Model-Based Reinforcement Learning via Meta-Policy Optimization

Ignasi Clavera*
UC Berkeley
iclavera@berkeley.edu

Jonas Rothfuss*
KIT, UC Berkeley
jonas.rothfuss@kit.edu

John Schulman
OpenAI

Yasuhiro Fujita
Preferred Networks

Tamim Asfour
Karlsruhe Inst. of Technology (KIT)

Pieter Abbeel
UC Berkeley, Covariant.AI

Abstract: Model-based reinforcement learning approaches carry the promise of being data efficient. However, due to challenges in learning dynamics models that sufficiently match the real-world dynamics, they struggle to achieve the same asymptotic performance as model-free methods. We propose Model-Based Meta-Policy-Optimization (MB-MPO), an approach that foregoes the strong reliance on accurate learned dynamics models. Using an ensemble of learned dynamic models, MB-MPO meta-learns a policy that can quickly adapt to any model in the ensemble with one policy gradient step. This steers the meta-policy towards internalizing consistent dynamics predictions among the ensemble while shifting the burden of behaving optimally w.r.t. the model discrepancies towards the adaptation step. Our experiments show that MB-MPO is more robust to model imperfections than previous model-based approaches. Finally, we demonstrate that our approach is able to match the asymptotic performance of model-free methods while requiring significantly less experience.

Keywords: Reinforcement Learning, Meta-Learning, Model-Based, Model-Free

1 Introduction

Most of the recent success in reinforcement learning was achieved using model-free reinforcement learning algorithms [1, 2, 3]. Model-free (MF) algorithms tend to achieve optimal performance, are generally applicable, and are easy to implement. However, this is achieved at the cost of being data intensive, which is exacerbated when combined with high-capacity function approximators like neural networks. Their high sample complexity presents a major barrier to their application to robotic control tasks, on which data gathering is expensive.

In contrast, model-based (MB) reinforcement learning methods are able to learn with significantly fewer samples by using a learned model of the environment dynamics against which policy optimization is performed. Learning dynamics models can be done in a sample efficient way since they are trained with standard supervised learning techniques, allowing the use of off-policy data. However, accurate dynamics models can often be far more complex than good policies. For instance, pouring water into a cup can be achieved by a fairly simple policy while modeling the underlying dynamics of this task is highly complex. Hence, model-based methods have only been able to learn good policies on a much more limited set of problems, and even when good policies are learned, they typically saturate in performance at a level well below their model-free counterparts [4, 5].

Model-based approaches tend to rely on accurate (learned) dynamics models to solve a task. If the dynamics model is not sufficiently precise, the policy optimization is prone to overfit on the deficiencies of the model, leading to suboptimal behavior or even to catastrophic failures. This problem is known in the literature as model-bias [6]. Previous work has tried to alleviate model-bias by characterizing the uncertainty of the models and learning a robust policy [6, 7, 8, 9, 10], often using ensembles to represent the posterior. This paper also uses ensembles, but very differently.

We propose Model-Based Meta-Policy-Optimization (MB-MPO), an orthogonal approach to previous model-based RL methods: while traditional model-based RL methods rely on the learned

*Equal contribution

2nd Conference on Robot Learning (CoRL 2018), Zürich, Switzerland.

dynamics models to be sufficiently accurate to enable learning a policy that also succeeds in the real world, we forego reliance on such accuracy. We are able to do so by learning an ensemble of dynamics models and framing the policy optimization step as a meta-learning problem. Meta-learning, in the context of RL, aims to learn a policy that adapts fast to new tasks or environments [11, 12, 13, 14, 15]. Using the models as learned simulators, MB-MPO learns a policy that can be quickly adapted to any of the fitted dynamics models with one gradient step. This optimization objective steers the meta-policy towards internalizing the parts of the dynamics prediction that are consistent among the ensemble while shifting the burden of behaving optimally w.r.t discrepancies between models towards the adaptation step. This way, the learned policy exhibits less model-bias without the need to behave conservatively. While much is shared with previous MB methods in terms of how trajectory samples are collected and the dynamic models are trained, the use of (and reliance on) learned dynamics models for the policy optimization is fundamentally different.

In this paper we show that 1) model-based policy optimization can learn policies that match the asymptotic performance of model-free methods while being substantially more sample efficient, 2) MB-MPO consistently outperforms previous model-based methods on challenging control tasks, 3) learning is still possible when the models are strongly biased. The low sample complexity of our method makes it applicable to real-world robotics. For instance, we are able learn an optimal policy in high-dimensional and complex quadrupedal locomotion within two hours of real-world data. Note that the amount of data required to learn such policy using model-free methods is $10\times$ - $100\times$ higher, and, to the best knowledge of the authors, no prior model-based method has been able to attain the model-free performance in such tasks.

2 Related Work

In this section, we discuss related work, including model-based RL and approaches that combine elements of model-based and model-free RL. Finally, we outline recent advances in the field of meta-learning.

Model-Based Reinforcement Learning: Addressing Model Inaccuracies. Impressive results with model-based RL have been obtained using simple linear models [16, 17, 18, 19]. However, like Bayesian models [6, 20, 21], their application is limited to low-dimensional domains. Our approach, which uses neural networks (NNs), is easily able to scale to complex high dimensional control problems. NNs for model learning offer the potential to scale to higher dimensional problems with impressive sample complexity [22, 23, 24, 25]. A major challenge when using high-capacity dynamics models is preventing policies from exploiting model inaccuracies. Several works approach this problem of model-bias by learning a distribution of models [26, 7, 10, 23], or by learning adaptive models [27, 28, 29]. We incorporate the idea of reducing model-bias by learning an ensemble of models. However, we show that these techniques do not suffice in challenging domains, and demonstrate the necessity of meta-learning for improving asymptotic performance.

Past work has also tried to overcome model inaccuracies through the policy optimization process. Model Predictive Control (MPC) compensates for model imperfections by re-planning at each step [30], but it suffers from limited credit assignment and high computational cost. Robust policy optimization [7, 8, 9] looks for a policy that performs well across models; as a result policies tend to be over-conservative. In contrast, we show that MB-MPO learns a robust policy in the regions where the models agree, and an adaptive one where the models yield substantially different predictions.

Model-Based + Model-Free Reinforcement Learning. Naturally, it is desirable to combine elements of model-based and model-free to attain high performance with low sample complexity. Attempts to combine them can be broadly categorized into three main approaches. First, differentiable trajectory optimization methods propagate the gradients of the policy or value function through the learned dynamics model [31, 32] . However, the models are not explicitly trained to approximate first order derivatives, and, when backpropagating, they suffer from exploding and vanishing gradients [10]. Second, model-assisted MF approaches use the dynamics models to augment the real environment data by imagining policy roll-outs [33, 29, 34, 22]. These methods still rely to a large degree on real-world data, which makes them impractical for real-world applications. Thanks to meta-learning, our approach could, if needed, adapt fast to the real-world with fewer samples. Third, recent work fully decouples the MF module from the real environment by entirely using samples from the learned models [35, 10]. These methods, even though considering the model uncertainty, still rely on precise estimates of the dynamics to learn the policy. In contrast, we meta-

learn a policy on an ensemble of models, which alleviates the strong reliance on precise models by training for adaption when the prediction uncertainty is high. Kurutach et al. [10] can be viewed as an edge case of our algorithm when no adaptation is performed.

Meta-Learning. Our approach makes use of meta-learning to address model inaccuracies. Meta-learning algorithms aim to learn models that can adapt to new scenarios or tasks with few data points. Current meta-learning algorithms can be classified in three categories. One approach involves training a recurrent or memory-augmented network that ingests a training dataset and outputs the parameters of a learner model [36, 37]. Another set of methods feeds the dataset followed by the test data into a recurrent model that outputs the predictions for the test inputs [12, 38]. The last category embeds the structure of optimization problems into the meta-learning algorithm [11, 39, 40]. These algorithms have been extended to the context of RL [12, 13, 15, 11]. Our work builds upon MAML [11]. However, while in previous meta-learning methods each task is typically defined by a different reward function, each of our tasks is defined by the dynamics of different learned models.

3 Background

3.1 Model-based Reinforcement Learning

A discrete-time finite Markov decision process (MDP) $\mathcal{M}$ is defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma, p_0, H)$. Here, $\mathcal{S}$ is the set of states, $\mathcal{A}$ the action space, $p(\mathbf{s_{t+1}}|\mathbf{s_t}, \mathbf{a_t})$ the transition distribution, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function, $p_0 : \mathcal{S} \to \mathbb{R}_+$ represents the initial state distribution, γ the discount factor, and H is the horizon of the process. We define the return as the sum of rewards $r(\boldsymbol{s}_t, \boldsymbol{a}_t)$ along a trajectory $\tau := (\boldsymbol{s}_0, \boldsymbol{a}_0, ..., \boldsymbol{s}_{H-1}, \boldsymbol{a}_{H-1}, \boldsymbol{s}_H)$. The goal of reinforcement learning is to find a policy $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^+$ that maximizes the expected return.

While model-free RL does not explicitly model state transitions, model-based RL methods learn the transition distribution, also known as dynamics model, from the observed transitions. This can be done with a parametric function approximator $\hat{p}_{\boldsymbol{\phi}}(\mathbf{s}'|\mathbf{s}, \mathbf{a})$. In such case, the parameters $\boldsymbol{\phi}$ of the dynamics model are optimized to maximize the log-likelihood of the state transition distribution.

3.2 Meta-Learning for Reinforcement Learning

Meta-RL aims to learn a learning algorithm which is able to quickly learn optimal policies in MDPs $\mathcal{M}_k$ drawn from a distribution $\rho(\mathcal{M})$ over a set of MDPs. The MDPs $\mathcal{M}_k$ may differ in their reward function $r_k(\boldsymbol{s}, \boldsymbol{a})$ and transition distribution $p_k(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$, but share action space $\mathcal{A}$ and state space $\mathcal{S}$.

Our approach builds on the gradient-based meta-learning framework MAML [11], which in the RL setting, trains a parametric policy $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})$ to quickly improve its performance on a new task with one or a few vanilla policy gradient steps. The meta-training objective for MAML can be written as:

$$\max_{\boldsymbol{\theta}} \; \mathbb{E}_{\substack{\mathcal{M}_k \sim \rho(\mathcal{M}) \\ \boldsymbol{s}_{t+1} \sim p_k \\ \boldsymbol{a}_t \sim \pi_{\boldsymbol{\theta}'}(\boldsymbol{a}_t|\boldsymbol{s}_t)}} \left[\sum_{t=0}^{H-1} r_k(\boldsymbol{s}_t, \boldsymbol{a}_t) \right] \quad \text{s.t.: } \boldsymbol{\theta}' = \boldsymbol{\theta} + \alpha \, \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\substack{\boldsymbol{s}_{t+1} \sim p_k \\ \boldsymbol{a}_t \sim \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t|\boldsymbol{s}_t)}} \left[\sum_{t=0}^{H-1} r_k(\boldsymbol{s}_t, \boldsymbol{a}_t) \right] \quad (1)$$

MAML attempts to learn an initialization $\boldsymbol{\theta}^*$ such that for any task $\mathcal{M}_k \sim \rho(\mathcal{M})$ the policy attains maximum performance in the respective task after one policy gradient step.

4 Model-Based Meta-Policy-Optimization

Enabling complex and high-dimensional real robotics tasks requires extending current model-based methods to the capabilities of mode-free while, at the same time, maintaining their data efficiency. Our approach, model-based meta-policy-optimization (MB-MPO), attains such goal by framing model-based RL as meta-learning a policy on a distribution of dynamic models, advocating to maximize the policy adaptation, instead of robustness, when models disagree. This not only removes the arduous task of optimizing for a single policy that performs well across differing dynamic models, but also results in better exploration properties and higher diversity of the collected samples, which leads to improved dynamic estimates.

We instantiate this general framework by employing an ensemble of learned dynamic models and meta-learning a policy that can be quickly adapted to any of the dynamic models with one policy gradient step. In the following, we first describe how the models are learned, then explain how the policy can be meta-trained on an ensemble of models, and, finally, we present our overall algorithm.

4.1 Model Learning

A key component of our method is learning a distribution of dynamics models, in the form of an ensemble, of the real environment dynamics. In order to decorrelate the models, each model differs in its random initialization and it is trained with a different randomly selected subset $\mathcal{D}_k$ of the collected real environment samples. In order to address the distributional shift that occurs as the policy changes throughout the meta-optimization, we frequently collect samples under the current policy, aggregate them with the previous data $\mathcal{D}$, and retrain the dynamic models with warm starts.

In our experiments, we consider the dynamics models to be a deterministic function of the current state $\boldsymbol{s}_t$ and action $\boldsymbol{a}_t$, employing a feed-forward neural network to approximate them. We follow the standard practice in model-based RL of training the neural network to predict the change in state $\Delta \boldsymbol{s} = \boldsymbol{s}_{t+1} - \boldsymbol{s}_t$ (rather than the next state $\boldsymbol{s}_{t+1}$) [22, 6]. We denote by $\hat{f}_{\boldsymbol{\phi}}$ the function approximator for the next state, which is the sum of the input state and the output of the neural network. The objective for learning each model $\hat{f}_{\boldsymbol{\phi}_k}$ of the ensemble is to find the parameter vector $\boldsymbol{\phi}_k$ that minimizes the ℓ_2 one-step prediction loss:

$$\min_{\boldsymbol{\phi}_k} \frac{1}{|\mathcal{D}_k|} \sum_{(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}) \in \mathcal{D}_k} \| \boldsymbol{s}_{t+1} - \hat{f}_{\boldsymbol{\phi}_k}(\boldsymbol{s}_t, \boldsymbol{a}_t) \|_2^2 \tag{2}$$

where $\mathcal{D}_k$ is a sampled subset of the training data-set $\mathcal{D}$ that stores the transitions which the agent has experienced. Standard techniques to avoid overfitting and facilitate fast learning are followed; specifically, 1) early stopping the training based on the validation loss, 2) normalizing the inputs and outputs of the neural network, and 3) weight normalization [41].

4.2 Meta-Reinforcement Learning on Learned Models

Given an ensemble of learned dynamic models for a particular environment, our core idea is to learn a policy which can adapt quickly to any of these models. To learn this policy, we use gradient based meta-learning with MAML (described in Section 3.2). To properly formulate this problem in the context of meta-learning, we first need to define an appropriate task distribution. Considering the models $\{\hat{f}_{\boldsymbol{\phi}_1}, \hat{f}_{\boldsymbol{\phi}_2}, ..., \hat{f}_{\boldsymbol{\phi}_K}\}$, which approximate the dynamics of the true environment, we can construct a uniform task distribution by embedding them into different MDPs $\mathcal{M}_k = (S, A, \hat{f}_{\boldsymbol{\phi}_k}, r, \gamma, p_0)$ using these learned dynamics models. We note that, unlike the experimental considerations of prior methods [12, 11, 14], in our work the reward function remains the same across tasks while the dynamics vary. Therefore, each task constitutes a different belief about what the dynamics in the true environment could be. Finally, we pose our objective as the following meta-optimization problem:

$$\max_{\boldsymbol{\theta}} \quad \frac{1}{K} \sum_{k=0}^{K} J_k(\boldsymbol{\theta}'_k) \qquad \text{s.t.:} \quad \boldsymbol{\theta}'_k = \boldsymbol{\theta} + \alpha \, \nabla_{\boldsymbol{\theta}} J_k(\boldsymbol{\theta}) \tag{3}$$

with $J_k(\boldsymbol{\theta})$ being the expected return under the policy $\pi_{\boldsymbol{\theta}}$ and the estimated dynamics model $\hat{f}_{\boldsymbol{\phi}_k}$.

$$J_k(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{a}_t \sim \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t | \boldsymbol{s}_t)} \left[\sum_{t=0}^{H-1} r(\boldsymbol{s}_t, \boldsymbol{a}_t) \middle| \boldsymbol{s}_{t+1} = \hat{f}_{\boldsymbol{\phi}_k}(\boldsymbol{s}_t, \boldsymbol{a}_t) \right] \tag{4}$$

For estimating the expectation in Eq. 4 and computing the corresponding gradients, we sample trajectories from the imagined MDPs. The rewards are computed by evaluating the reward function, which we assume as given, in the predicted states and actions $r(\hat{f}_{\boldsymbol{\phi}_k}(\boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1}, \boldsymbol{a}_t))$. In particular, when estimating the adaptation objectives $J_k(\boldsymbol{\theta})$, the meta-policy $\pi_{\boldsymbol{\theta}}$ is used to sample a set of imaginary trajectories $\mathcal{T}_k$ for each model $\hat{f}_{\boldsymbol{\phi}_k}$. For the meta-objective $\frac{1}{K} \sum_{k=0}^{K} J_k(\boldsymbol{\theta}'_k)$, we generate trajectory roll-outs $\mathcal{T}'_k$ with the models $\hat{f}_{\boldsymbol{\phi}_k}$ and the policies $\pi_{\boldsymbol{\theta}'_k}$ obtained from adapting the parameters $\boldsymbol{\theta}$ to the k-th model. Thus, no real-world data is used for the data intensive step of meta-policy optimization.

In practice, any policy gradient algorithm can be chosen to perform the meta-update of the policy parameters. In our implementation, we use Trust-Region Policy Optimization (TPRO) [1] for maximizing the meta-objective, and employ vanilla policy gradient (VPG) [42] for the adaptation step. To reduce the variance of the policy gradient estimates a linear reward baseline is used.

Algorithm 1 MB-MPO

Require: Inner and outer step size α, β

Initialize the policy $\pi_{\boldsymbol{\theta}}$, the models $\hat{f}_{\boldsymbol{\phi}_1}, \hat{f}_{\boldsymbol{\phi}_2}, ..., \hat{f}_{\boldsymbol{\phi}_K}$ and $\mathcal{D} \leftarrow \emptyset$
repeat
  Sample trajectories from the real environment with the adapted policies $\pi_{\boldsymbol{\theta}'_1}, ..., \pi_{\boldsymbol{\theta}'_K}$. Add them to $\mathcal{D}$.
  Train all models using $\mathcal{D}$.
  for all models $\hat{f}_{\boldsymbol{\phi}_k}$ **do**
    Sample imaginary trajectories $\mathcal{T}_k$ from $\hat{f}_{\boldsymbol{\phi}_k}$ using $\pi_{\boldsymbol{\theta}}$
    Compute adapted parameters $\boldsymbol{\theta}'_k = \boldsymbol{\theta} + \alpha \nabla_{\boldsymbol{\theta}} J_k(\boldsymbol{\theta})$ using trajectories $\mathcal{T}_k$
    Sample imaginary trajectories $\mathcal{T}'_k$ from $\hat{f}_{\boldsymbol{\phi}_k}$ using the adapted policy $\pi_{\boldsymbol{\theta}'_k}$
  end for
  Update $\boldsymbol{\theta} \rightarrow \boldsymbol{\theta} - \beta \frac{1}{K} \sum_k \nabla_{\boldsymbol{\theta}} J_k(\boldsymbol{\theta}'_k)$ using the trajectories $\mathcal{T}'_k$
until the policy performs well in the real environment
return Optimal pre-update parameters $\boldsymbol{\theta}^*$

4.3 Algorithm

In the following, we describe the overall algorithm of our approach (see Algorithm 1). First, we initialize the models and the policy with different random weights. Then, we proceed to the data collection step. In the first iteration, a uniform random controller is used to collect data from the real-world, which is stored in a buffer $\mathcal{D}$. At subsequent iterations, trajectories from the real-world are collected with the adapted policies $\{\pi_{\boldsymbol{\theta}'_1}, ..., \pi_{\boldsymbol{\theta}'_K}\}$, and then aggregated with the trajectories from previous iterations. The models are trained with the aggregated real-environment samples following the procedure explained in section 4.1. The algorithm proceeds by imagining trajectories from each the ensemble of models $\{f_{\boldsymbol{\phi}_1}, ..., f_{\boldsymbol{\phi}_K}\}$ using the policy $\pi_{\boldsymbol{\theta}}$. These trajectories are are used to perform the inner adaptation policy gradient step, yielding the adapted policies $\{\pi_{\boldsymbol{\theta}'_1}, ..., \pi_{\boldsymbol{\theta}'_K}\}$. Finally, we generate imaginary trajectories using the adapted policies $\pi_{\boldsymbol{\theta}'_k}$ and models $f_{\boldsymbol{\phi}_k}$, and optimize the policy towards the meta-objective (as explained in section 4.2). We iterate through these steps until desired performance is reached. The algorithm returns the optimal pre-update parameters $\boldsymbol{\theta}^*$.

5 Benefits of the Algorithm

Meta-learning a policy over an ensemble of dynamic models using imaginary trajectory roll-outs provides several benefits over traditional model-based and model-based model-free approaches. In the following we discuss several such advantages, aiming to provide intuition for the algorithm.

Regularization effect during training. Optimizing the policy to adapt within one policy gradient step to any of the fitted models imposes a regularizing effect on the policy learning (as [43] observed in the supervised learning case). The meta-optimization problem steers the policy towards higher plasticity in regions with high dynamics model uncertainty, shifting the burden of adapting to model discrepancies towards the inner policy gradient update.

We consider plasticity as the policy's ability to change its (conditional) distribution with a small change (i.e. gradient update) in the parameter space. The policy plasticity is manifested in the statistical distance between the pre- and post-update policy. In section 6.3 we analyze the connection between model uncertainty and the policy plasticity, finding a strong positive correlation between the model ensembles predictive variance and the KL-divergence between $\pi_{\boldsymbol{\theta}}$ and $\pi_{\boldsymbol{\theta}'_k}$. This effect prevents the policy to learn sub-optimal behaviors that arise in robust policy optimization. More importantly, this regularization effect fades away once the dynamics models get more accurate, which leads to asymptotic optimal policies if enough data is provided to the learned models. In section 6.4, we show how this property allows us to learn from noisy and highly biased models.

Tailored data collection for fast model improvement. Since we sample real-environment trajectories using the different policies $\{\pi_{\boldsymbol{\theta}'_1}, ..., \pi_{\boldsymbol{\theta}'_K}\}$ obtained by adaptation to each model, the collected training data is more diverse which promotes robustness of the dynamic models. Specifically, the

adapted policies tend to exploit the characteristic deficiencies of the respective dynamic models. As a result, we collect real-world data in regions where the dynamic models insufficiently approximate the true dynamics. This effect accelerates correcting the imprecision of the models leading to faster improvement. In Appendix A.1, we experimentally show the positive effect of tailored data collection on the performance.

Fast fine-tuning. Meta-learning optimizes a policy for fast adaptation [11] to a set of tasks. In our case, each task corresponds to a different believe of what the real environment dynamics might be. When optimal performance is not achieved, the ensemble of models will present high discrepancy in their predictions, increasing the likelihood of the real dynamics to lie in the believe distribution's support. As a result, the learned policy is likely to exhibit high adaptability towards the real environment, and fine-tuning the policy with VPG on the real environment leads to faster convergence than training the policy from scratch or from any other MB initialization.

Simplicity. Our approach, contrary to previous methods, is simple: it does not rely on parameter noise exploration, careful reinitialization of the model weights or policy's entropy, hard to train probabilistic models, and it does not need to address the model distribution mismatch [23, 10, 35].

6 Experiments

The aim of our experimental evaluation is to examine the following questions: 1) How does MB-MPO compare against state-of-the-art model-free and model-based methods in terms of sample complexity and asymptotic performance? 2) How does the model uncertainty influence the policy's plasticity? 3) How robust is our method against imperfect models?

To answer the posed questions, we evaluate our approach on six continuous control benchmark tasks in the Mujoco simulator [44]. A depiction of the environments as well a detailed description of the experimental setup can be found in Appendix A.3. In all of the following experiments, the pre-update policy is used to report the average returns obtained with our method. The performance reported are averages over at least three random seeds. The source code and the experiments data is available on our supplementary website [†].

6.1 Comparison to State-of-the-Art: Model-Free

We compare our method in sample complexity and performance to four state-of-the-art model free RL algorithms: Deep Deterministic Policy Gradient (DDPG) [2], Trust Region Policy Optimization [1], Proximal Policy Optimization (PPO) [45], and Actor Critic using Kronecker-Factored Trust Region (ACKTR) [46]. The results are shown in Figure 1.

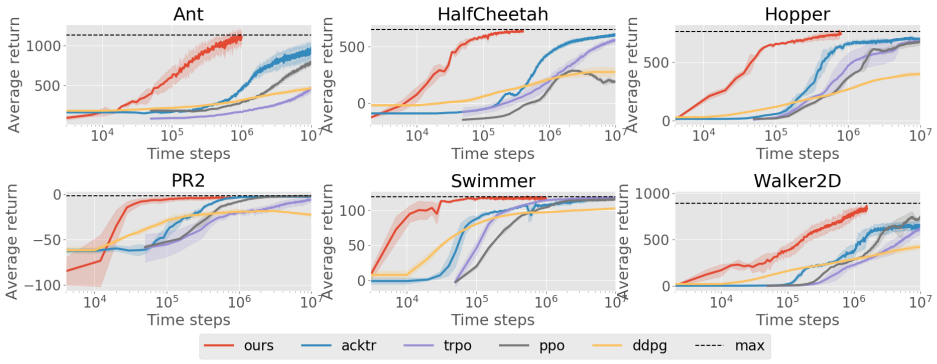

Figure 1: Learning curves of MB-MPO ("ours") and four state-of-the-art model-free methods in six different Mujoco environments with a horizon of 200. MB-MPO is able to match the asymptotic performance of model-free methods with two orders of magnitude less samples.

In all the locomotion tasks we are able to achieve maximum performance using between 10 and 100 times less data than model-free methods. In the most challenging domains: ant, hopper, and

[†]https://sites.google.com/view/mb-mpo

walker2D; the data complexity of our method is two orders of magnitude less than the MF. In the easier tasks: the simulated PR2 and swimmer, our method achieves the same performance of MF using 20-50$\times$ less data. These results highlight the benefit of MB-MPO for real robotics tasks; the amount of real-world data needed for attaining maximum return corresponds to 30 min in the case of easier domains and to 90 min in the more complex ones.

6.2 Comparison to State-of-the-Art: Model-Based

We also compare our method against recent model-based work: Model-Ensemble Trust-Region Policy Optimization (ME-TRPO) [10], and the model-based approach introduced in Nagabandi et al. [22], which uses MPC for planning (MB-MPC).

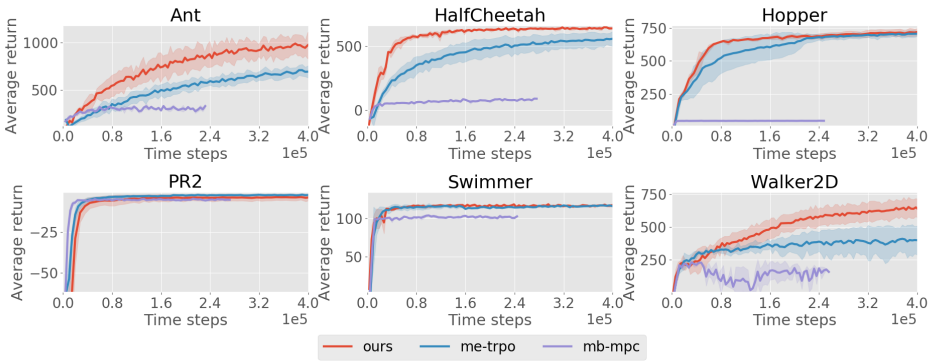

Figure 2: Learning curves of MB-MPO ("ours") and two MB methods in 6 different Mujoco environments with a horizon of 200. MB-MPO achieves better asymptotic performance and faster convergence rate than previous MB methods.

The results, shown in Figure 2, highlight the strength of MB-MPO in complex tasks. MB-MPC struggles to perform well on tasks that require robust planning, and completely fails in tasks where medium/long-term planning is necessary (as in the case of hopper). In contrast, ME-TRPO is able to learn better policies, but the convergence to such policies is slower when compared to MB-MPO . Furthermore, while ME-TRPO converges to suboptimal policies in complex domains, MB-MPO is able to achieve max-performance.

6.3 Model Uncertainty and Policy Plasticity

In section 6.3 we hypothesize that the meta-optimization steers the policy towards higher plasticity in regions with high dynamics model uncertainty while embedding consistent model predictions into the pre-update policy. To empirically analyze this hypothesis, we conduct an experiment in a simple 2D-Point environment where the agent, starting uniformly from $[-2, 2]^2$, must go to the goal position $(0, 0)$. We use the average KL-divergence between $\pi_{\boldsymbol{\theta}}$ and the different adapted policies $\pi_{\boldsymbol{\theta}'_k}$ to measure the plasticity conditioned on the state $\boldsymbol{s}$.

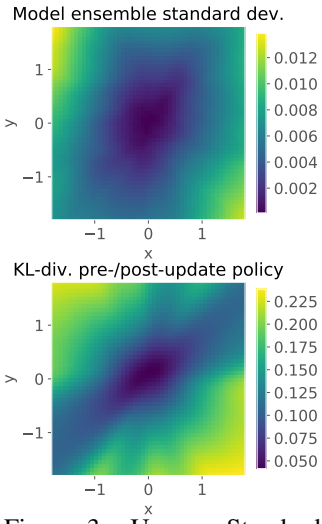

Figure 3: Upper: Standard deviation of model ensemble predictions Lower: KL-divergence between pre- and post-update policy (after 50 MB-MPO iterations in the 2-D Point env). The x and y axis denote the state-space dimensions of the 2-D Point environment

Figure 3 depicts the KL-divergence between the pre- and post-update policy, as well as the standard deviation of the predictions of the ensemble over the state space. Since the agent steers towards the center of the environment, more transition data is available in this region. As a result the models present higher accuracy in the center. The results indicate a strong positive correlation between model uncertainty and the KL-divergence between pre- and post-update policy. We find this connection between policy plasticity and predictive uncertainty consistently throughout the training and among different hyper-parameter configurations.

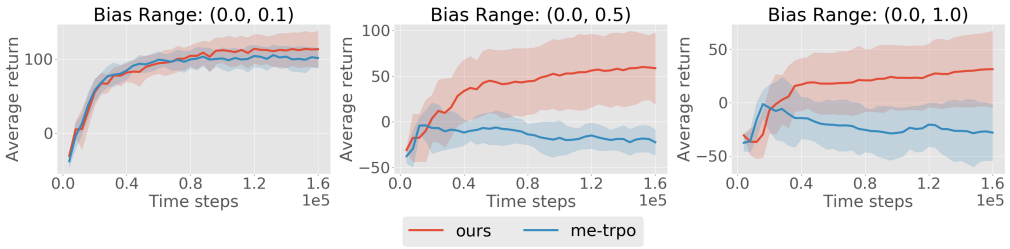

Figure 4: Comparison of MB-MPO ("ours") and ME-TRPO using 5 biased and noisy dynamic models in the half-cheetah environment with a horizon of 100 time steps. A bias term b is sampled uniformly from a denoted interval in every iteration. During the iterations we add to the predicted observation a Gaussian noise $\mathcal{N}(b, 0.1)$.

6.4 Robustness to Imperfect Dynamic Models and Compounding Errors

We pose the question of how robust our proposed algorithm is w.r.t. imperfect dynamics predictions. We examine it in two ways. First, with an illustrative example of a model with clearly wrong dynamics. Specifically, we add biased Gaussian noise $\mathcal{N}(b, 0.1^2)$ to the next state prediction, whereby the bias $b \sim \mathcal{U}(0, b_{\max})$ is re-sampled in every iteration for each model. Second, we present a realistic case on which long horizon predictions are needed. Bootstrapping the model predictions for long horizons leads to high compounding errors, making policy learning on such predictions challenging.

Figure 4 depicts the performance comparison between our method and ME-TRPO on the half-cheetah environment for various values of $b_{\max}$. Results indicate that our method consistently outperforms ME-TRPO when exposed to biased and noisy dynamics models. ME-TPRO catastrophically fails to learn a policy in the presence of strong bias (i.e. $b_{\max} = 0.5$ and $b_{\max} = 1.0$), but our method, despite the strongly compromised dynamic predictions, is still able to learn a locomotion behavior with a positive forward velocity.

This property also manifests itself in long horizon tasks. Figure 5 compares the performance of our approach with inner learning rate $\alpha = 10^{-3}$ against the edge case $\alpha = 0$, where no adaption is taking place. For each random seed, MB-MPO steadily converges to maximum performance. However, when there is no adaptation, the learning becomes unstable and different seeds exhibit different behavior: proper learning, getting stuck in sub-optimal behavior, and even unlearning good behaviors.

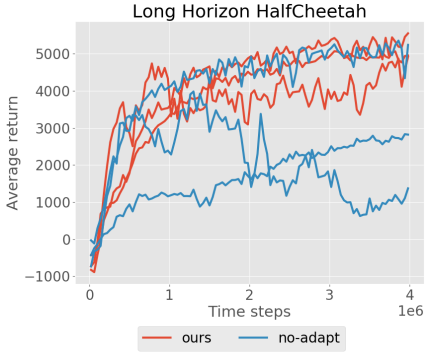

Figure 5: Comparison of our method with and without adaptation. Depicted is the development of average returns during training with three different random seeds on the half-cheetah environment with a horizon of 1000 time steps.

7 Conclusion

In this paper, we present a simple and generally applicable algorithm, model-based meta-policy optimization (MB-MPO), that learns an ensemble of dynamics models and meta-optimizes a policy for adaptation in each of the learned models. Our experimental results demonstrate that meta-learning a policy over an ensemble of learned models provides the recipe for reaching the same level of performance as state-of-the-art model-free methods with substantially lower sample complexity. We also compare our method against previous model-based approaches, obtaining better performance and faster convergence. Our analysis demonstrate the ineffectiveness of prior approaches to combat model-bias, and showcases the robustness of our method against imperfect models. As a result, we are able to extend model-based to more complex domains and longer horizons. One direction that merits further investigation is the usage of Bayesian neural networks, instead of ensembles, to learn a distribution of dynamics models. Finally, an exciting direction of future work is the application of MB-MPO to real-world systems.

Acknowledgments

We thank A. Gupta, C. Finn, and T. Kurutach for the feedback on the earlier draft of the paper. IC was supported by La Caixa Fellowship. The research leading to these results received funding from the EU Horizon 2020 Research and Innovation programme under grant agreement No. 731761 (IMAGINE) and was supported by Berkeley Deep Drive, Amazon Web Services, and Huawei.

References

[1] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust Region Policy Optimization. ICML, 2015.

[2] T. P. Lillicrap et al. Continuous control with deep reinforcement learning. CoRR, abs/1509.02971, 2015.

[3] D. Silver et al. Mastering the game of Go with deep neural networks and tree search. Nature, Jan. 2016.

[4] M. P. Deisenroth, G. Neumann, and J. Peters. A survey on policy search for robotics. Found. Trends Robot, 2, 2013.

[5] V. Pong, S. Gu, M. Dalal, and S. Levine. Temporal Difference Models: Model-Free Deep RL for Model-Based Control. In ICLR, 2018.

[6] M. Deisenroth and C. E. Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In ICML, pages 465–472, 2011.

[7] A. Rajeswaran, S. Ghotra, B. Ravindran, and S. Levine. EPOpt: Learning Robust Neural Network Policies Using Model Ensembles. 10 2016.

[8] K. Zhou, J. C. Doyle, and K. Glover. Robust and Optimal Control. Prentice-Hall, Inc., 1996.

[9] S. H. Lim, H. Xu, and S. Mannor. Reinforcement learning in robust markov decision processes. In NIPS. 2013.

[10] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel. Model-Ensemble Trust-Region Policy Optimization. In ICLR, 2018.

[11] C. Finn, P. Abbeel, and S. Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In ICML, 2017.

[12] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel. RL$ˆ2$: Fast Reinforcement Learning via Slow Reinforcement Learning. 11 2016.

[13] J. X. Wang et al. Learning to reinforcement learn. CoRR, abs/1611.05763, 2017.

[14] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. A Simple Neural Attentive Meta-Learner. In ICLR, 7 2018.

[15] F. Sung, L. Zhang, T. Xiang, T. M. Hospedales, and Y. Yang. Learning to learn: Meta-critic networks for sample efficient learning. CoRR, abs/1706.09529, 2017.

[16] J. A. Bagnell and J. G. Schneider. Autonomous helicopter control using reinforcement learning policy search methods. In ICRA, volume 2, pages 1615–1620. IEEE, 2001.

[17] P. Abbeel, M. Quigley, and A. Y. Ng. Using inaccurate models in reinforcement learning. In ICML, 2006.

[18] S. Levine and P. Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In NIPS, pages 1071–1079, 2014.

[19] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. Journal of Machine Learning Research, 17(39):1–40, 2016.

[20] D. Nguyen-Tuong, M. Seeger, and J. Peters. Local gaussian process regression for real time online model learning and control. In NIPS, pages 1193–1200, 2009.

[21] S. Kamthe and M. P. Deisenroth. Data-efficient reinforcement learning with probabilistic model predictive control. CoRR, abs/1706.06491, 2017.

[22] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine. Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning. 8 2017.

[23] K. Chua, R. Calandra, R. Mcallister, and S. Levine. Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. 2019.

[24] A. Punjani and P. Abbeel. Deep learning helicopter dynamics models. In ICRA, pages 3223–3230, 2015.

[25] N. Wahlström, T. B. Schön, and M. P. Deisenroth. From pixels to torques: Policy learning with deep dynamical models. CoRR, abs/1502.02251, 2015.

[26] S. Depeweg, F. Doshi-velez, and S. Udluft. Learning and Policy Search in Stochastic Dynamical Systems with Bayesian Neural Networks. In ICML, 2017.

[27] I. Clavera, A. Nagabandi, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn. Learning to adapt: Meta-learning for model-based control. CoRR, abs/1803.11347, 2018.

[28] J. Fu, S. Levine, and P. Abbeel. One-shot learning of manipulation skills with online dynamics adaptation and neural network priors. CoRR, abs/1509.06841, 2015.

[29] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine. Continuous deep Q-learning with model-based acceleration. In ICML. JMLR.org, 2016.

[30] I. Lenz, R. A. Knepper, and A. Saxena. Deepmpc: Learning deep latent features for model predictive control. In Robotics: Science and Systems, 2015.

[31] N. Mishra, P. Abbeel, and I. Mordatch. Prediction and Control with Temporal Segment Models. In ICML, 2017.

[32] N. Heess et al. Learning Continuous Control Policies by Stochastic Value Gradients. 10 2015.

[33] R. S. Sutton and R. S. Dyna, an integrated architecture for learning, planning, and reacting. ACM SIGART Bulletin, 2(4), 7 1991.

[34] T. Weber et al. Imagination-Augmented Agents for Deep Reinforcement Learning.

[35] V. Feinberg, A. Wan, I. Stoica, M. I. Jordan, J. E. Gonzalez, and S. Levine. Model-Based Value Expansion for Efficient Model-Free Reinforcement Learning. 2018.

[36] J. Schmidhuber. Evolutionary principles in self-referential learning. on learning now to learn: The meta-meta-meta...-hook. Diploma thesis, Technische Universitat Munchen, Germany, 14 May 1987.

[37] M. Andrychowicz, M. Denil, S. G. Colmenarejo, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas. Learning to learn by gradient descent by gradient descent. CoRR, abs/1606.04474, 2016.

[38] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. One-shot learning with memory-augmented neural networks. arXiv preprint arXiv:1605.06065, 2016.

[39] M. Hüsken and C. Goerick. Fast learning for problem classes using a knowledge based network initialization. In IJCNN, 2000.

[40] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In ICLR, 2018.

[41] T. Salimans and D. P. Kingma. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. In NIPS, 2 2016.

[42] J. Peters and S. Schaal. Policy gradient methods for robotics. In IROS, pages 2219–2225, Oct 2006.

[43] A. Nichol, J. Achiam, and J. S. Openai. On First-Order Meta-Learning Algorithms.

[44] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In IROS, 2012.

[45] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms. CoRR, 2017.

[46] Y. Wu, E. Mansimov, S. Liao, R. B. Grosse, and J. Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. CoRR, abs/1708.05144, 2017.

[47] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In AISTATS, 2011.

[48] G. Brockman et al. Openai gym. CoRR, abs/1606.01540, 2016.

[49] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. In ICLR, 2016.

[50] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel. Benchmarking deep reinforcement learning for continuous control. In ICML, page 13291338, 2016.

A Appendix

A.1 Tailored Data Collection

We present the effects of collecting data using tailored exploration. We refer to tailored exploration as the effect of collecting data using the post-update policies – the policies adapted to each specific model. When training policies on learned models they tend to exploit the deficiencies of the model, and thus overfitting to it. Using the post-update policies to collect data results in exploring the regions of the state space where these policies overfit and the model is inaccurate. Iteratively collecting data in the regions where the models are innacurate has been shown to greatly improve the performance [47].

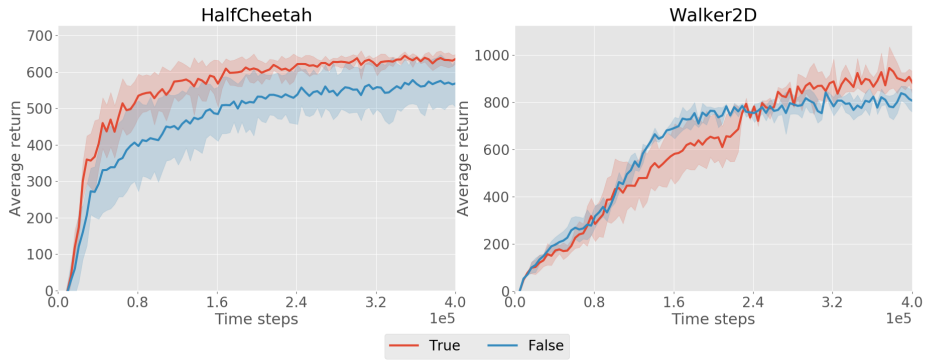

Figure 6: Tailored exploration study in the half-cheetah and walker2D environment. "True" means the data is collected by using tailored exploration, and "False" is the result of not using it, i.e., using the pre-update policy to collect data.

The effect of using tailored exploration is shown in Figure 6. In the half-cheetah and the walker we get an improvement of 12% and 11%, respectively. The tailored exploration effect cannot be accomplished by robust optimization algorithms, such as ME-TRPO. Those algorithms learn a single policy that is robust across models. The data collection using such policy will not exploit the regions in which each model fails resulting in less accurate models.

A.2 Hyperparameter Study

We perform a hyperparameter study (see Figure 7) to assess the sensitivity of MB-MPO to its parameters. Specifically, we vary the inner learning rate α, the size of the ensemble, and the number of meta gradient steps before collecting further real environment samples. Consistent with the results in Figure 5, we find that adaptation significantly improves the performance when compared to the non-adaptive case of $\alpha = 0$. Increasing the number of models and meta gradient steps per iteration results in higher performance at a computational cost. However, as the computational burden is increased the performance gains diminish.

Up to a certain level, increasing the number of meta gradient steps per iteration improves performance. Though, too many meta gradients steps (i.e. 60) can lead to early convergence to a suboptimal policy. This may be due to the fact that the variance of the Gaussian policy distribution is also learned. Usually, the policies variance decreases during the training. If the number of meta-gradient steps is too large, the policy loses its exploration capabilities too early and can hardly improve once the models are more accurate. This problem can be alleviated using a fixed policy variance, or by adding an entropy bonus the learning objective.

A.3 Experiment Setup

In the following we provide a detailed description of the setup used in the experiments presented in section 6:

Environments:

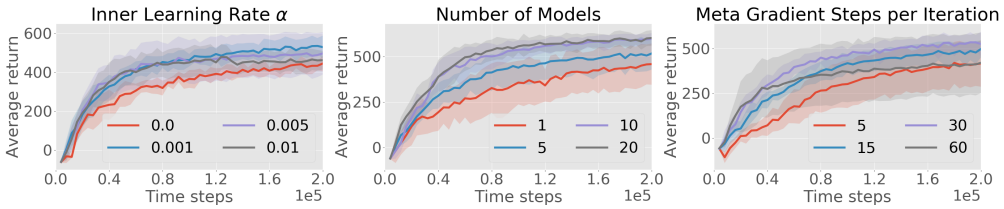

Figure 7: Hyper-parameter study in the the half-cheetah environment of a) the inner learning rate α, b) the number of dynamic models in the ensemble, and c) the number of meta gradient steps before collecting real environment samples and refitting the dynamic models.

Figure 8: Mujoco environments used in our experiments. Form left to right: swimmer, half-cheetah, walker2D, PR2, hopper, and ant.

We benchmark MB-MPO on six continuous control benchmark tasks in the Mujoco simulator [44], shown in Fig. 8. Five of these tasks, namely swimmer, half-cheetah, walker2D, hopper and ant, involve robotic locomotion and are provided trough the OpenAI gym [48].

The sixth, the 7-DoF arm of the PR2 robot, has to reach arbitrary end-effector positions. Thereby, the PR2 robot is torque controlled. The reward function is comprised of the squared distance of the end-effector (TCP) to the goal and energy / control costs:

$$r(s,a) = -||s_{\text{TCP}} - x_{\text{goal}}||_2^2 - 0.05 * ||a||_2^2$$

In section 6.3 we use the simple 2D-Point environment to analyze the connection between policy plasticity and model uncertainty. The corresponding MDP is defined as follows:

$$\begin{aligned}
\mathcal{S} &= \mathbb{R}^2 \\
\mathcal{A} &= [-0.1, 0.1]^2 \\
p_0(s_0) &= \mathcal{U}_{[-2,2]^2}(s_0) \quad \text{(uniform distribution over } [-2,2]^2\text{)} \\
p(s_{t+1}|s_t, a_t) &= \delta(s_t + a_t) \\
r(s_t, a_t) &= -||s_t||_2^2 \\
H &= 30
\end{aligned}$$

Policy: We use a Gaussian policy $\pi_\theta(a|s) = \mathcal{N}(a|\mu(a)_{\theta_\mu}, \sigma_{\theta_\sigma})$ with diagonal covariance matrix. The mean $\mu(a)_{\theta_\mu}$ is computed by a neural network (2 hidden layers of size 32, tanh nonlinearity) which receives the current state s as an input. During the policy optimization, both the weights θ_μ of the neural network and the standard deviation vector σ_{θ_σ} are learned.

Advantage-Estimation: We use generalized advantage estimation (GAE) [49] with $\gamma = 0.99$ and $\lambda = 1$ in conjunction with a linear reward baseline as in [50] to estimate advantages.

Dynamics Model Ensemble: In all experiments (except in Figure 7b) we use an ensemble of 5 fully connected neural networks. For the different environments the following hidden layer sizes were used:

- Ant, Walker: (512, 512, 512)
- PR2, Swimmer, Hopper, Half-Cheetah: (512, 512)
- 2D-Point-Env: (128, 128)

In all models, we used weight normalization and ReLu nonlinearities. For the minimization of the l^2 prediction error, the Adam optimizer with a batch-size of 500 was employed. In the first iteration all models are randomly initialized. In later iterations, the models are trained with warm starts using the parameters of the previous iteration. In each iteration and for each model in the ensemble the

transition data buffer $\mathcal{D}$ is randomly split in a training (80%) and validation (20%) set. The latter split is used to compute the validation loss after each training epoch on the shuffled training split. A rolling average of the validation losses with a persistence of 0.95 is maintained throughout the epochs. Each model's training is stopped individually as soon as the rolling validation loss average decreases.

Meta-Policy Optimization: As described in section 4.2, the policy parameters θ are optimized using the gradient-based meta learning framework MAML. For the inner adaptation step we use a gradient step-size of $\alpha = 0.001$. For maximizing the meta-objective specified in equation 3 we use the policy gradient method TPRO [1] with KL-constraint $\delta = 0.01$. Since computing the gradients of the meta-objective involves second order terms such as the Hessian of the policy's log-likelihood, computing the necessary Hessian vector products for TRPO analytically is very compute intensive. Hence, we use a finite difference approximation of the vector product of the Fisher Information Matrix and the gradients as suggested in [11]. If not denoted differently, 30 meta-optimization steps are performed before new trajectories are collected from the real environment.

Trajectory collection: In each algorithm iteration 4000 environment transitions (20 trajectories of 200 time steps) are collected. For the meta-optimization, 100000 imaginary environment transitions are sampled.

A.4 Computational Analysis

In this section we compare the computational complexity of MB-MPO against TRPO. Specifically, we report the wall clock time that it takes both algorithms to reach maximum performance on the half-cheetah environment when running the experiments on an Amazon Web Services EC2 c4.4xlarge compute instance. Our method only requires 20% more compute time than TRPO (7 hours instead of 5.5), while attaining $70\times$ reduction in sample complexity. The main time bottleneck of our method compared with the model-free algorithms is training the models.

Notice that when running real world experiment, our method will be significantly faster than model-free approaches since the bottleneck then would shift towards the data collection step.