakshminarayanan et al. [2017]. Janner et al. [2019] address two distinct learning problems: using interaction between the agent and the environment to train the model and, simultaneously, employing model-based rollouts to train the agent. The agent queries the model in short rollouts branched off from states that were observed during environment interaction. The length of these rollouts is gradually increased throughout training, balancing model usage against the risk of model exploitation.

[*]Equal Contribution

17th European Workshop on Reinforcement Learning (EWRL 2024).

Janner et al. [2019] answer the question *"When to trust your model?"* with time-based arguments. Essentially, uniform model accuracy is postulated after sufficiently long training which motivates to use the model for rollouts of predetermined length. However, both the training time and the rollout lengths are notoriously difficult to preschedule. Furthermore, the assumption of uniform model improvement is problematic for complex systems.

Instead, we consider model accuracy as a local property, putting the question *"Where to trust your model?"* at the heart of our approach. The inherent uncertainty of PE models allows for adaptive rollout lengths: wherever the model is uncertain, rollouts are terminated quickly, while longer rollouts can be generated where it is certain.

In this paper, we analyze the learning process of Dyna-style MBRL and present Model-Based Actor-Critic with Uncertainty-Aware Rollout Adaption (MACURA), an algorithm with an easy-to-tune mechanism for model-based rollout length scheduling. In particular, we present the following technical contributions:

- We show monotonic improvement restricting model usage to a subset $\mathcal{E}$ of the state space $\mathcal{S}$;
- We construct the subset $\mathcal{E}$ based on a novel model uncertainty measure; and
- We outperform state-of-the-art Dyna-style MBRL methods with regard to data efficiency and asymptotic performance on the MuJoCo benchmark.

2 Background

In the following, we introduce the fundamental concepts of MBRL, and Dyna-style architectures.

2.1 Reinforcement Learning

We assume the environment to be represented by a discounted Markov decision process (MDP) defined by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, p, \gamma, \rho_0)$, with $\mathcal{S}$ the state and $\mathcal{A}$ the action space, while a dynamics function $p(s' \mid s, a)$ describes transitions between states $s \in \mathcal{S}$ and actions $a \in \mathcal{A}$. Reward $r \in \mathbb{R}$ is generated from a reward function $r(s, a)$ and is discounted by $\gamma \in (0, 1)$. The MDP is initialized from an initial state distribution ρ_0. The RL agent aims to find an optimal policy π^* that maximizes the expected discounted sum of rewards, henceforth referred to as expected return η. Thus,

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \eta[\pi] = \underset{\pi}{\operatorname{argmax}}\, E_\pi \left[\sum_{t=0}^{\infty} \gamma^t r\left(s_t, a_t\right) \right], \tag{1}$$

with $s_0 \sim \rho_0$, $a_t \sim \pi(\cdot \mid s_t)$, and $s_{t+1} \sim p(\cdot \mid s_t, a_t)$. The Q-function represents $\eta[\pi]$ conditioned on specific state-action pairs and is given by $Q^\pi(s_t, a_t) = E_\pi \left[\sum_{k=t}^{\infty} \gamma^{k-t} r\left(s_k, a_k\right) \middle| s_t, a_t \right]$.

2.2 Probabilistic Ensemble Models

In MBRL, we learn a dynamics model $\tilde{p}(s' \mid s, a)$ to approximate the unknown environment dynamics $p(s' \mid s, a)$. In the following, we consider the particularly effective PE approach Lakshminarayanan et al. [2017], Chua et al. [2018] as the model class for dynamics learning. A PE consists of E probabilistic neural networks (PNN) with parameters θ_e, $e \in \{1, \ldots, E\}$, which are trained on bootstrapped datasets via a negative *log*-likelihood loss to approximate the distribution over the next state with a Gaussian distribution

$$\tilde{p}_{\theta_e}(s' \mid s, a) = \mathcal{N}(\mu_{\theta_e}(s, a), \Sigma_{\theta_e}(s, a)). \tag{2}$$

The PE architecture allows a distinction between aleatoric uncertainty due to the process noise of the environment and epistemic uncertainty due to the parametric uncertainty of the model. While aleatoric uncertainty corresponds to high individual variance estimates $\Sigma_{\theta_e}(s, a)$, epistemic uncertainty is measured via model disagreement. Lakshminarayanan et al. [2017] define epistemic uncertainty as the pairwise Kullback-Leibler (KL) divergence D_{KL} between the individual PNN predictions p_{θ_e} and the Gaussian mixture distribution of the ensemble prediction $\tilde{p}_{\mathrm{PE}}$, given by

$$u_{\mathrm{KL}} = \sum_{e=1}^{E} D_{\mathrm{KL}}(\tilde{p}_{\theta_e}(s' \mid s, a) \| \tilde{p}_{\mathrm{PE}}(s' \mid s, a)), \tag{3}$$

with $\tilde{p}_{\mathrm{PE}}(s' \mid s, a) := \frac{1}{E} \sum_{e=1}^{E} \tilde{p}_{\theta_e}(s' \mid s, a)$. In the following, we assume $r(s, a)$ is known.

2.3 Dyna-Style Model-Based Reinforcement Learning

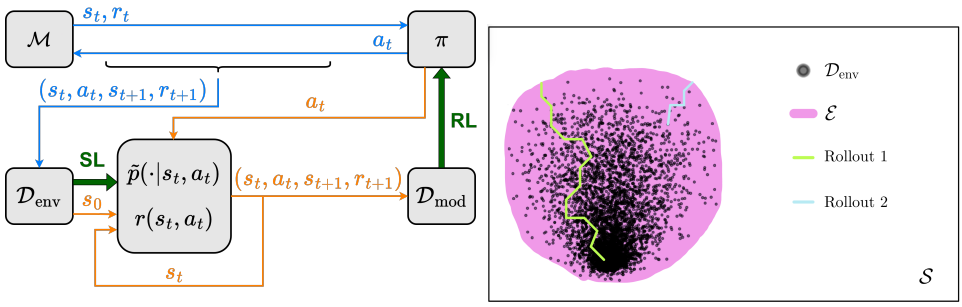
We focus on Dyna-style MBRL Sutton [1991] and MBPO Janner et al. [2019] in particular as it represents the current state-of-the-art approach. A schematic of the architecture is depicted in Figure 1a. In MBPO, a SAC agent Haarnoja et al. [2018a,b] with policy π interacts with the environment. The corresponding agent-environment interaction data are stored in a replay buffer $\mathcal{D}_{\text{env}}$ and are used to train a dynamics model $\tilde{p}$. This model generates experience data in branched model-based rollouts that are stored in a replay buffer $\mathcal{D}_{\text{mod}}$ and used to train the model-free agent. Branched model-based rollouts are thus essential to Dyna-style MBRL as the accuracy of the generated experience data determines the performance of the agent. The mechanism is illustrated in Algorithm 1.

Branched model-based rollouts start at random $s_0 \sim \mathcal{U}(\mathcal{D}_{\text{env}})$, with $\mathcal{U}(\cdot)$ the uniform distribution, and stop at a maximum rollout length $T_{\text{max}} \in \mathbb{N}$. During the rollout, the propagating PNN model within the PE is randomly sampled for each time step. Actions and states are sampled from the respective Gaussians of the policy and the PNN model.

The environment buffer $\mathcal{D}_{\text{env}}$ therefore serves two purposes for model-based branched rollouts. It acts as the training data set for $\tilde{p}$, determining where the model is accurate, and it induces the set of start states for model-based rollouts, influencing the data distribution in $\mathcal{D}_{\text{mod}}$.

A key advantage of branched model-based rollouts is that they reduce the quantity of environment data necessary for learning. This advantage stems from two learning mechanisms. First, the number of update steps per observed transition is limited due to instabilities in value function learning Chen et al. [2021]. The model allows a multitude of transitions to be generated, mitigating this problem. Second, the data distribution in $\mathcal{D}_{\text{mod}}$ differs from that of $\mathcal{D}_{\text{env}}$ and can be more informative to the agent. Generalization capabilities of $\tilde{p}$ allow for a richer set of transitions to be collected. Further, model-based rollouts are conducted under the current policy π of the agent. Therefore, model-based rollouts shift the off-policy distribution in $\mathcal{D}_{\text{env}}$ more towards an on-policy distribution. This generally puts a stronger emphasis on the effects of the current policy and interesting areas of $\mathcal{S}$, accelerating policy improvement. The longer branched model-based rollouts are, the more the data distributions of $\mathcal{D}_{\text{env}}$ and $\mathcal{D}_{\text{mod}}$ may differ. Typically, T_{max} is gradually increased throughout training.

Algorithm 3 in Appendix A provides a detailed description of MBPO. We build on the eminent MBPO method, modifying its core mechanism of branched model-based rollouts. We develop a method to estimate *where* to trust the model and build a new adaptive rollout scheme around it. This scheme may likewise benefit a multitude of derivatives of MBPO such as Zhang et al. [2020], Lai et al. [2020, 2021], Morgan et al. [2021], Fröhlich et al. [2022], Luis et al. [2023b,a].



(a) Dyna-style MBRL. (b) Where to trust your Model?

Figure 1: Algorithmic Diagrams. *(a) An agent with policy π interacts with the environment $\mathcal{M}$. This data is stored in $\mathcal{D}_{\text{env}}$ and used to train a dynamics model $\tilde{p}$ via supervised learning (SL). Model-based rollouts under π are performed from start states s_0 in $\mathcal{D}_{\text{env}}$ and stored in $\mathcal{D}_{\text{mod}}$. The policy is trained on $\mathcal{D}_{\text{mod}}$ via reinforcement learning (RL). (b) $\mathcal{D}_{\text{env}}$ induces a set of sufficient model accuracy $\mathcal{E} \subseteq \mathcal{S}$. A notion of $\mathcal{E}$ allows to reason whether rollouts are in a region of sufficient model accuracy. We use this reasoning to schedule rollout length.*

Algorithm 1 "Vanilla" Branched Model-based Rollouts

Given $\mathcal{D}_{\text{env}}$, $\mathcal{D}_{\text{mod}}$, $\tilde{p}_{\theta_{1,\dots,E}}$, π, and $T_{\max}$
$s_0 \sim \mathcal{U}(\mathcal{D}_{\text{env}})$
for $t = 0, \dots, T_{\max} - 1$ **do**
  $e_t \sim \mathcal{U}(1, \dots, E)$
  $a_t \sim \pi(\cdot \mid s_t)$
  $s_{t+1} \sim \tilde{p}_{\theta_{e_t}}(\cdot \mid s_t, a_t)$
  $r_{t+1} = r(s_t, a_t)$
  $\mathcal{D}_{\text{mod}} \leftarrow \mathcal{D}_{\text{mod}} \cup \{(s_t, a_t, r_{t+1}, s_{t+1})\}$
end for

3 Where to Trust your Model?

Our key idea is to define a subset $\mathcal{E}_{\tilde{p},k} \subseteq \mathcal{S}$ on which the model $\tilde{p}$ is sufficiently accurate at time k. We refer to this subset as $\mathcal{E}$ for brevity and put it at the heart of our approach.

The main technical **problem formulation** then becomes: How can we construct the set $\mathcal{E}$ that ensures a desired accuracy? In particular, we need to:

i) quantify the notion of *sufficiently accurate*;
ii) estimate $\mathcal{E}$ from a given model $\tilde{p}$; and
iii) expand $\mathcal{E}$ quickly via informative data in $\mathcal{D}_{\text{env}}$.

Once we have a suitable $\mathcal{E}$, we obtain a straightforward yet precise mechanism for branched model-based rollouts. As long as the rollout stays within $\mathcal{E}$, the benefits of long rollouts discussed in Section 2.3 outweigh the risk of model exploitation, while the opposite is the case once it leaves $\mathcal{E}$.

Such a distinction is vital as model-based rollouts are performed from random start states of $\mathcal{D}_{\text{env}}$, as depicted in Figure 1b. Depending on the rollout policy, these start states can either lead to rollouts staying in $\mathcal{E}$ for a long time without the necessity to terminate quickly, or leaving it early, where a careful rollout length adaption is crucial. A fixed rollout length based on training time Janner et al. [2019] cannot account for local differences in model accuracy.

4 Monotonic Improvement under Dynamics Misalignment on $\mathcal{E}$

Following the insights set out in Section 3, we provide a formal justification of our approach. We analyze the effect of dynamics mismatch on the expected return, which will be instrumental for formally defining $\mathcal{E} \subseteq \mathcal{S}$ in Section 5 that will be used to schedule rollout length.

4.1 Formulation of Monotonic Improvement

In the context of branched model-based rollouts, we define two MDPs, $\hat{\mathcal{M}}$ and $\tilde{\mathcal{M}}$, on the same $\mathcal{S}$, $\mathcal{A}$, $r(s,a)$, γ and start state distribution ρ_{BR}. Here, $\hat{\mathcal{M}}$ represents the MDP for branched rollouts under the environment dynamics $p(s' \mid s, a)$, whereas $\tilde{\mathcal{M}}$ is the MDP for branched model-based rollouts following model dynamics $\tilde{p}(s' \mid s, a)$. Both, $\hat{\mathcal{M}}$ and $\tilde{\mathcal{M}}$, have identical start states $s_0^{\hat{\mathcal{M}}} = s_0^{\tilde{\mathcal{M}}} \in \mathcal{E}$.

The MDPs are coupled through the following stopping times (which are random variables and thus measurable functions $T : \Omega \to \mathbb{N}$ as indicated by the argument $T(\omega)$):

$$T(\omega) := \min\{T^{\hat{\mathcal{M}}}, T^{\tilde{\mathcal{M}}}\} - 1, \tag{4}$$

$$T^{\hat{\mathcal{M}}}(\omega) = \min\{t \in \mathbb{N} \mid s_t^{\hat{\mathcal{M}}} \in \mathcal{E}^{\text{C}}\}, \tag{5}$$

$$T^{\tilde{\mathcal{M}}}(\omega) = \min\{t \in \mathbb{N} \mid s_t^{\tilde{\mathcal{M}}} \in \mathcal{E}^{\text{C}}\}, \tag{6}$$

with $\mathcal{E}^{\text{C}}$ the complement of $\mathcal{E}$. The stopping times (5) and (6) denote the first time a rollout under $\hat{\mathcal{M}}$ and $\tilde{\mathcal{M}}$ leaves $\mathcal{E}$, respectively. Thus, restricting the rollout length to $T(\omega)$ enforces that branched rollouts remain in $\mathcal{E}$.

We show a monotonic improvement similar to Luo et al. [2018], Janner et al. [2019], Pan et al. [2020]

$$\eta[\pi] \geq \tilde{\eta}[\pi] - C, \tag{7}$$

where $\eta[\pi]$ corresponds to the expected return of the policy π in $\hat{\mathcal{M}}$, while $\tilde{\eta}[\pi]$ denotes the expected return of π under $\tilde{\mathcal{M}}$. As long as the agent improves by more than C in $\tilde{\mathcal{M}}$, we can guarantee improvement in $\hat{\mathcal{M}}$.

Theorem 4.1. *Suppose the expected return following policy π under $\hat{\mathcal{M}}$ is denoted by $\eta[\pi]$ and $\tilde{\eta}[\pi]$ describes the expected return following π under $\tilde{\mathcal{M}}$, then we can define a lower bound for $\eta[\pi]$ on $\mathcal{E} \subseteq \mathcal{S}$ of the form*

$$\eta[\pi] \geq \tilde{\eta}[\pi] - 2r_{\max} \sum_{t=0}^{T(\omega)} \gamma^t \sum_{\tau=0}^{t} \Delta p_{\mathcal{E}}[\pi], \tag{8}$$

with

$$\Delta p_{\mathcal{E}}[\pi] := \sup_{s \in \mathcal{E}, a \sim \pi} D_{\mathrm{TV}}\left(p\left(s' \mid s, a\right) \| \tilde{p}\left(s' \mid s, a\right)\right). \tag{9}$$

Proof. See Appendix B, Theorem B.4. □

We define $C := 2r_{\max} \sum_{t=0}^{T(\omega)} \gamma^t \sum_{\tau=0}^{t} \Delta p_{\mathcal{E}}[\pi]$, which intuitively represents the accumulated worst-case dynamics misalignment. The choice of $\mathcal{E}$ thus influences the bound C since the supremum in (9) is taken over $\mathcal{E}$ and $T(\omega)$ directly depends on $\mathcal{E}$.

4.2 Interpretation of the Result

In contrast to improvement bounds in previous work Luo et al. [2018], Janner et al. [2019], Pan et al. [2020], Theorem 4.1 closely resembles branched model-based rollouts and allows a practical mechanism to be inferred directly.

Rollouts under $\tilde{\mathcal{M}}$ represent the data generated in the practical MBRL algorithm, while $\hat{\mathcal{M}}$ captures the true environment behavior. Both share $\rho_{\mathrm{BR}} = \mathcal{U}(\mathcal{D}_{\mathrm{env}})$, the start state distribution of model-based rollouts introduced in Algorithm 1, where we assume all states in $\mathcal{D}_{\mathrm{env}}$ to be within $\mathcal{E}$ as depicted in Figure 1b. Theorem 4.1 thus bounds the difference in expected return between using the model MDP $\tilde{\mathcal{M}}$ for generating branched model rollouts as compared to performing these rollouts under the environment MDP $\hat{\mathcal{M}}$. Thus, Theorem 4.1 specifically considers model exploitation in branched model-based rollouts that are the predominant data-generating process for training the agent.

By construction, both processes start from identical start states and deviate from each other based on dynamic misalignment upper bounded by $\Delta p_{\mathcal{E}}[\pi]$ until $T(\omega)$ is reached. Unfortunately, the result of Theorem 4.1 is not directly amenable for algorithmic use, as the process $\hat{\mathcal{M}}$ is unknown in practice. Specifically, we are unable to detect when $\hat{\mathcal{M}}$ leaves $\mathcal{E}$ as indicated by $T^{\hat{\mathcal{M}}}(\omega)$ in (5) and need an approximation to obtain a practical algorithm. Assuming trajectories under $\hat{\mathcal{M}}$ and $\tilde{\mathcal{M}}$ stay sufficiently close to each other and replacing $T(\omega)$ with $T^{\tilde{\mathcal{M}}}(\omega) - 1$, we obtain an effective approximation that works well in practice.

5 Constructing $\mathcal{E}$ from Model Uncertainty

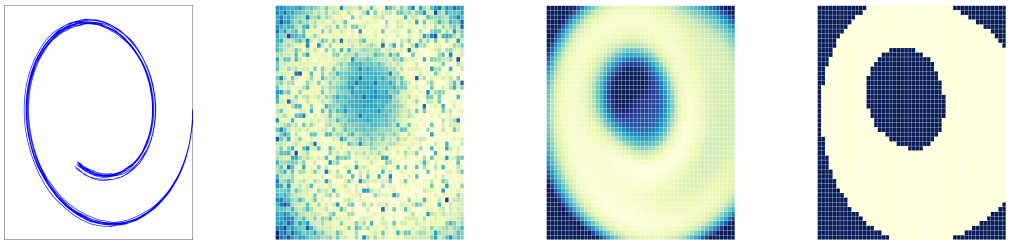
In the following, we define $\mathcal{E}$ such that it represents parts of the state space with high model accuracy.

5.1 Defining $\mathcal{E}$ in Practice

Following Theorem 4.1, ideally we would define

$$\mathcal{E}^* := \{s \in \mathcal{S} \mid D_{\mathrm{TV}}\left(p\left(s' \mid s, a\right) \| \tilde{p}\left(s' \mid s, a\right)\right) \leq \kappa, a \sim \pi(\cdot \mid s)\} \tag{10}$$

such that dynamics misalignment is upper-bounded by a threshold κ. This definition of $\mathcal{E}$ yields $\Delta p_{\mathcal{E}}[\pi] \leq \kappa$ in (9) and thus allows C to be influenced by choosing κ. Estimating the ideal set $\mathcal{E}^*$ is intractable. Instead, we leverage these insights and consider a slightly different reformulation that yields a computationally efficient, numerically well-behaved, and meaningful definition of $\mathcal{E}$.



(a) $\mathcal{D}_{\text{env}}$ (b) $D_{\text{TV}}(p\|\tilde{p})$ (c) u_{GJS} (d) $\mathcal{E}$ with $u_{\text{GJS}} < \kappa$

Figure 2: Constructing $\mathcal{E}$ on a toy example. *(a) Data to train the PE model. (b) Dynamics misalignment. (c) Proposed measure for model uncertainty* (11)*. (d) Set of sufficient model accuracy to perform branched model-based rollouts* (13) .

Assuming that the individual PNNs of the PE model have sufficient representational capacity to model $p(s'|s,a)$ accurately makes epistemic uncertainty u_{PE} an expressive quantity for dynamics misalignment. One option to determine epistemic uncertainty would be using the formulation in (3) and setting $u_{\text{PE}} = u_{\text{KL}}$. If all ensemble members agree sufficiently well on a subset of $\mathcal{S} \times \mathcal{A}$, the PE model approximates the environment dynamics to a sufficient degree of accuracy. Following this reasoning, we construct $\mathcal{E}_{\text{PE}} := \{s \in \mathcal{S} \mid u_{\text{PE}}(s,a) < \kappa, a \sim \pi(\cdot \mid s)\}$ such that some uncertainty u_{PE} under the rollout policy π is upper bounded by a threshold κ.

5.2 Efficient Measure for Model Uncertainty

In an algorithmic implementation, efficient computation of u_{PE} is vital as uncertainty is queried for every model-based transition. The original formulation in (3) has no closed-form solution, rendering it unsuitable for this purpose. Instead, we propose an estimate based on the geometric Jensen-Shannon (GJS) divergence Nielsen [2019]

$$u_{\text{GJS}}(s,a) = \frac{2}{E(E-1)} \sum_{e=1}^{E} \sum_{f=1}^{e-1} D_{\text{GJS}}\left(\mathcal{N}_e \| \mathcal{N}_f\right), \tag{11}$$

with $\mathcal{N}_i =: \mathcal{N}(\mu_{\theta_i}(s,a), \Sigma_{\theta_i}(s,a))$. The Jensen-Shannon divergence is a symmetrized version of the KL divergence but has no closed-form solution for Gaussian distributions. While losing some of the properties of the Jensen-Shannon divergence, the GJS divergence

$$D_{\text{GJS}}\left(\mathcal{N}_e \| \mathcal{N}_f\right) = \frac{1}{2} D_{\text{KL}}\left(\mathcal{N}_e \| \mathcal{N}_{ef}\right) + \frac{1}{2} D_{\text{KL}}\left(\mathcal{N}_f \| \mathcal{N}_{ef}\right) \tag{12}$$

with variance $\Sigma_{ef} = \left(\frac{1}{2}\left(\Sigma_{\theta_e}(s,a)\right)^{-1} + \frac{1}{2}\left(\Sigma_{\theta_f}(s,a)\right)^{-1}\right)^{-1}$ and mean $\mu_{ef} = \Sigma_{ef}\left(\frac{1}{2}\left(\Sigma_{\theta_e}(s,a)\right)^{-1}\mu_{\theta_e}(s,a) + \frac{1}{2}\left(\Sigma_{\theta_f}(s,a)\right)^{-1}\mu_{\theta_f}(s,a)\right)$ recovers a closed-form solution for Gaussian distributions. This allows us to compute model uncertainty in a closed form replacing the comparison to the Gaussian mixture distribution in (3) with a pairwise comparison between PNN predictions. Exploiting the symmetry of the GJS divergence allows to reduce the number of pairwise comparisons.

Thus, u_{GJS} yields an efficient-to-compute uncertainty measure, leading to a practical definition of

$$\mathcal{E} := \{s \in \mathcal{S} \mid u_{\text{GJS}}(s,a) < \kappa, a \sim \pi(\cdot \mid s)\} \tag{13}$$

that is directly applicable to algorithmic use.

5.3 Illustrative Example

As an illustrative example, we use a pendulum with known dynamics and a two-dimensional state. Figure 2 visualizes the construction of $\mathcal{E}$ according to (13) for this system. An in-depth description is provided in Appendix C.

We create a $\mathcal{D}_{\text{env}}$ with a characteristic spiral form by performing rollouts with a feedback controller π_{FL} as depicted in Figure 2a, and use the data to train a PE dynamics model. In the following, dynamic misalignment and uncertainty of this model under π_{FL} are analyzed over $\mathcal{S}$ using heat maps.

Dynamics misalignment as used in (10) is depicted in Figure 2b, where a clear trend can be observed: dynamic misalignment is low, indicated in yellow, close to data in $\mathcal{D}_{\text{env}}$ but grows further away, where blue corresponds to high misalignment. For approximations of the total variation distance we use the common upper bound (24) in Appendix C.4.

Figure 2c shows the proposed model uncertainty u_{GJS} used in (13). We observe a smooth behavior of the uncertainty estimate and rediscover the data distribution in $\mathcal{D}_{\text{env}}$ for the lowest uncertainty values indicated in bright yellow. Most importantly, areas of low uncertainty coincide with areas of low dynamics misalignment in Figure 2b.

As depicted in Figure 2d, choosing a suitable threshold κ allows $\mathcal{E}$ to be defined such that a considerable portion of $\mathcal{S}$ can be explored in model-based rollouts while avoiding areas of high dynamics misalignment. Here $\mathcal{E}$ is visualized in yellow, while blue regions indicate $\mathcal{E}^{\text{C}}$.

6 MACURA: Model-Based Actor-Critic with Uncertainty-Aware Rollout Adaption

Combining the insights into how to choose a set $\mathcal{E}$ to enforce monotonic improvement, discussed in Section 4, with those into how to construct $\mathcal{E}$ using model uncertainty, detailed in Section 5, we present an uncertainty-aware adaption scheme for model-based rollouts. We further discuss expanding $\mathcal{E}$ by efficiently exploring the environment. These building blocks lead to the Model-Based Actor-Critic with Uncertainty-Aware Rollout Adaption (MACURA) algorithm.

6.1 Uncertainty-Based Rollout Adaption

The set $\mathcal{E}$ depends on the uncertainty threshold κ (13). Thus, an appropriate choice of κ is critical in algorithmic design. We propose an adaptive mechanism for determining κ that transfers to different applications and stages of training.

In the algorithm, M branched model-based rollouts are performed in parallel. As these start from states in $\mathcal{D}_{\text{env}}$, we evaluate the model uncertainty after the first prediction step to update κ proportionally to the current model uncertainty.

We therefore define the base uncertainty $\hat{u}_{\text{GJS},k}$ to be the ζ quantile of the M uncertainty measures after the first prediction step at the k^{th} round of model-based rollouts. We use this heuristic as an upper bound on what can be considered certain. Further, we introduce $\xi \in \mathbb{R}^+$ as a tunable scaling factor, which allows $\mathcal{E}$ to be increased or shrunk by scaling $\hat{u}_{\text{GJS}}$ up or down. To stabilize κ over iterations, we define it to be the average scaled base uncertainty:

$$\kappa = \frac{\xi}{K} \sum_{k=1}^{K} \hat{u}_{\text{GJS},k}, \tag{14}$$

with K the rounds of rollouts performed thus far.

Theorem 4.1 indicates that the difference in expected return accumulates over the rollout length, even with bounded dynamics misalignment. Since we only provide a pointwise bound for each transition step, we still need to enforce a maximum rollout length $T_{\max}$, larger than the typical length of adapted rollouts, to avoid extensive error accumulation.

During the experimental evaluation in Section 7, we see that predefined values of $T_{\max} = 10$ and $\zeta = 95\%$ perform well across different applications, requiring no environment-specific tuning. Thus, ξ is a single, interpretable hyperparameter defining the rollout scheme. This makes the proposed uncertainty-aware rollout adaption mechanism formulated in Algorithm 2 considerably easier to tune than those in existing work Janner et al. [2019], Pan et al. [2020].

Due to its variable rollout length, the proposed model-based rollout scheme produces varying amounts of data to train the model-free agent. As the update-to-data ratio plays a crucial role in the stability of model-free RL Chen et al. [2021], we adapt the number of update steps

$$G = \left\lfloor G_{\max} \frac{|\mathcal{D}_{\text{mod}}|}{|\mathcal{D}_{\text{mod}}|_{\max}} \right\rceil \tag{15}$$

Algorithm 2 Uncertainty-Aware Adapted Branched Model-Based Rollouts

Given $\mathcal{D}_{\text{env}}$, $\mathcal{D}_{\text{mod}}$, $\tilde{p}_{\theta_{1,\dots,E}}$, π, ζ, ξ, and $T_{\max}$
$s_0 \sim \mathcal{U}(\mathcal{D}_{\text{env}})$
for $t = 0, \dots, T_{\max} - 1$ **do**
  $e_t \sim \mathcal{U}(1, \dots, E)$
  $a_t \sim \pi(\cdot \mid s_t)$
  $s_{t+1} \sim \tilde{p}_{\theta_{e_t}}(\cdot \mid s_t, a_t)$
  $r_{t+1} = r(s_t, a_t)$
  $u_{\text{GJS}}(s_t, a_t)$ according to (11)
  if $t = 0$ **then**
    update κ according to (14)
  end if
  if $u_{\text{GJS}}(s_t, a_t) < \kappa$ **then**
    $\mathcal{D}_{\text{mod}} \leftarrow \mathcal{D}_{\text{mod}} \cup \{(s_t, a_t, r_{t+1}, s_{t+1})\}$
  else
    break
  end if
end for

to the SAC agent according to the amount of data in the model buffer $|\mathcal{D}_{\text{mod}}|$ compared to its capacity $|\mathcal{D}_{\text{mod}}|_{\max}$. We use this ratio to scale the maximum number of update steps $G_{\max}$ and round the result to the nearest integer. An empirical analysis of how the update-to-data ratio interacts with varying rollout lengths is provided in Appendix D.7.

6.2 Expanding $\mathcal{E}$ through Environment Exploration

Through the notion of $\mathcal{E}$, MACURA has a reliable estimate of where to trust the model $\tilde{p}$ that is trained on $\mathcal{D}_{\text{env}}$. In line with common understanding in system identification Ljung [1998], more informative data yields a better model. Thus, employing effective exploration mechanisms to generate a meaningful $\mathcal{D}_{\text{env}}$ will improve the model and thus expand $\mathcal{E}$, increasing the effectiveness of model-based rollouts.

Although we do not place a strong focus on different exploration mechanisms for Dyna-style MBRL in this work, we tested different approaches. As we discuss in Section 7, we see MACURA perform particularly well with pink exploration noise Eberhard et al. [2023], which introduces a certain degree of temporal correlation between consecutive actions blending white noise and Brownian motion.

We combine the mechanisms above in MACURA, with pseudocode in Algorithm 4 of Appendix A.

7 Experiments and Discussion

Next, we evaluate MACURA on the MuJoCo Todorov et al. [2012] benchmark. A direct comparison to state-of-the-art Dyna-style methods reveals a substantial improvement in data efficiency and asymptotic performance. Further, we provide an ablation over MACURA building blocks.

7.1 Experimental Setup

We compare MACURA to model-based MBPO Janner et al. [2019] and M2AC Pan et al. [2020] approaches as well as the SAC Haarnoja et al. [2018b] algorithm, which represents the model-free learner in all of the methods above. All implementations[2] are based on the recent mbrl-lib library Pineda et al. [2021]. A detailed description of the experimental setup is provided in Appendix D.1.

7.2 Performance Evaluation

The results on the MuJoCo benchmark are depicted in Figure 3. We see that MACURA learns substantially faster than MBPO and M2AC, especially in high-dimensional environments. MACURA

[2]Code available at: `https://github.com/Data-Science-in-Mechanical-Engineering/macura`

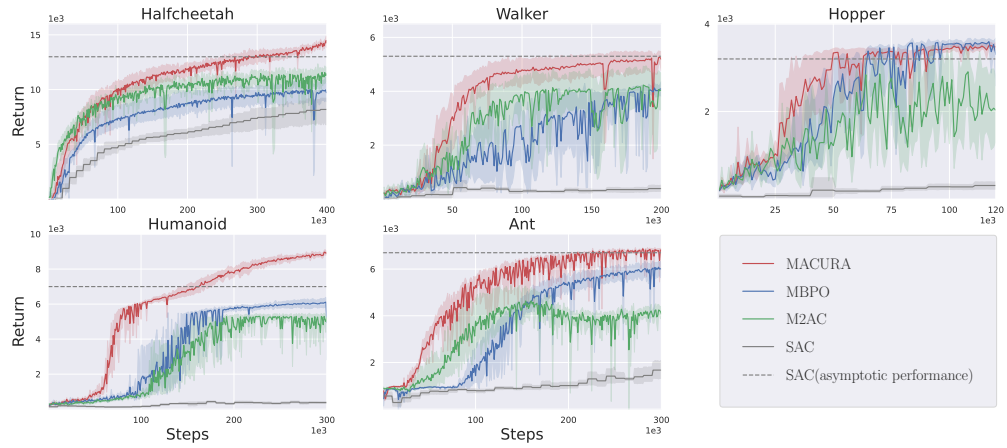
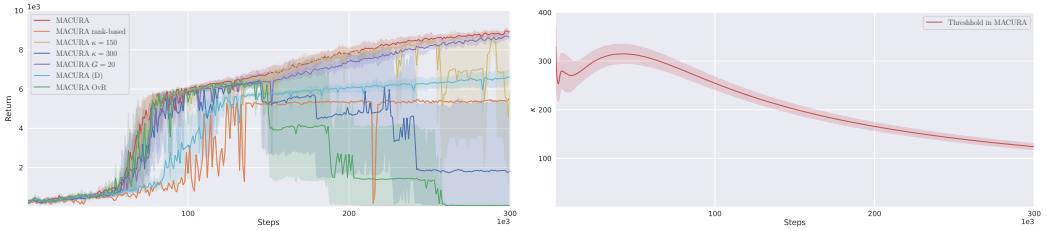


Figure 3: Performance on the MuJoCo Benchmark. *MACURA shows substantial improvements in data efficiency and asymptotic performance over state-of-the-art Dyna-style MBRL approaches (MBPO, M2AC) in most tasks. Most noticeably, MACURA is on par with or outperforms the asymptotic performance of the model-free SAC baseline.*



(a) Ablation on MACURA building blocks.

(b) Uncertainty threshold κ.

Figure 4: Ablation Study *(a) The rank-based rollout mechanism with GJS uncertainty* (11) *and adaptive threshold* (14) *yields strong performance. (b) MACURA initially keeps κ comparatively high and reduces κ over time as more precise information is required to refine the policy.*

further shows considerably stronger asymptotic performance than other model-based approaches . Remarkably, MACURA frequently even outperforms SAC. All model-based methods learn substantially faster than SAC. MBPO with fine-tuned rollout schedules can compete with M2AC.

7.3 Ablation and Further Results

Finally, we provide an ablation study on the building blocks of MACURA on the Humanoid task in Figure 4a. The analysis reveals that the performance gain of MACURA mostly stems from the combination of a threshold-based rollout length adaption mechanism, the self-tuning threshold κ, and the reliable GJS uncertainty estimate.

Replacing the threshold-based rollout adaption of MACURA with the rank-based heuristic of Pan et al. [2020] reduces performance. The threshold-based mechanism, however, requires a reliable uncertainty estimate as in (11) and an adaptive threshold κ as presented in (14) for stable learning.

Replacing the GJS uncertainty estimate with One-vs-Rest (OvR) uncertainty Pan et al. [2020] leads to divergence. We assume this is attributed to the brittleness of the OvR uncertainty estimate as illustrated in Appendix C.4. Thus, faulty predictions are frequently used for training.

Figure 4b depicts the adaption of κ according to (14) for the standard MACURA runs in Figure 4a. Following our intuition, κ is comparatively high in the early stages of training, as uncertain data is sufficient to learn an initial policy. Throughout training, κ decreases as more precise information is required to further improve the policy. Keeping κ fixed at 300 and 150, respectively, destabilizes learning at different stages of training due to model exploitation.

Replacing the gradient step adaption in (15) with a fixed number of gradient steps $G = 20$ yields moderate performance reduction on Humanoid, while deterministic environment interaction hinders an effective expansion of $\mathcal{E}$ and thus substantially reduces performance.

We provide further experimental results in Appendix D. Additional insights on tuning the uncertainty-aware rollout scheme are presented in D.2 and D.3, while the importance of exploration in the environment is discussed in D.4 and D.5. Performance in tasks with process noise is presented inD.6, D.7 provides detailed results for rollout length and update step adaption, D.8 discusses the impact of different uncertainty estimates, and D.9 provides results for long experiments.

8 Related Work

A variety of improvements to the general idea of deep Dyna-style MBRL Janner et al. [2019] have been proposed. These comprise online parameter tuning Lai et al. [2021], reduction of model error during model-based rollouts Fröhlich et al. [2022], Lai et al. [2020], Shen et al. [2020], improved exploration in the model Morgan et al. [2021], Zhang et al. [2020], model learning Ji et al. [2022], Wang et al. [2023], Wu et al. [2022], and consideration of model uncertainty in the model-free Q-function Luis et al. [2023b,a], Wang et al. [2022]. All the above are orthogonal to our method.

Using uncertainty in the model or the RL agent to control model usage is a common technique in MBRL. One approach is to use model data where the agent is uncertain Kalweit and Boedecker [2017], Nguyen et al. [2018]. Model-based offline RL methods Yu et al. [2020], Zhai et al. [2024], Jeong et al. [2023] often construct a pessimistic MDP that penalizes model uncertainty in the value function. Zhang et al. [2021] adapt rollout length in a multi-agent setting based on the error in the policy model of opponent agents. In value expansion methods, rollout steps are frequently reweighted based on model uncertainty Buckman et al. [2018], Jeong et al. [2023], Vuong and Tran [2019]. Further, Abbas et al. [2020] address uncertainty due to model inadequacy.

Despite the importance of reliable model-based rollouts, adapting rollout length based on model accuracy has received little attention. An exception is the M2AC algorithm Pan et al. [2020], which is closest to this work. M2AC schedules rollout lengths using a rank-based filtering heuristic depending on model uncertainty. They introduce a reward penalty for model uncertainty similar to offline MBRL approaches, which can, however, hinder the expansion of the known subset of $\mathcal{S}$ in an online setting. Further, the uncertainty estimate of M2AC is comparatively brittle as presented in Appendix C.5. The proposed rollout scheme of MACURA, instead, takes a strictly spatial perspective on rollout length which is conceptually different from M2AC and generally new in Dyna-style MBRL.

9 Conclusion

Learning predictive dynamics models to tame data requirements of RL is an established concept, leading to state-of-the-art MBRL approaches like MBPO that achieve high data efficiency and asymptotic performance. This work builds on the successful MBPO architecture, addressing the critical question of adapting the length of model-based rollouts. While it is common knowledge that models are only helpful *where* they are accurate, and they are accurate only *where* they have seen data, only few works in MBRL address model accuracy in general, and its spatial nature in particular. We make the consideration of model accuracy as a local property a fundamental building block of our theoretical analysis of Dyna-style MBRL, which provides us with an effective mechanism for model usage. Combining this mechanism with an easy-to-compute and expressive estimate for model accuracy, we propose the Model-Based Actor-Critic with Uncertainty-Aware Rollout Adpation (MACURA) algorithm. Benchmarking on MuJoCo, we show that MACURA outperforms the current state-of-the-art substantially concerning data efficiency and asymptotic performance. Finally, the rollout mechanism of MACURA solely introduces one essential hyperparameter, making it considerably easier to tune than competitor approaches.

Acknowledgments and Disclosure of Funding

ZF Friedrichshafen AG partially funded this research. Furthermore, the research was in part supported by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) through the project EEMotion. Computations were performed with computing resources granted by RWTH Aachen University under projects rwth1428, rwth1472, and rwth1552.

References

Z. Abbas, S. Sokota, E. J. Talvitie, and M. White. Selective Dyna-style Planning Under Limited Model Capacity. *arXiv*, July 2020. doi: 10.48550/arXiv.2007.02418.

J. Adamy. *Nonlinear Systems and Controls*. Springer, Berlin, Germany, 2022. ISBN 978-3-662-65633-4.

J. Buckman, D. Hafner, G. Tucker, E. Brevdo, and H. Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *Int. Conf. on Neural Information Processing Systems*. 2018.

X. Chen, C. Wang, Z. Zhou, and K. Ross. Randomized Ensembled Double Q-Learning: Learning Fast Without a Model. *Int. Conf. on Learning Representations*, 2021.

K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. *Adv. in Neural Information Processing Systems*, 2018.

O. Eberhard, J. Hollenstein, C. Pinneri, and G. Martius. Pink noise is all you need: Colored noise exploration in deep reinforcement learning. In *Int. Conf. on Learning Representations*, 2023.

B. Frauenknecht, T. Ehlgen, and S. Trimpe. Data-efficient deep reinforcement learning for vehicle trajectory control. *IEEE Int. Conf. on Intelligent Transportation Systems*, 2023.

L. P. Fröhlich, M. Lefarov, M. N. Zeilinger, and F. Berkenkamp. On-Policy Model Errors in Reinforcement Learning. *Int. Conf. on Learning Representations*, 2022.

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Int. Conf. on Machine Learning*. 2018a.

T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine. Soft Actor-Critic Algorithms and Applications. *arXiv*, 2018b.

M. Janner, J. Fu, M. Zhang, and S. Levine. When to trust your model: model-based policy optimization. In *Int. Conf. on Neural Information Processing Systems*. Curran Associates Inc., 2019.

J. Jeong, X. Wang, M. Gimelfarb, H. Kim, B. Abdulhai, and S. Sanner. Conservative Bayesian Model-Based Value Expansion for Offline Policy Optimization. *Int. Conf. on Learning Representations*, Oct. 2023. doi: 10.48550/arXiv.2210.03802.

T. Ji, Y. Luo, F. Sun, M. Jing, F. He, and W. Huang. When to update your model: constrained model-based reinforcement learning. In *Int. Conf. on Neural Information Processing Systems*, pages 23150–23163. Curran Associates Inc., Red Hook, NY, USA, Nov. 2022. ISBN 978-1-71387108-8. doi: 10.5555/3600270.3601952.

G. Kalweit and J. Boedecker. Uncertainty-driven Imagination for Continuous Deep Reinforcement Learning. In *Conf. on Robot Learning*. 2017.

I. Kostrikov, L. M. Smith, and S. Levine. Demonstrating A Walk in the Park: Learning to Walk in 20 Minutes With Model-Free Reinforcement Learning, 2023.

H. Lai, J. Shen, W. Zhang, and Y. Yu. Bidirectional Model-based Policy Optimization. In *Int. Conf. on Machine Learning*. PMLR, 2020.

H. Lai, J. Shen, W. Zhang, Y. Huang, X. Zhang, R. Tang, Y. Yu, and Z. Li. On Effective Scheduling of Model-based Reinforcement Learning. *Int. Conf. on Neural Information Processing Systems*, 2021.

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Int. Conf. on Neural Information Processing Systems*. 2017.

L. Ljung. System Identification. In *Signal Analysis and Prediction*. Birkhäuser, Boston, MA, 1998.

C. Lu, P. J. Ball, J. Parker-Holder, M. A. Osborne, and S. J. Roberts. Revisiting Design Choices in Offline Model-Based Reinforcement Learning. *arXiv*, Oct. 2021. doi: 10.48550/arXiv.2110.04135.

C. E. Luis, A. G. Bottero, J. Vinogradska, F. Berkenkamp, and J. Peters. Model-Based Epistemic Variance of Values for Risk-Aware Policy Optimization. *arXiv*, 2023a.

C. E. Luis, A. G. Bottero, J. Vinogradska, F. Berkenkamp, and J. Peters. Model-Based Uncertainty in Value Functions. *Int. Conf. on Artificial Intelligence and Statistics*, 2023b.

Y. Luo, H. Xu, Y. Li, Y. Tian, T. Darrell, and T. Ma. Algorithmic Framework for Model-based Deep Reinforcement Learning with Theoretical Guarantees. *arXiv*, 2018.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2015.

A. S. Morgan, D. Nandha, G. Chalvatzaki, C. D'Eramo, A. M. Dollar, and J. Peters. Model Predictive Actor-Critic: Accelerating Robot Skill Acquisition with Deep Reinforcement Learning. In *IEEE Int. Conf. on Robotics and Automation*. IEEE Press, 2021.

N. M. Nguyen, A. Singh, and K. Tran. Improving model-based rl with adaptive rollout using uncertainty estimation. 2018.

F. Nielsen. On the Jensen–Shannon Symmetrization of Distances Relying on Abstract Means. *Entropy*, 2019.

OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang. Solving rubik's cube with a robot hand. *arXiv preprint*, 2019a.

OpenAI, C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. de Oliveira Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang. Dota 2 with large scale deep reinforcement learning. 2019b.

F. Pan, J. He, D. Tu, and Q. He. Trust the model when it is confident: masked model-based actor-critic. In *Int. Conf. on Neural Information Processing Systems*. 2020.

L. Pineda, B. Amos, A. Zhang, N. O. Lambert, and R. Calandra. MBRL-Lib: A Modular Library for Model-based Reinforcement Learning. *arXiv*, 2021.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms. *ArXiv*, 2017.

J. Shen, H. Zhao, W. Zhang, and Y. Yu. Model-based policy optimization with unsupervised model adaptation. In *Int. Conf. on Neural Information Processing Systems*. 2020.

R. S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 1991.

E. Todorov, T. Erez, and Y. Tassa. MuJoCo: A physics engine for model-based control. In *Int. Conf. on Intelligent Robots and Systems*. IEEE, 2012.

T.-L. Vuong and K. Tran. Uncertainty-aware Model-based Policy Optimization. *arXiv*, June 2019. doi: 10.48550/arXiv.1906.10717.

X. Wang, W. Wongkamjan, R. Jia, and F. Huang. Live in the Moment: Learning Dynamics Model Adapted to Evolving Policy. In *Int. Conf. on Machine Learning*, pages 36470–36493. PMLR, July 2023. URL `https://proceedings.mlr.press/v202/wang23an`.

Z. Wang, J. Wang, Q. Zhou, B. Li, and H. Li. Sample-Efficient Reinforcement Learning via Conservative Model-Based Actor-Critic. *AAAI Conf. on Artificial Intelligence*, 2022.

Z. Wu, C. Yu, C. Chen, J. Hao, and H. H. Zhuo. Plan to predict: learning an uncertainty-foreseeing model for model-based reinforcement learning. In *Int. Conf. on Neural Information Processing Systems*, pages 15849–15861. Curran Associates Inc., Red Hook, NY, USA, Nov. 2022. ISBN 978-1-71387108-8. doi: 10.5555/3600270.3601423.

P. Wurman, S. Barrett, K. Kawamoto, J. MacGlashan, K. Subramanian, W. T. J., R. Capobianco, A. Devlic, F. Eckert, F. Fuchs, L. Gilpin, P. Khandelwal, V. Kompella, H. Lin, P. MacAlpine, D. Oller, T. Seno, C. Sherstan, T. M. D., H. Aghabozorgi, L. Barrett, R. Douglas, D. Whitehead, P. Dürr, P. Stone, M. Spranger, and H. Kitano. Outracing champion Gran Turismo drivers with deep reinforcement learning. *Nature*, 2022.

T. Yu, G. Thomas, L. Yu, S. Ermon, J. Y. Zou, S. Levine, C. Finn, and T. Ma. Mopo: Model-based offline policy optimization. In *Adv. in Neural Information Processing Systems*, 2020.

Y. Zhai, Y. Li, Z. Gao, X. Gong, K. Xu, D. Feng, D. Bo, and H. Wang. Optimistic Model Rollouts for Pessimistic Offline Policy Optimization. *arXiv*, 2024.

C. Zhang, S. R. Kuppannagari, and V. K. Prasanna. Maximum Entropy Model Rollouts: Fast Model Based Policy Optimization without Compounding Errors. *arXiv*, 2020.

W. Zhang, X. Wang, J. Shen, and M. Zhou. Model-based Multi-agent Policy Optimization with Adaptive Opponent-wise Rollouts. *ResearchGate*, pages 3384–3391, Aug. 2021. doi: 10.24963/ijcai.2021/466.

A Pseudocode Algorithms

Algorithm 3 "Vanilla" Dyna-style Deep Model-based Reinforcement Learning

Initialize: dynamics model $\tilde{p}_\theta$, RL policy π, environment replay buffer $\mathcal{D}_{\text{env}} \leftarrow \emptyset$, model replay buffer $\mathcal{D}_{\text{mod}} \leftarrow \emptyset$, rollout length schedule for $T_{\max}$, steps before retraining R, number of model-based rollouts M, RL update steps G.

for each iteration **do**
  $s_0 \sim \rho_0(s)$
  for each environment step **do**
    $a_t \sim \pi(\cdot \mid s_t)$
    $s_{t+1} \sim p(\cdot \mid s_t, a_t)$
    $r_{t+1} = r(s_t, a_t)$
    $\mathcal{D}_{\text{env}} \leftarrow \mathcal{D}_{\text{env}} \cup \{(s_t, a_t, r_{t+1}, s_{t+1})\}$
    if $\mod$ (environment step, R) $= 0$ **then**
      for each epoch **do**
        Train $\tilde{p}_\theta$ on $\mathcal{D}_{\text{env}}$
      end for
      Evict old data from $\mathcal{D}_{\text{mod}}$
      for $m \in M$ model rollouts **do**
        $s_0^m \sim \mathcal{U}(\mathcal{D}_{\text{env}})$
        for $t = 0, \ldots, T_{\max} - 1$ **do**
          $e_t^m \sim \mathcal{U}(1, \ldots, E)$
          $a_t^m \sim \pi(\cdot \mid s_t^m)$
          $s_{t+1}^m \sim \tilde{p}_{\theta_{e_t^m}}(\cdot \mid s_t^m, a_t^m)$,
          $r_{t+1}^m = r(s_t^m, a_t^m)$
          $\mathcal{D}_{\text{mod}} \leftarrow \mathcal{D}_{\text{mod}} \cup \{(s_t^m, a_t^m, r_{t+1}^m, s_{t+1}^m)\}$
        end for
      end for
    end if
    for G gradient steps **do**
      Train π on $\mathcal{D}_{\text{mod}} \cup \mathcal{D}_{\text{env}}$
    end for
  end for
end for

Algorithm 4 Model-based Actor-Critic with Uncertainty-aware Rollout Adaption (MACURA)

Initialize: dynamics model $\tilde{p}_\theta$, RL policy π, environment replay buffer $\mathcal{D}_{\text{env}} \leftarrow \emptyset$, model replay buffer $\mathcal{D}_{\text{mod}} \leftarrow \emptyset$, steps before retraining R, number of model-based rollouts M, maximum RL update steps $G_{\max}$.
Initialize $\tilde{p}_\theta, \pi, \mathcal{D}_{\text{env}} \leftarrow \emptyset$, $\mathcal{D}_{\text{mod}} \leftarrow \emptyset$, fixed $T_{\max}$, ζ
for each iteration **do**
  $s_0 \sim \rho_0(s)$
  for each environment step **do**
    $a_t \sim \pi(\cdot \mid s_t)$ with correlated exploration noise Eberhard et al. [2023]
    $s_{t+1} \sim p(\cdot \mid s_t, a_t)$
    $r_{t+1} = r(s_t, a_t)$
    $\mathcal{D}_{\text{env}} \leftarrow \mathcal{D}_{\text{env}} \cup \{(s_t, a_t, r_{t+1}, s_{t+1})\}$
    if mod (environment step, R) $= 0$ **then**
      $K \leftarrow K + 1$
      $k \leftarrow K$
      for each epoch **do**
        Train $\tilde{p}_\theta$ on $\mathcal{D}_{\text{env}}$
      end for
      Evict old data from $\mathcal{D}_{\text{mod}}$
      for $m \in M$ model rollouts **do**
        $s_0^m \sim \mathcal{U}(\mathcal{D}_{\text{env}})$
        $e_0^m \sim \mathcal{U}(1, \dots, E)$
        $a_0^m \sim \pi(\cdot \mid s_0^m)$
        $s_1^m \sim \tilde{p}_{\theta_{e_0^m}}(\cdot \mid s_0^m, a_0^m)$
        $r_1^m = r(s_0^m, a_0^m)$
        $u_{\text{GJS}}(s_0^m, a_0^m)$ according to (11)
        if $u_{\text{GJS}}(s_0^m, a_0^m) < \kappa$ **then**
          $\mathcal{D}_{\text{mod}} \leftarrow \mathcal{D}_{\text{mod}} \cup \{(s_0^m, a_0^m, r_1^m, s_1^m)\}$
        else
          stop rollout m and discard data
        end if
      end for
      $\hat{u}_{\text{GJS},k} = \inf \left\{ u_{\text{GJS}}(s_0, a_0) \in \left\{ u_{\text{GJS}}(s_0^1, a_0^1), \dots, u_{\text{GJS}}(s_0^M, a_0^M) \right\} : \zeta \leq \text{CDF}_k(u_{\text{GJS}}(s_0, a_0)) \right\}$[3]
      $\kappa \leftarrow \frac{\xi}{K} \sum_{k=1}^{K} \hat{u}_{\text{GJS},k}$
      for $t = 1, \dots, T_{\max} - 1$ **do**
        for $m \in M$ model rollouts **do**
          $e_t^m \sim \mathcal{U}(1, \dots, E)$
          $a_t^m \sim \pi(\cdot \mid s_t^m)$
          $s_{t+1}^m \sim \tilde{p}_{\theta_{e_t^m}}(\cdot \mid s_t^m, a_t^m)$,
          $r_{t+1}^m = r(s_t^m, a_t^m)$
          $u_{\text{GJS}}(s_t^m, a_t^m)$ according to (11)
          if $u_{\text{GJS}}(s_t^m, a_t^m) < \kappa$ **then**
            $\mathcal{D}_{\text{mod}} \leftarrow \mathcal{D}_{\text{mod}} \cup \{(s_t^m, a_t^m, r_{t+1}^m, s_{t+1}^m)\}$
          else
            stop rollout m and discard data
          end if
        end for
      end for
    end if
    for $G = \left\lfloor G_{\max} \frac{|\mathcal{D}_{\text{mod}}|}{|\mathcal{D}_{\text{mod}}|_{\max}} \right\rceil$ gradient steps **do**
      Update π on $\mathcal{D}_{\text{mod}} \cup \mathcal{D}_{\text{env}}$
    end for
  end for
end for

B Proofs

Lemma B.1 (Return mismatch with respect to state distribution shift)**.** *Be the expected return following policy π in $\hat{\mathcal{M}}$*

$$\mathbb{E}_{s\sim\hat{\mathcal{M}},a\sim\pi}\left[\sum_{t=0}^{T(\omega)}\gamma^t r_{t+1}^{\hat{\mathcal{M}}}\right] := \eta[\pi]$$

and the expected return following the same policy in $\widetilde{\mathcal{M}}$

$$\mathbb{E}_{s\sim\widetilde{\mathcal{M}},a\sim\pi}\left[\sum_{t=0}^{T(\omega)}\gamma^t r_{t+1}^{\widetilde{\mathcal{M}}}\right] := \tilde{\eta}[\pi],$$

then

$$|\eta[\pi]-\tilde{\eta}[\pi]| \leq 2{r_{\max}}^{4}\sum_{t=0}^{T(\omega)}\gamma^t D_{\mathrm{TV}}\left(p^t(s)\|\tilde{p}^t(s)\right)^{5}.$$

Proof.

$$\begin{aligned}
&|\eta[\pi]-\tilde{\eta}[\pi]| \\
&= \left|\mathbb{E}_{s\sim\hat{\mathcal{M}},a\sim\pi}\left[\sum_{t=0}^{T(\omega)}\gamma^t r_{t+1}^{\hat{\mathcal{M}}}\right] - \mathbb{E}_{s\sim\widetilde{\mathcal{M}},a\sim\pi}\left[\sum_{t=0}^{T(\omega)}\gamma^t r_{t+1}^{\widetilde{\mathcal{M}}}\right]\right| \\
&= \left|\int_{\mathcal{E}}\int_{\mathcal{A}}\left(\sum_{t=0}^{T(\omega)}\gamma^t\left(p^t(s,a)-\tilde{p}^t(s,a)\right)\right) r(s,a)\,da\;ds\right| \\
&= \left|\sum_{t=0}^{T(\omega)}\int_{\mathcal{E}}\int_{\mathcal{A}}\gamma^t\left(p^t(s,a)-\tilde{p}^t(s,a)\right) r(s,a)\,da\;ds\right| \\
&\leq \sum_{t=0}^{T(\omega)}\int_{\mathcal{E}}\int_{\mathcal{A}}\gamma^t\left|p^t(s,a)-\tilde{p}^t(s,a)\right| r(s,a)\,da\;ds \\
&\leq r_{\max}\sum_{t=0}^{T(\omega)}\int_{\mathcal{E}}\int_{\mathcal{A}}\gamma^t\left|p^t(s,a)-\tilde{p}^t(s,a)\right|\,da\;ds \\
&= r_{\max}\sum_{t=0}^{T(\omega)}\int_{\mathcal{E}}\int_{\mathcal{A}}\gamma^t\left|\left(p^t(s)-\tilde{p}^t(s)\right)\pi(a\mid s)\right|\,da\;ds \\
&= r_{\max}\sum_{t=0}^{T(\omega)}\gamma^t\int_{\mathcal{E}}\left|p^t(s)-\tilde{p}^t(s)\right|\,ds \\
&= 2r_{\max}\sum_{t=0}^{T(\omega)}\gamma^t D_{\mathrm{TV}}\left(p^t(s)\|\tilde{p}^t(s)\right)
\end{aligned}$$

□

[3] $\mathrm{CDF}_k(u_{\mathrm{GJS}}(s_0,a_0))$ denotes the cumulative distribution function of GJS uncertainty estimates at time step 0 at the k^{th} round of model-based rollouts.

[4] Dynamics mismatch is especially an issue in high-rewarding areas of the state-action space. In this work, however, we will neglect the dynamics-reward coupling and instead, only focus on the dynamics mismatch. Thus, we consider the conservative upper bound of $r_{\max}$.

[5] We follow the common abuse of notation introduced in Janner et al. [2019], formulating the TV distance with respect to the probability densities rather than the stochastic process as would be formally correct.

Lemma B.2 (Recursive Formulation)**.** *Be $D_{TV}\left(p^t(\iota)\|\tilde{p}^t(\iota)\right) := \epsilon_t$ for an arbitrary ι and by construction $\epsilon_0 = 0$. Further be $\mathbb{E}_{s\sim\tilde{p}^{t-1}}\left[D_{TV}\left(p\left(s' \mid s\right)\|\tilde{p}\left(s' \mid s\right)\right)\right] := \delta_t$ and $\delta_0 = \epsilon_0 = 0$. Then we can bound ϵ_t by*

$$\epsilon_t \leq \sum_{\tau=0}^{t} \delta_\tau$$

Proof.

$$\begin{aligned}
&\epsilon_t = D_{\mathrm{TV}}\left(p^t(s')\|\tilde{p}^t(s')\right)\\
&= \frac{1}{2}\int_{\mathcal{E}} \left|p^t(s') - \tilde{p}^t(s')\right| \, ds'\\
&= \frac{1}{2}\int_{\mathcal{E}} \left|\int_{\mathcal{E}} p(s' \mid s)p^{t-1}(s) - \tilde{p}(s' \mid s)\tilde{p}^{t-1}(s)\, ds\right| \, ds'\\
&\leq \frac{1}{2}\int_{\mathcal{E}}\int_{\mathcal{E}} \left|p(s' \mid s)p^{t-1}(s) - \tilde{p}(s' \mid s)\tilde{p}^{t-1}(s)\right| \, ds \; ds'\\
&= \frac{1}{2}\int_{\mathcal{E}}\int_{\mathcal{E}} \left|p(s' \mid s)p^{t-1}(s) - p(s' \mid s)\tilde{p}^{t-1}(s) + p(s' \mid s)\tilde{p}^{t-1}(s) - \tilde{p}(s' \mid s)\tilde{p}^{t-1}(s)\right| \, ds \; ds'\\
&\leq \frac{1}{2}\int_{\mathcal{E}}\int_{\mathcal{E}} \tilde{p}^{t-1}(s)\left|p(s' \mid s) - \tilde{p}(s' \mid s)\right| \, ds \; ds' \; + \frac{1}{2}\int_{\mathcal{S}}\int_{\mathcal{E}} p(s' \mid s)\left|p^{t-1}(s) - \tilde{p}^{t-1}(s)\right| \, ds \; ds'\\
&= \mathbb{E}_{s\sim\tilde{p}^{t-1}}\left[\frac{1}{2}\int_{\mathcal{E}} \left|p(s' \mid s) - \tilde{p}(s' \mid s)\right| \, ds'\right] + \frac{1}{2}\int_{\mathcal{E}} \left|p^{t-1}(s) - \tilde{p}^{t-1}(s)\right| \, ds\\
&= \mathbb{E}_{s\sim\tilde{p}^{t-1}}\left[D_{\mathrm{TV}}\left(p(s' \mid s)\|\tilde{p}(s' \mid s)\right)\right] + D_{\mathrm{TV}}\left(p^{t-1}(s)\|\tilde{p}^{t-1}(s)\right)\\
&= \delta_t + \epsilon_{t-1} = \delta_t + \delta_{t-1} + \epsilon_{t-2} = \ldots =\\
&= \epsilon_0 + \sum_{\tau=1}^{t}\delta_\tau = \delta_0 + \sum_{\tau=1}^{t}\delta_\tau = \sum_{\tau=0}^{t}\delta_\tau
\end{aligned}$$

□

Lemma B.3 (Dependency on dynamics mismatch)**.** *Be*

$$\Delta p_{\mathcal{E}}[\pi] := \sup_{s\in\mathcal{E}, a\sim\pi}\left\{D_{TV}\left(p\left(s' \mid s, a\right)\|\tilde{p}\left(s' \mid s, a\right)\right)\right\}$$

then

$$\mathbb{E}_{s\sim\tilde{p}^{t-1}}\left[D_{TV}\left(p\left(s' \mid s\right)\|\tilde{p}\left(s' \mid s\right)\right)\right] \leq \Delta p_{\mathcal{E}}[\pi]$$

Proof.

$$\begin{aligned}
&\mathbb{E}_{s\sim\tilde{p}^{t-1}}\left[D_{TV}\left(p\left(s' \mid s\right)\|\tilde{p}\left(s' \mid s\right)\right)\right]\\
&= \frac{1}{2}\int_{\mathcal{E}}\int_{\mathcal{E}} \tilde{p}^{t-1}\left(s\right)\left|p\left(s' \mid s\right) - \tilde{p}\left(s' \mid s\right)\right| \, ds \; ds'\\
&= \frac{1}{2}\int_{\mathcal{E}}\int_{\mathcal{E}} \tilde{p}^{t-1}\left(s\right)\left|\int_{\mathcal{A}}\left(p\left(s' \mid s, a\right) - \tilde{p}\left(s' \mid s, a\right)\right)\pi(a \mid s)\, da\right| \, ds \; ds'\\
&\leq \frac{1}{2}\int_{\mathcal{E}}\int_{\mathcal{E}}\int_{\mathcal{A}} \tilde{p}^{t-1}\left(s\right)\pi\left(a \mid s\right)\left|p\left(s' \mid s, a\right) - \tilde{p}\left(s' \mid s, a\right)\right| \, da \; ds \; ds'\\
&= \mathbb{E}_{s\sim\tilde{p}^{t-1}, a\sim\pi}\left[\frac{1}{2}\int_{\mathcal{E}}\left|p\left(s' \mid s, a\right) - \tilde{p}\left(s' \mid s, a\right)\right| \, ds'\right]\\
&= \mathbb{E}_{s\sim\tilde{p}^{t-1}, a\sim\pi}\left[D_{\mathrm{TV}}\left(p\left(s' \mid s, a\right)\|p\left(s' \mid s, a\right)\right)\right]\\
&\leq \Delta p_{\mathcal{E}}[\pi]
\end{aligned}$$

□

Theorem B.4 (Monotonic Improvement under Dynamics Misalignment on $\mathcal{E} \subseteq \mathcal{S}$)**.** *We define two MDPs $\hat{\mathcal{M}}$ and $\widetilde{\mathcal{M}}$ with a common state space $\mathcal{S}$, action space $\mathcal{A}$ and reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^+$* [6]*. $\hat{\mathcal{M}}$ has dynamics $p(s' \mid s, a) : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, while $\widetilde{\mathcal{M}}$ has dynamics $\tilde{p}(s' \mid s, a) : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. For both MDPs, we define probability densities*

$$\mathbb{P}[s_t^{\hat{\mathcal{M}}} \in \mathcal{B}] = \int_{\mathcal{B}} p^t(s)\, ds$$

$$\mathbb{P}[s_t^{\hat{\mathcal{M}}} \in \mathcal{B}, a_t^{\hat{\mathcal{M}}} \in \mathcal{C}] = \int_{\mathcal{B}} \int_{\mathcal{C}} p^t(s, a)\, da\ ds$$

as well as a dynamics function

$$p(s' \mid s) = \int_{\mathcal{A}} p(s' \mid s, a)\pi(a \mid s)\, da$$

and equivalently,

$$\mathbb{P}[s_t^{\widetilde{\mathcal{M}}} \in \mathcal{B}] = \int_{\mathcal{B}} \tilde{p}^t(s)\, ds$$

$$\mathbb{P}[s_t^{\tilde{\mathcal{M}}} \in \mathcal{B}, a_t^{\tilde{\mathcal{M}}} \in \mathcal{C}] = \int_{\mathcal{B}} \int_{\mathcal{C}} \tilde{p}^t(s, a)\, da\ ds$$

$$\tilde{p}(s' \mid s) = \int_{\mathcal{A}} \tilde{p}(s' \mid s, a)\pi(a \mid s)\, da$$

for all Borel-measuarble sets $\mathcal{B} \subseteq \mathcal{S}, \mathcal{C} \subseteq \mathcal{A}$ and conditional probability densities $\pi(a \mid s) : \mathcal{S} \to \mathcal{A}$.

Further, we define a coupling between $\hat{\mathcal{M}}$ and $\widetilde{\mathcal{M}}$ via a random stopping time

$T^{\hat{\mathcal{M}}}(\omega) := \min\{t \in \mathbb{N} \mid s_t^{\hat{\mathcal{M}}}(\omega) \in \mathcal{E}^C\}, T^{\tilde{\mathcal{M}}}(\omega) := \min\{t \in \mathbb{N} \mid s_t^{\tilde{\mathcal{M}}}(\omega) \in \mathcal{E}^C\}, T(\omega) := \min\{T^{\hat{\mathcal{M}}}, T^{\tilde{\mathcal{M}}}\} - 1,$

where s_t is a trajectory with respect to the MDPs $\hat{\mathcal{M}}$ or $\widetilde{\mathcal{M}}$ respectively and regarded as a random variable, $\mathcal{E} \subseteq \mathcal{S}$, and $\mathcal{E}^C$ the compliment of $\mathcal{E}$.

as well as identical start states $s_0^{\widetilde{\mathcal{M}}} = s_0^{\hat{\mathcal{M}}} \in \mathcal{E}$.

Suppose the expected return following policy π in $\hat{\mathcal{M}}$ is denoted by

$$\mathbb{E}_{s \sim \hat{\mathcal{M}}, a \sim \pi}\left[\sum_{t=0}^{T(\omega)} \gamma^t r_{t+1}^{\hat{\mathcal{M}}}\right] := \eta[\pi]$$

and $\tilde{\eta}[\pi]$ describes the expected return following the same policy in $\widetilde{\mathcal{M}}$

$$\mathbb{E}_{s \sim \widetilde{\mathcal{M}}, a \sim \pi}\left[\sum_{t=0}^{T(\omega)} \gamma^t r_{t+1}^{\widetilde{\mathcal{M}}}\right] := \tilde{\eta}[\pi],$$

then we can define a lower bound for $\eta[\pi]$ of the form

$$\eta[\pi] \geq \tilde{\eta}[\pi] - 2r_{\max} \sum_{t=0}^{T(\omega)} \gamma^t \sum_{\tau=0}^{t} \Delta p_{\mathcal{E}}[\pi].$$

Proof.

$$\eta[\pi] \geq \tilde{\eta}[\pi] - |\eta[\pi] - \tilde{\eta}[\pi]|$$

Using Lemma B.1

$$\eta[\pi] \geq \tilde{\eta}[\pi] - 2r_{\max} \sum_{t=0}^{T(\omega)} \gamma^t D_{TV}\left(p^t(s) \| \tilde{p}^t(s)\right)$$

[6] We assume rewards to be strictly positive. An equivalent reward function can be trivially constructed from any bounded reward function $r(s, a) \in [r_{\min}, r_{\max}] \forall s \in \mathcal{S}, a \in \mathcal{A}$

Using Lemma B.2

$$\eta[\pi] \geq \tilde{\eta}[\pi] - 2r_{\max} \sum_{t=0}^{T(\omega)} \gamma^t \sum_{\tau=0}^{t} \mathbb{E}_{s \sim \tilde{p}^{\tau-1}} \left[D_{TV} \left(p\left(s' \mid s\right) \| \tilde{p}\left(s' \mid s\right) \right) \right]$$

Using Lemma B.3

$$\eta[\pi] \geq \tilde{\eta}[\pi] - 2r_{\max} \sum_{t=0}^{T(\omega)} \gamma^t \sum_{\tau=0}^{t} \Delta p_{\mathcal{E}}[\pi]$$

□

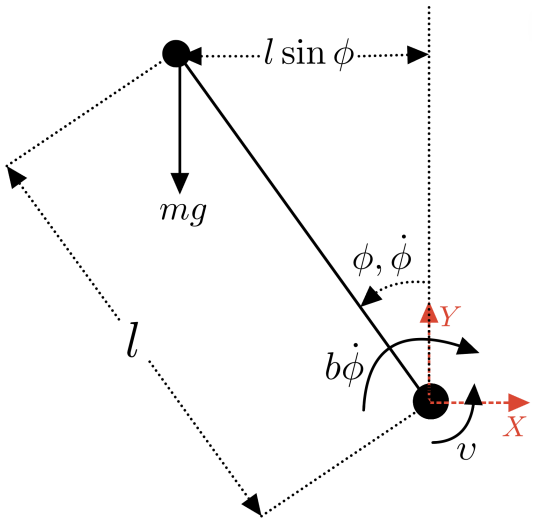


Figure 5: Free body diagram of the pendulum.

C Toy Example

In order to test the proposed uncertainty measure u_{GJS} (11), we use a two-dimensional toy example of a pendulum. We discuss the dynamics of the pendulum in Section C.1, the feedback controller used as policy in Section C.2, and the experimental setup leading to Figure 2 in Section C.3. We further compare our uncertainty estimate to the one proposed in Pan et al. [2020] and show a substantially more reliable behavior of the u_{GJS} measure in Section C.5.

C.1 Pendulum Dynamics

Figure 5 shows a free-body diagram of the pendulum near the upper fixed point. We can write the equation of motion for such a pendulum as

$$ml^2\ddot{\phi} - mgl\sin\phi + b\phi = \upsilon, \tag{16}$$

where m is the mass of the pendulum, l is its length, g is the acceleration due to gravity, and b is the coefficient of viscous friction. Further, υ is the torque applied at the base of the pendulum that is used to control the pendulum. We define the continuous-time state as $x = \begin{bmatrix}\phi & \dot{\phi}\end{bmatrix}^T$ with $\phi \in [-3, 3][\mathrm{rad}]$ the pendulum angle and $\dot{\phi} \in [-16, 16][\frac{\mathrm{rad}}{\mathrm{s}}]$ the pendulum's angular velocity. The nonlinear state-space equation for this system in continuous time are

$$\dot{x} = \begin{bmatrix} \dot{\phi} \\ \frac{1}{ml^2}\left(\upsilon + mgl\sin\phi - b\dot{\phi}\right) \end{bmatrix} = f(x, \upsilon). \tag{17}$$

When considering the system in discrete time, we sample observations and apply actions $a = \upsilon$ at discrete time points separated by a fixed time interval Δt. We simulate the pendulum by integrating the state equation (17) between the sampling time steps, keeping the applied action fixed throughout the integration. We additionally consider homoscedastic Gaussian process noise with covariance matrix Σ. Hence we obtain the discrete-time MDP dynamics as

$$p(\cdot|s, a) = \mathcal{N}\left(\mu(s, a), \Sigma\right), \tag{18}$$

with,

$$\mu(s, a) = \int_0^{\Delta t} f(x(t), a)dt, \tag{19}$$

where $x(0) = s$. The values for the different parameters can be seen in Table 1.

Table 1: Parameters of the Pendulum.

Parameter	Value
mass m	0.1
length l	1
acceleration due to gravity g	9.81
coefficient of viscous friction b	0.1
sampling time interval Δt	0.01
process noise covariance matrix Σ	$\begin{bmatrix} 10^{-6} & 0 \\ 0 & 10^{-3} \end{bmatrix}$

C.2 Controller

The pendulum is controlled via the applied torque υ. We can use feedback linearization Adamy [2022] to obtain a controller of the form

$$\pi_{\mathrm{FL}}(s) = \upsilon_{\mathrm{FL}}(x) = ml^2 \left(\ddot{\phi}^d + K_D(\dot{\phi}^d - \dot{\phi}) + K_P(\phi^d - \phi) \right) - mgl \sin \phi + b\dot{\phi}, \tag{20}$$

where ϕ^d, $\dot{\phi}^d$, and $\ddot{\phi}^d$ denote the desired pendulum angle, angular velocity, and angular acceleration, respectively. The positive constants K_P and K_D denote the proportional and derivative gains respectively. We choose them to give under-damped feedback dynamics. The values can be seen in Table 2. For our purpose, we use the upper fixed point as the desired position, that is, $\phi^d = 0$, $\dot{\phi}^d = 0$ and $\ddot{\phi}^d = 0$.

Table 2: Parameters of the Controller.

Parameter	Value
proportional gain K_P	25
derivative gain K_D	1

C.3 Data Generation

We create interaction data that is stored $\mathcal{D}_{\mathrm{env}}$ to train a PE model. For this, we let the feedback linearization controller (20) interact with the pendulum (17). Each trajectory starts from a fixed start state $\begin{bmatrix} \phi_0 & \dot{\phi}_0 \end{bmatrix}^T$. The trajectories are terminated upon reaching $T_{\max}$ steps. Ten such trajectories are generated. This results in a data distribution with a characteristic spiral shape as depicted in Figure 2a and 6a. The parameters used for generating the trajectories can be found in Table 3.

Table 3: Trajectory parameters for data generation.

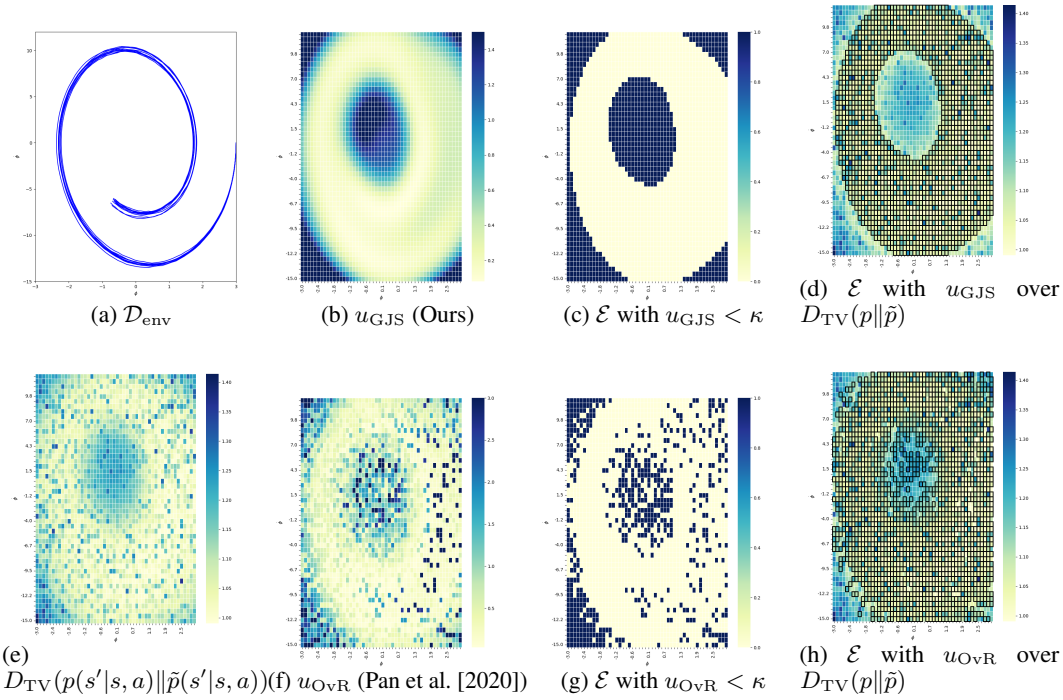
Parameter	Value
initial angle ϕ_0	3
initial angular velocity $\dot{\phi}_0$	0
number of steps $T_{\max}$	170

C.4 Illustrating the u_{GJS} Uncertainty Measure

To illustrate the effectiveness of the proposed uncertainty measure u_{GJS}, we investigate the connection between dynamics misalignment $D_{\mathrm{TV}}\left(p\left(s' \mid s, a\right) \| \tilde{p}\left(s' \mid s, a\right)\right)$ and model uncertainty on the toy example.

We train a PE dynamics model on data created according to Section C.3 and evaluate both quantities over $\mathcal{S}$. To do this, we discretize the state space into a uniform grid $\{s_{ij}\}$ where i is used to index over ϕ and j is used to index over $\dot{\phi}$. For each s_{ij} the corresponding action is obtained as

$$a_{ij} = \pi_{\mathrm{FL}}(s_{ij}). \tag{21}$$



(a) $\mathcal{D}_{\text{env}}$ (b) u_{GJS} (Ours) (c) $\mathcal{E}$ with $u_{\text{GJS}} < \kappa$ (d) $\mathcal{E}$ with u_{GJS} over $D_{\text{TV}}(p\|\tilde{p})$

(e) $D_{\text{TV}}(p(s'|s,a)\|\tilde{p}(s'|s,a))$ (f) u_{OvR} (Pan et al. [2020]) (g) $\mathcal{E}$ with $u_{\text{OvR}} < \kappa$ (h) $\mathcal{E}$ with u_{OvR} over $D_{\text{TV}}(p\|\tilde{p})$

Figure 6: Effectiveness of the GJS uncertainty measure

First, we evaluate dynamics misalignment $D_{\text{TV}}\left(p\left(s' \mid s,a\right)\|\tilde{p}\left(s' \mid s,a\right)\right)$ over the predefined grid. We obtain the true and predicted next state distribution as

$$p_{ij} = p(\cdot|s_{ij}, a_{ij}), \tag{22}$$

querying the dynamics of the toy example (18), and

$$\tilde{p}_{ij,e} = \tilde{p}_{\theta_e}(\cdot|s_{ij}, a_{ij}) \tag{23}$$

for each PNN prediction with $e \in (1 \dots E)$.

As there is no closed-form solution for computing $D_{\text{TV}}\left(p_{ij}\|\tilde{p}_{ij}\right)$ we use the common upper bound of the total variation distance with respect to the Hellinger distance D_{H}:

$$D_{\text{TV}}\left(p_{ij}\|\tilde{p}_{ij}\right) \leq \sqrt{2} D_{\text{H}}\left(p_{ij}\|\tilde{p}_{ij}\right). \tag{24}$$

Figures 2b and 6e show a heatmap over $\mathcal{S}$ of the discretized dynamics misalignment measurements

$$d_{ij} = \frac{1}{E}\sum_{e=1}^{E} \sqrt{2} D_{\text{H}}(p_{ij}||\tilde{p}_{ij,e}). \tag{25}$$

Similarly, we evaluate model uncertainty in Figures 2c and 6b over $\mathcal{S}$ plotting

$$u_{\text{GJS},ij} = u_{\text{GJS}}(s_{ij}, a_{ij}). \tag{26}$$

As discussed in Section 5.3, both align well. Choosing a suitable κ allows to construct a meaningful set $\mathcal{E}$ as depicted in Figures 2d and 6c that aligns with areas of low dynamics misalignment as illustrated in Figure 6d.

C.5 Illustrating the u_{OvR} Uncertainty Measure of the M2AC Algorithm Pan et al. [2020]

Similar to MACURA, Pan et al. [2020] propose to adapt the length of branched model-based rollouts in Dyna-style MBRL using model uncertainty. Therefore, they present the One-versus-Rest u_{OvR} uncertainty estimate for PE models:

$$u_{\text{OvR}}(s,a) = D_{\text{KL}}\left(\mathcal{N}(\mu_{\theta_e}(s,a), \Sigma_{\theta_e}(s,a))\|\mathcal{N}(\mu_{\theta_{-e}}(s,a), \Sigma_{\theta_{-e}}(s,a))\right). \tag{27}$$

This uncertainty estimate is defined as the Kullback-Leibler divergence between a randomly chosen PNN prediction $e \sim \mathcal{U}(1, \dots, E)$ and a Gaussian distribution defined by merging the remaining PNNs of the PE model, such that

$$\mu_{\theta_{-e}}(s,a) = \frac{1}{E-1} \sum_{f \neq e}^{E} \mu_{\theta_f}(s,a) \tag{28}$$

and

$$\Sigma_{\theta_{-e}}(s,a) = \frac{1}{E-1} \sum_{f \neq e}^{E} \left(\Sigma_{\theta_f}(s,a) + \mu_{\theta_f}(s,a) \mu_{\theta_f}(s,a)^\top \right) - \mu_{\theta_{-e}}(s,a) \mu_{\theta_{-e}}(s,a)^\top. \tag{29}$$

We evaluate the connection between dynamics misalignment and the uncertainty estimate u_{OvR} on the pendulum toy example. Here we use the exact same setup as the one discussed in Section C.4, including the identical PE model. The only difference is that we compute the model uncertainty according to

$$u_{\mathrm{OvR},ij} = u_{\mathrm{OvR}}(s_{ij}, a_{ij}). \tag{30}$$

The evaluation over $\mathcal{S}$ is depicted in Figure 6f. We see a substantially more noisy uncertainty estimate of u_{OvR} as compared to the u_{GJS} uncertainty estimate proposed in MACURA and depicted in Figure 6b. Most importantly, we see u_{OvR} to be overconfident in areas around $\phi = 0$ and $\dot{\phi} = 0$ where the u_{OvR} model uncertainty is low, while dynamics misalignment is high as can be seen from Figure 6e.

Trying to construct a subset

$$\mathcal{E}_{\mathrm{OvR}} := \{ s \in \mathcal{S} \mid u_{\mathrm{OvR}}(s,a) < \kappa_{\mathrm{OvR}}, a \sim \pi(\cdot \mid s) \}. \tag{31}$$

choosing a suitable κ_{OvR} does not yield a reasonable result as shown in Figure 6g. The set has no clear boundary, due to the noisiness of the u_{OvR} uncertainty estimate. More importantly, the set does not align with areas of low dynamics misalignment, as depicted in Figure 6h, due to the overconfidence of u_{OvR}.

D Experiments

In the following, we discuss the details of the experimental setup in Section D.1 and different approaches to exploring the environment in Dyna-style MBRL in Section D.5.

D.1 Experimental Setup

Instead of the original implementation of MBPO[7], all implementations of this work are based on the more recent mbrl-lib library Pineda et al. [2021]. For MBPO and SAC, we use the implementations provided by the library, while M2AC is reimplemented as an extension to MBPO, as no open-source version of the M2AC code is available. We further implement the MACURA algorithm based on the mbrl-lib version of MBPO. The code is available online[8].

We run five random seeds for each experiment. Plots show the mean over the corresponding runs as a solid line and the 95% confidence interval as a shaded region.

In all experiments, the training data for the SAC agent comprises 95% $\mathcal{D}_{\text{mod}}$ and 5% $\mathcal{D}_{\text{env}}$ for MBPO, M2AC, and MACURA.

To achieve results comparable to the ones published in Janner et al. [2019], we tune the MBPO hyperparameters according to Table 4.

Table 4: Hyperparameters MBPO

Environment	Humanoid	Ant	Halfcheetah	Walker	Hopper
Epochs	300	300	400	200	125
Steps per Epoch	1000				
PNNs per PE	7				
PNN Layers	4				
PNN Nodes per Layer	400	200			
Critic Layers	3				
Critic Nodes per Layer	2048	1024			
Actor Layers	3				
Actor Nodes per Layer	2048	1024			
SAC Target Ent.	-10	-1	-4	-1	0
SAC Updates G	20		10	30	
Rollouts per Round M	406		203	406	
Rollouts length $T_{\max}$	$1 \to 25$	$1 \to 25$	1		$1 \to 15$
Episodes Schedule $T_{\max}$	$20 \to 300$	$20 \to 100$			$20 \to 100$

For MACURA, we choose the hyperparameters very close to MBPO for a fair comparison. In the MBPO implementation, the number of model-based rollouts per iteration M needs to be a multiple of the number of ensemble members. We remove this requirement for the MACURA rollout scheme, allowing us to choose M to plain hundreds. Also, we observe that the actual number of update steps G according to (15) in MACURA is roughly one-half of $G_{\max}$. Thus we choose $G_{\max}$ for MACURA to be twice as much as G in MBPO. We further introduce the hyperparameters $T_{\max}$, ζ, and ξ of the uncertainty-aware rollout adaption scheme of MACURA. Here, we keep $T_{\max}$ and ζ constant among all environments and solely tune ξ for the specific task. A comprehensive overview of the MACURA hyperparameters is provided in Table 5.

In the case of M2AC, we have issues producing stable results. This forces us to deviate from the hyperparameter setup of MBPO. Similar to MACURA, we remove the requirement for M to be a multiple of the number of ensemble members, thus choosing M to plain hundreds. In some environments, we observe M2AC yield better results with a fixed temperature parameter of the SAC agent, thus we disable automatic entropy tuning in these cases[9]. For the study of different

[7] `https://github.com/jannerm/mbpo`

[8] `https://github.com/Data-Science-in-Mechanical-Engineering/macura`

[9] If the SAC Target Ent. hyperparameter is shown as N/A in the hyperparameter tables, this means that automatic entropy tuning Haarnoja et al. [2018b] is not used for this case, instead a fixed temperature parameter Haarnoja et al. [2018a] of 0.2 is used.

Table 5: Hyperparameters MACURA

Environment	Humanoid	Ant	Halfcheetah	Walker	Hopper
Epochs	300	300	400	200	125
Steps per Epoch	1000				
PNNs per PE	7				
PNN Layers	4				
PNN Nodes per Layer	400	200			
Critic Layers	3				
Critic Nodes per Layer	2048	1024			
Actor Layers	3				
Actor Nodes per Layer	2048	1024			
SAC Target Ent.	-10	-1	-4	-1	0
SAC Updates $G_{\max}$	40		20	60	
Rollouts per Round M	400		200	400	
Rollouts length $T_{\max}$	10				
Quantile Factor ζ	0.95				
Scaling Factor ξ	5		2	0.3	30

Table 6: Hyperparameters M2AC (Deterministic Environment Policy)

Environment	Humanoid	Ant	Halfcheetah	Walker	Hopper
Epochs	300	300	400	200	125
Steps per Epoch	1000				
PNNs per PE	7				
PNN Layers	4				
PNN Nodes per Layer	200				
Critic Layers	3				
Critic Nodes per Layer	1024		512	1024	
Actor Layers	3				
Actor Nodes per Layer	1024		512	1024	
SAC Target Ent.	-17	N/A	-6	-1	0
SAC Updates G	20		10	30	
Rollouts per Round M	400		200	400	
Rollouts length $T_{\max}$	10				

exploration schemes in Section D.5, we find that hyperparameter settings of M2AC that are stable for deterministic environment interaction, destabilize when exploring and vice versa. Therefore, we provide two sets of hyperparameters. Table 6 represents the hyperparameter setting for deterministic interaction, and Table 7 shows the hyperparameters used when exploring with white or pink noise.

Finally, we choose the hyperparameters of the SAC baseline, according to Table 8. Similar to M2AC, SAC yields better results with a fixed temperature hyperparameter[10].

D.2 Tuning the Uncertainty-Aware Adaption Scheme

The uncertainty-aware rollout adaption scheme comprises three hyperparameters. We set $T_{\max} = 10$ and $\zeta = 95\%$ for all tasks and never tune them any further [11]. However, the scaling factor ξ requires application-specific tuning.

Figure 7b shows the influence of different choices of ξ on the performance on Humanoid. While we see that MACURA performs well for intermediate values for ξ, learning destabilizes for both high and low choices. This is expected for high values of ξ as they enforce model exploitation.

[10] If the SAC Target Ent. hyperparameter is shown as N/A in the hyperparameter table, this means that automatic entropy tuning Haarnoja et al. [2018b] is not used for this case, instead a fixed temperature parameter Haarnoja et al. [2018a] of 0.2 is used.

[11] While potential performance gains are possible tuning $T_{\max}$ and ζ, our focus is to underscore the ease of tuning MACURA.

Table 7: Hyperparameters M2AC (Stochastic Environment Policy)

Environment	Humanoid	Ant	Halfcheetah	Walker	Hopper
Epochs	300	300	400	200	125
Steps per Epoch	1000				
PNNs per PE	7				
PNN Layers	4				
PNN Nodes per Layer	200				
Critic Layers	3				
Critic Nodes per Layer	1024				
Actor Layers	3				
Actor Nodes per Layer	1024				
SAC Target Ent.	-17	-1	-4	N/A	
SAC Updates G	20		10		30
Rollouts per Round M	400		200	400	
Rollouts length $T_{\max}$	10				

Table 8: Hyperparameters SAC

Environment	Humanoid	Ant	Halfcheetah	Walker	Hopper
Critic Layers	3				
Critic Nodes per Layer	256				
Actor Layers	3				
Actor Nodes per Layer	256				
SAC Target Ent.	-17	N/A	-6	-6	-3
SAC Updates G	1				

However, that low values of ξ also destabilize training is more surprising. We assume that the on-policy nature of Dyna-style MBRL, discussed in Section 2.3, in combination with the effective rollout adaption mechanism of MACURA leads to narrow data distributions in $\mathcal{D}_{\text{mod}}$ when restricting model uncertainty too harshly. Consequently, the Q-functions of the model-free agent overfit, which destabilizes learning. A practical guide on tuning ξ is provided in Appendix D.3.

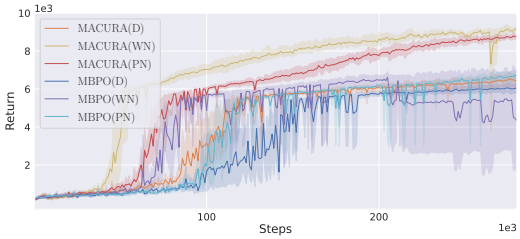
D.3 Discussion ξ

The parameter ξ allows us to tune what *sufficiently* certain is. The general idea is that in the first rollout step, the model is evaluated in states that are in $\mathcal{D}_{\text{env}}$ and have been seen before. Thus, the uncertainty values we get for this first step are values of the GJS that correspond to being *certain*. So we choose the base uncertainty $\hat{u}_{\text{GJS}}$ to be the $\zeta = 0.95$ quantile of uncertainties after this first rollout step. Taking this base uncertainty as the uncertainty threshold, which corresponds to $\xi = 1$, yields reasonable results in all environments. In this case, whenever a prediction step overshoots the 0.95 quantile of uncertainty within the data support, we consider it to have left $\mathcal{E}$ and terminate the model-based rollout.

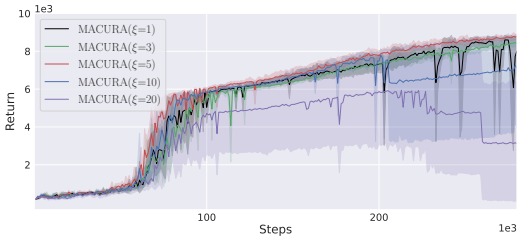
However, the model is often very certain within the data support towards the end of the training, leading to a low uncertainty threshold κ resulting from $\xi = 1$. This leads to a narrow data distribution in $\mathcal{D}_{\text{mod}}$. Repeatedly training the critics on this narrow distribution causes overfitting. This introduces instability towards the end of training as discussed in Section D.2 and shown in Figure 7b. Thus, we typically choose $\xi > 1$. This is true for all MuJoCo environments but Walker. The Walker dynamics seem hard to learn for the PE model. This can e.g. be seen from the learning behavior of MBPO in Figure 3. Even though MBPO solely performs model-based rollouts of length 1 throughout training, learning appears rather brittle. Thus, in the Walker task, the 0.95 quantile of uncertainties after the first rollout step is too uncertain for stable learning. Therefore, we chose $\xi < 1$ to stabilize learning in this case.

From our experience, ξ can be tuned in the following way:

- Perform a run with $\xi = 1$. This should result in stable learning as long as the agent is in the early stages of training and improves sufficiently fast. As soon as the algorithm is close to its



(a) Exploration Schemes on MACURA and MBPO. *Impact of deterministic (D), white noise (WN), and pink noise (PN) exploration on algorithmic performance.*



(b) Tuning scaling parameter ξ. *MACURA performs well for intermediate ξ. Too large values of ξ lead to model exploitation, too small values lead to overfitting of the agent.*

asymptotic performance, the uncertainty threshold might get too low such that instabilities in learning occur.

- If this is the case, increase ξ. As indicated in Figure 7b, there is a relatively broad range of ξ values that yields stable performance. If ξ is chosen too large, however, instabilities due to model exploitation occur.
- If instabilities occur early in training, when running MACURA with $\xi = 1$ the model probably approximates the true dynamics poorly. This can be checked e.g. by reproducing trajectories in $\mathcal{D}_{\text{env}}$ through model-based rollouts and considering the deviation. In this case, ξ can be reduced until the learning behavior is stable.

The fact that the entire rollout mechanism can be tuned by choosing ξ, from our experience, makes tuning considerably easier than determining all the required hyperparameters of the M2AC mechanism or designing a suitable rollout schedule for MBPO.

D.4 The Importance of Environment Exploration

Following the state-of-the-art implementation Pineda et al. [2021], the agent interacts deterministically with the environment in MBPO and M2AC, while we choose pink noise exploration for MACURA in Figure 3. The impact of exploration on MACURA and MBPO is illustrated in Figure 7a for the Humanoid task. We observe a general trend that is discussed in more detail in Appendix D.5. For both algorithms, deterministic interaction yields limited yet stable performance. Classic white noise exploration, where actions are sampled independently, has the potential of strong performance, as can be seen in MACURA, while introducing the risk of destabilizing learning, as in the case of MBPO. Pink noise exploration, instead, shows an intermediate, overall best behavior, combining high performance with stable learning. The strongest gains from environment interaction can be observed for MACURA, underscoring the effectiveness of the notion of $\mathcal{E}$ for model usage.

D.5 Exploration in the Environment

The implementation used in this work, based on the mbrl-lib library Pineda et al. [2021], uses deterministic environment exploration for MBPO yielding similar results to the original implementation Janner et al. [2019]. As far as we can reconstruct, the original implementation Janner et al. [2019] uses white noise exploration in the agent environment interaction, however, when implementing white noise exploration in the mbrl-lib code, we observe substantially different results from those reported in Janner et al. [2019]. These range from better performance to divergence. For our reimplementation of M2AC based on the mbrl-lib code, deterministic environment interaction also yields the most stable results. Thus, we present the performance of deterministic environment interaction for MBPO and M2AC in Figure 3.

In Figures 8 - 12 we present the performance of MACURA, MBPO, and M2AC with the considered exploration schemes: deterministic interaction, white exploration noise and pink noise exploration Eberhard et al. [2023], on the MuJoCo benchmark.

We have issues producing stable results for M2AC, despite putting the most tuning effort among the compared approaches. We find that hyperparameter settings that are stable for deterministic interaction destabilize in the case of exploration noise and vice versa. Thus we provide a set

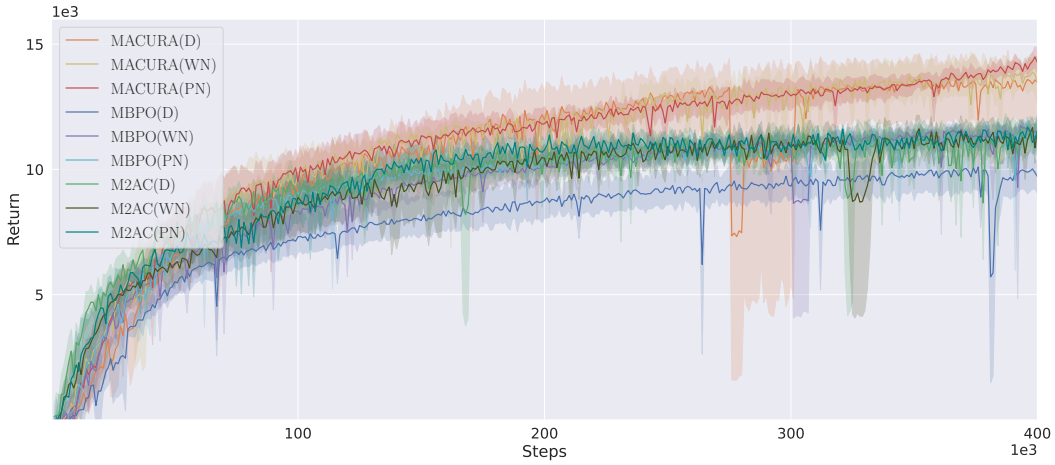


Figure 8: Exploration Schemes on MACURA, MBPO, and M2AC on Halfcheetah. *Impact of deterministic (D), white noise (WN), and pink noise (PN) exploration on algorithmic performance.*

of hyperparameters for deterministic environment interaction and one that we use for white and pink noise exploration. For MACURA and MBPO we run the same sets of hyperparameters for all exploration schemes. Generally, we see the performance of M2AC fall short as compared to MACURA and MBPO.

For MBPO, we use the fine-tuned rollout schemes reported in Janner et al. [2019]. White noise exploration in most cases positively impacts asymptotic performance but increases the variance among runs. In some instances, we observe that white noise exploration destabilizes learning in MBPO, leading to divergence. For pink noise exploration, we can report a generally positive impact on the performance of MBPO, increasing data efficiency and asymptotic performance without destabilizing learning. In some environments, MBPO with a fine-tuned rollout schedule and pink noise exploration can even compete with MACURA.

MACURA with deterministic environment interaction shows the strongest performance among deterministic interaction methods in most environments. White noise exploration in some cases yields better results than pink noise exploration, while considerably increasing variance among trained agents. Different from MBPO, however, white noise exploration does not destabilize MACURA up to the point of divergence. Overall, we observe that MACURA with pink noise exploration yields the best data efficiency, asymptotic performance, and stability among all approaches on the benchmark. Combined with a considerably easier-to-tune rollout length scheduling mechanism than MBPO and M2AC we consider this the most promising approach for Dyna-style MBRL.

D.6 Noisy Environment

We evaluate MBPO, M2AC, and MACURA on a noisy version of the Halfcheetah environment. To introduce noise, we take the approach proposed in Pan et al. [2020] and add noise to the actions applied to the environment. Therefore, the action applied to the environment is $\tilde{a}_t = a_t + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \Sigma)$. The covariance matrix Σ is a diagonal matrix with diagonal elements σ^2. The additive noise is not observable to the agent and therefore introduces process noise. Following the experimental setup in Pan et al. [2020], we conduct three experiments with $\sigma = 0.05$, $\sigma = 0.1$, and $\sigma = 0.2$ respectively. The corresponding results are depicted in Figure 13. MACURA consistently outperforms MBPO and M2AC.

D.7 Rollout Horizon, Data Uncertainty, Gradient Steps

We evaluate the rollout - and gradient step adaption mechanism of MACURA on the Walker environment. We compare MACURA to M2AC and MBPO.

First, we consider the average rollout length of the respective approaches throughout training as depicted in Figure 14. Therefore, we record the rollout lengths of all M model-based rollouts

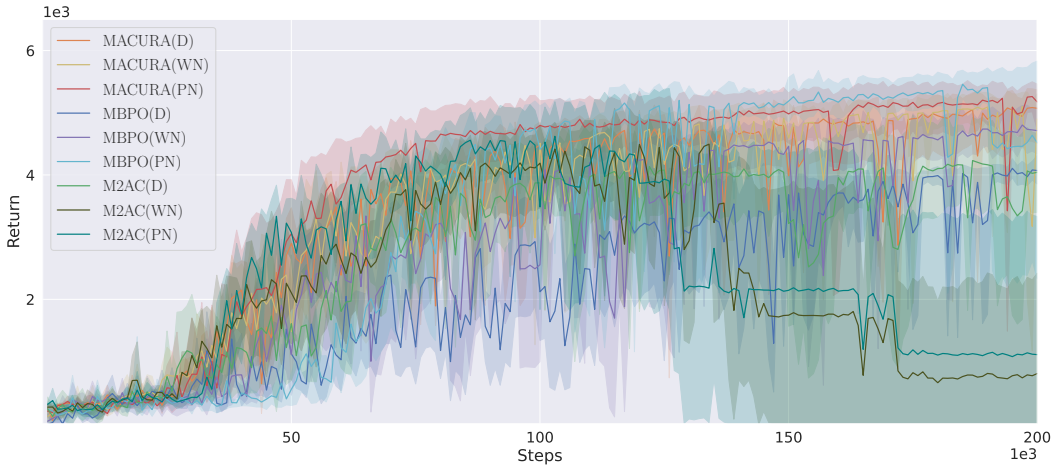


Figure 9: Exploration Schemes on MACURA, MBPO, and M2AC on Walker. *Impact of deterministic (D), white noise (WN), and pink noise (PN) exploration on algorithmic performance.*

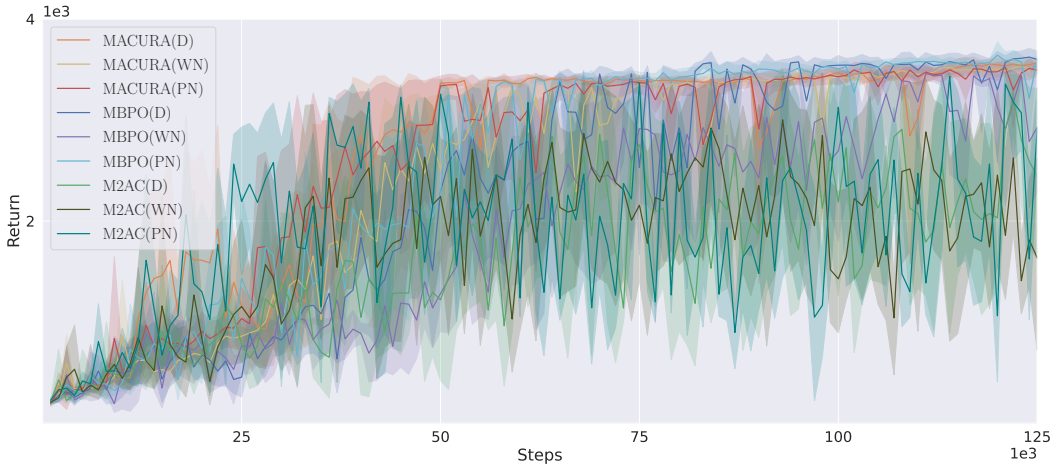


Figure 10: Exploration Schemes on MACURA, MBPO, and M2AC on Hopper. *Impact of deterministic (D), white noise (WN), and pink noise (PN) exploration on algorithmic performance.*

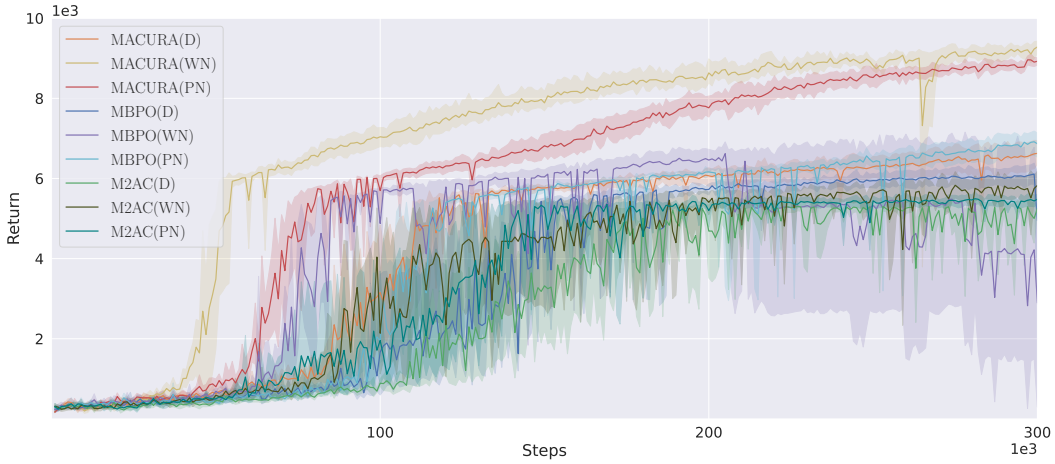


Figure 11: Exploration Schemes on MACURA, MBPO, and M2AC on Humanoid. *Impact of deterministic (D), white noise (WN), and pink noise (PN) exploration on algorithmic performance.*

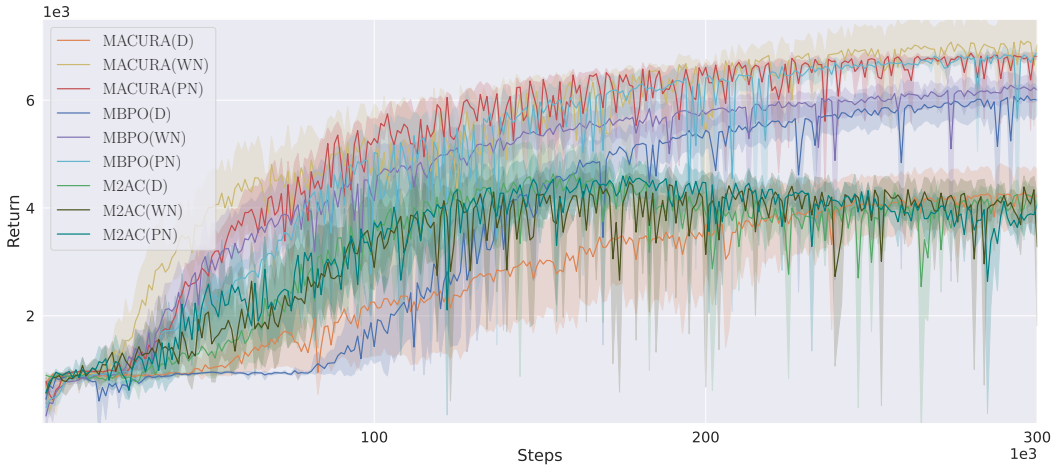


Figure 12: Exploration Schemes on MACURA, MBPO, and M2AC on Ant. *Impact of deterministic (D), white noise (WN), and pink noise (PN) exploration on algorithmic performance.*

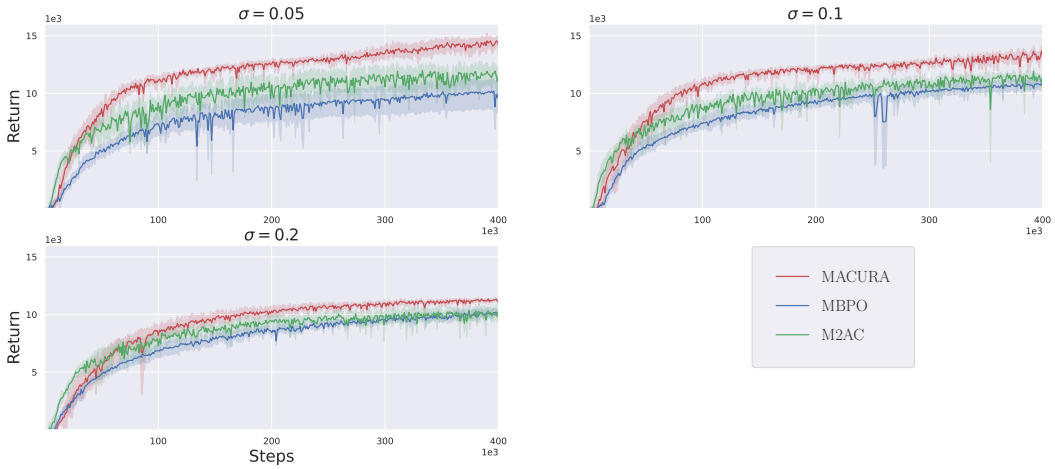


Figure 13: Returns obtained by MACURA, MBPO, and M2AC on noisy Halfcheetah.

performed in parallel during one round of model-based rollouts and compute their average. As proposed in Janner et al. [2019], MBPO performs one-step rollouts throughout training leading to an average rollout length of one. The rank-based filtering heuristic of M2AC terminates a fixed quantile of model-based rollouts after each rollout step. Effectively, this yields a predefined distribution over rollout lengths, which does not change throughout training. As a side effect, this leads to a constant average rollout length as depicted in Figure 14. MACURA, instead, has a threshold-based rollout length adaption mechanism, thus the average rollout length varies throughout training. MACURA performs short rollouts in the early stages of training, where the model has limited capabilities and increases the rollout length as the model improves. Different from MBPO and M2AC, MACURA can even discard the first rollout step enabling an average rollout length lower than one.

Next, we investigate the uncertainty of the corresponding data created in model-based rollouts. We measure uncertainty with the variance of the mean predictions of the individual PNNs within the PE. To recover a scalar uncertainty, we take the Frobenius norm of the resulting matrix:

$$u_{\mathrm{MV}}(s,a) = \left\| \frac{1}{E} \sum_{e=1}^{E} (\mu_{\theta_e}(s,a) - \mu_{\mathrm{PE}}(s,a))(\mu_{\theta_e}(s,a) - \mu_{\mathrm{PE}}(s,a))^{\top} \right\|_{\mathrm{F}} \tag{32}$$

Figure 14: Average rollout lengths for MACURA, MBPO, and M2AC on Walker.

with $\mu_{\mathrm{PE}}(s,a) = \frac{1}{E}\sum_{e=1}^{E}\mu_{\theta_e}(s,a)$. By doing this, we aim to avoid artifacts introduced by comparing uncertainty in the metrics used in M2AC or MACURA and instead provide an impartial analysis of how uncertainty evolves in model-based rollouts.

The average uncertainty corresponding to the respective Dyna-style approaches is depicted in Figure 15. MBPO has the highest average uncertainty that reduces throughout training. M2AC shows lower average uncertainty than MBPO. We assume this is the case, as low-uncertainty rollouts are propagated for several rollout steps reducing the mean uncertainty. MACURA shows an average uncertainty comparable to M2AC. However, the initial average uncertainty in MACURA is substantially lower than in M2AC and MBPO due to the threshold-based rollout adaption mechanism. Different from the rank-based heuristic of M2AC that always takes a particular most certain percentage into account (which can include very uncertain data when initially most data is bad), MACURA can discard all the data that is above the uncertainty threshold.

It should be noted that the average uncertainty gives a rough impression of data quality but is not a suitable metric for detecting low-quality outliers that occur with low probability. These outliers, however, have a strong influence on the learning process from our experience.

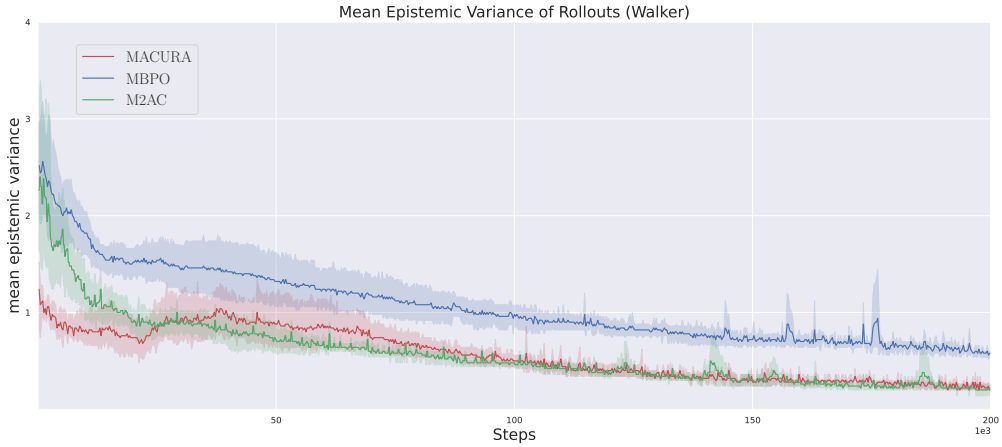


Figure 15: Mean epistemic uncertainty of rollouts for MACURA, MBPO, and M2AC on Walker.

The varying amounts of data created in model-based rollouts require adapting the amount of SAC updates in MACURA. Figure 16 shows the average gradient steps G of the respective approaches. Both MBPO and M2AC have a $G = 30$ fixed throughout training, while MACURA adapts G depending on the occupancy of $\mathcal{D}_{\mathrm{mod}}$ (15) that depends on the average rollout length depicted in Figure 14. Therefore, MACURA performs fewer SAC updates in the early stages of training but

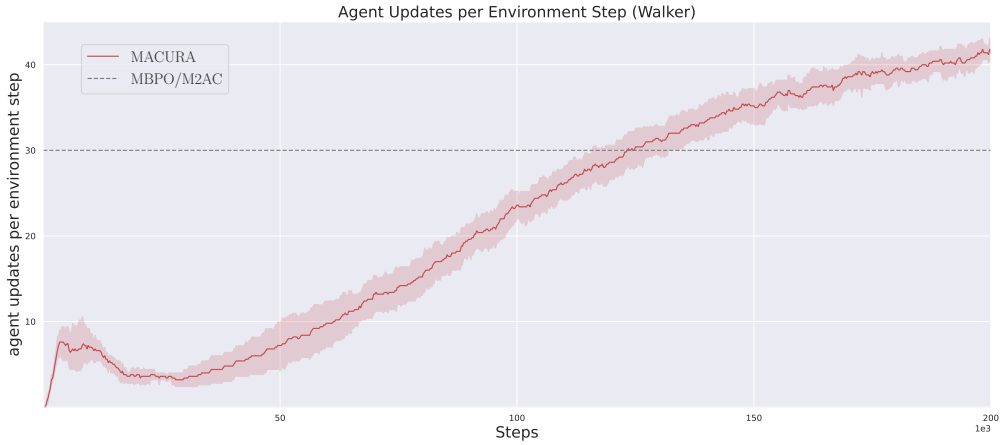


Figure 16: SAC Update Steps

catches up as the model improves. Towards the end of training, MACURA performs a bit over 40 SAC updates per timestep. Simply increasing G to 40 in MBPO and M2AC, however, does not yield stable results, as depicted in Figure 17 (in the figure G is referred to as update-to-data (UTD) ratio).

Figure 17: Comparison of returns obtained on Walker.

D.8 Uncertainty Estimates

In the following, we investigate the performance of MACURA and M2AC with different uncertainty estimates. Results on the Humanoid environment are depicted in Figure 18.

M2AC with the GJS uncertainty estimate learns slightly faster and shows slightly stronger performance with less variance. We assume this to be the case, as the GJS estimate is more reliable in detecting low-quality data, thus stabilizing learning.

MACURA with OvR uncertainty estimate Pan et al. [2020] performs well up to a return of about 7000 and subsequently diverges presumably due to the brittleness of the OvR uncertainty estimate.

Additionally, we conduct experiments with MACURA using the ensemble variance estimate proposed in Lu et al. [2021] and mean variance (32).

While MACURA works with both these uncertainty metrics, they yield weaker results than the GJS estimate. This aligns with our intuition, as ensemble variance is a combined estimate for epistemic and aleatoric uncertainty, while we are solely interested in epistemic uncertainty. The variance of ensemble means solely addresses epistemic variance but has no information about the

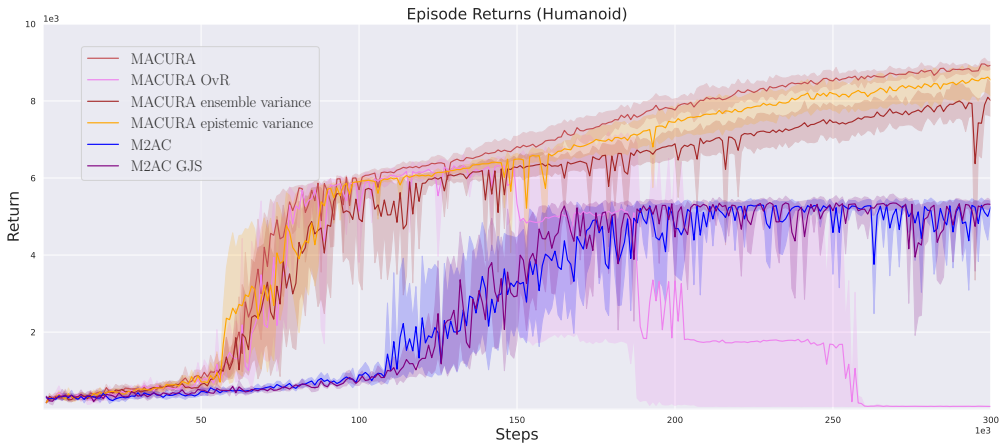


Figure 18: MACURA and M2AC with different uncertainty metrics on Humanoid.

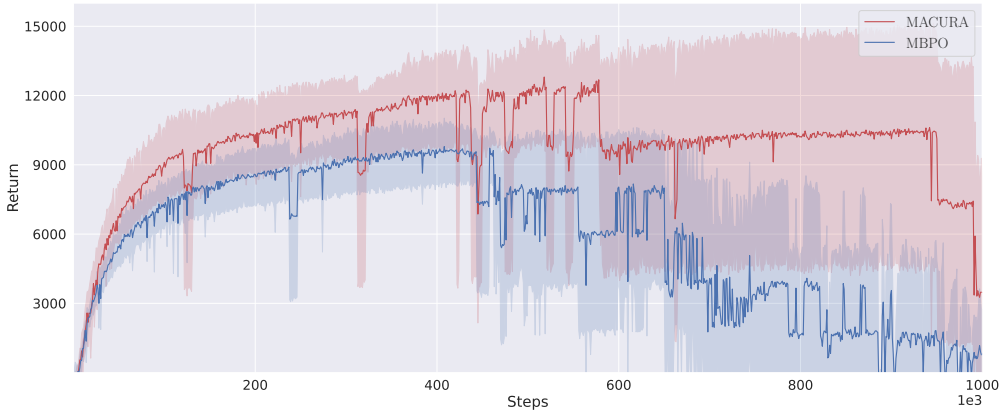


Figure 19: MACURA and MBPO with prolonged experiment length on Halfcheetah.

aleatoric uncertainty at all, which can be problematic. Epistemic uncertainty, however, is defined by the disagreement of individual PNN predictions, which rather corresponds to the overlap in the probability mass of the individual predictions. The GJS uncertainty estimate measures this overlap in probability mass, thus providing a better metric for epistemic uncertainty and works better in practice.

D.9 Prolonged Experiments

In the following, we discuss the behavior of MBPO and MACURA, when trained beyond the typically reported length of experiments Janner et al. [2019], Pan et al. [2020]. Figure 19 depicts results on Halfcheetah for 1,000,000 environment interactions as opposed to the 400,000 environment interactions presented in Figure 3. We do not consider M2AC in these experiments as stabilizing M2AC even for the first 400,000 environment interactions is a substantial challenge. Both MBPO and MACURA destabilize after an extended amount of training. From our experience, this is due to overfitting of either the model or the critic after training on similar data distributions over and over again. The training heuristics for model and critic of MBPO that are largely inherited by MACURA are not well suited for continuing training. Extending these algorithms to a continuing training setting requires further research addressing *when to train your model and agent* that is beyond the scope of this work.