LIVE IN THE MOMENT: LEARNING DYNAMICS MODEL ADAPTED TO EVOLVING POLICY

Anonymous authors
Paper under double-blind review

ABSTRACT

Model-based reinforcement learning (RL) often achieves higher sample efficiency in practice than model-free RL by learning a dynamics model to generate samples for policy learning. Previous works learn a dynamics model that fits under the empirical state-action visitation distribution for all historical policies, i.e., the sample replay buffer. However, in this paper, we observe that fitting the dynamics model under the distribution for *all historical policies* does not necessarily benefit model prediction for the *current policy* since the policy in use is constantly evolving over time. The evolving policy during training will cause state-action visitation distribution shifts. We theoretically analyze how this distribution shift over historical policies affects the model learning and model rollouts. We then propose a novel dynamics model learning method, named *Policy-adapted Dynamics Model Learning (PDML)*. PDML dynamically adjusts the historical policy mixture distribution to ensure the learned model can continually adapt to the state-action visitation distribution of the evolving policy. Experiments on a range of continuous control environments in MuJoCo show that PDML achieves significant improvement in sample efficiency and higher asymptotic performance combined with the state-of-the-art model-based RL methods.

1 INTRODUCTION

Recent years have witnessed great successes of Reinforcement Learning (RL) in many complex decision-making tasks, such as robotics (Polydoros & Nalpantidis, 2017; Yang et al., 2022) and chess games (Silver et al., 2016; Schrittwieser et al., 2020). Among RL methods, a wide range of works in model-free RL (Schulman et al., 2015; Lillicrap et al., 2016; Haarnoja et al., 2018; Fujimoto et al., 2018; Hu et al., 2021) have shown promising performance. However, model-free methods can be impractical for real-world scenarios (Dulac-Arnold et al., 2021) since massive samples from the real environment are required for policy training, resulting in low sample efficiency.

Model-based RL is considered one of the solutions to improve sample efficiency. Most of the model-based RL algorithms first use supervised learning techniques to learn a dynamics model based on the samples obtained from the real environment, and then use this learned dynamics model to generate massive samples to derive a policy (Luo et al., 2018; Janner et al., 2019). Therefore, it is crucial to learn a dynamics model which can accurately simulate the underlying transition dynamics of the real environment since the policy is trained based on the model-generated samples. If the learned dynamics has a high prediction error, the model-generated samples will be biased, and the policy induced by these samples will be sub-optimal. To reduce the model prediction error and learn an accurate dynamics model, some advanced architectures such as model ensemble (Kurutach et al., 2018; Chua et al., 2018) and multi-step model (Asadi et al., 2019) have been proposed to improve the multi-step prediction accuracy of the learned dynamics model. Besides, the idea of a generative adversarial network (GAN) (Goodfellow et al., 2014) is used to design the training process of a dynamics model (Shen et al., 2020; Eysenbach et al., 2021) to reduce the distribution mismatch between model-generated samples and real samples. Those previous works mentioned above aim to learn a dynamics model that can fit all historical policies. To be precise, when training the dynamics model, they randomly select the training data from the real samples obtained by all historical policies in the replay buffer. This learned dynamics model needs to adapt to the state-action visitation distribution of all historical policies to obtain a dynamics model that predicts transitions accurately under different policies.

However, since we only use the current newest policy to interact with the learned model to generate samples for policy learning during model rollouts, learning such a dynamics model that fits under (highly likely sub-optimal) historical policies may be unnecessary. Due to the state-action visitation distribution shift during policy updating, the state-action pairs visited by historical policies may not appear in the state-action visitation distribution of the current policy, and vice versa. Thus, learning these samples may not benefit model rollouts. Besides, in many complex tasks, it is hard to predict all samples from all historical policies due to limited model capacity (Abbas et al., 2020), and as shown later in our paper, trying to learn every sample from historical policies can even hurt the accuracy when predicting the transitions induced by the current policy. Therefore, there is an objective mismatch between model learning and model rollouts — model learning tries to fit samples from state-action visitation distribution for all historical policies, whereas model rollouts require accurate prediction of the transitions induced by the current policy.

In this paper, we investigate how to learn an accurate dynamics model for model rollouts based on existing samples. **(a)** To begin with, we confirm through experiments that although the dynamics model learned by the previous methods has a low overall prediction error on all transitions obtained by historical policies, its prediction error for the current newest policy can still be very high. This leads to inaccurate model-generated samples which can hurt the sample efficiency and asymptotic performance of the policy. **(b)** We then derive an upper bound of the expected performance gap between the model rollouts and real environment rollouts. According to this upper bound, we analyze how the distribution of historical policies affects model learning and model rollouts. The theoretical result suggests that the historical policy distribution used for model learning should be more inclined towards policies that are closer to the current policy rather than a uniform distribution over all historical policies to ensure the model prediction accuracy for model rollouts. **(c)** Motivated by this insight, we propose a novel dynamics model learning method named *Policy-adapted Dynamics Model Learning (PDML)*. Instead of learning a dynamics model that fits under a uniform mixture of all historical policies, PDML adjusts the historical policy distribution by reducing the total variation distance between the historical policy mixture and the current policy, then learns a policy-adapted dynamics model according to this adjusted historical policy distribution. **(d)** We conduct systematic and extensive experiments on a range of continuous control benchmark MuJoCo environments (Todorov et al., 2012). Experimental results show that PDML significantly improves the sample efficiency and asymptotic performance of the state-of-the-art model-based RL methods.

Summary of contributions: **(1)** Through detailed experimental results, we establish that learning a dynamics model that fits a uniform mixture of all historical policies may not be accurate enough for model rollouts. **(2)** We propose an upper bound of an expected performance gap between the model rollouts and the real environment rollouts, and theoretically analyze how the distribution over historical policies affects model learning and model rollouts. **(3)** We propose *Policy-adapted Dynamics Model Learning (PDML)*, which dynamically adjusts the distribution over the historical policy sequence and allows the learned model to continuously adapt to the evolving policy. **(4)** Experimental results on a range of MuJoCo environments demonstrate that PDML can achieve significant improvement in sample efficiency and higher asymptotic performance combined with the state-of-the-art model-based RL methods.

2 Background

2.1 Preliminaries

Reinforcement learning. Consider a Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, T, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, and $T(s'|s,a)$ is the transition dynamics in the real world. The reward function is denoted as $r(s,a)$ and γ is the discount factor. Reinforcement learning aims to find an optimal policy π which can maximize the expected sum of discounted rewards

$$\pi = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_{\substack{s_t \sim T(\cdot|s_{t-1},a_{t-1}) \\ a_t \sim \pi(a|s_t)}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]. \tag{1}$$

In model-based RL, the transition dynamics T in the real world is unknown, and we aim to construct a model $\hat{T}(s'|s,a)$ of transition dynamics and use it to improve the policy. In this paper, we concentrate on the Dyna-style (Sutton, 1990) model-based RL, which uses the learned dynamics model to generate samples and train the policy.

Policy mixture. During policy learning, we consider the historical policies at iteration step k as a historical policy sequence $\Pi^k = \{\pi_1, \pi_2, ..., \pi_k\}$. For each policy in the policy sequence, we denote its state-action visitation distribution as $\rho^{\pi_i}(s,a)$, and the policy mixture distribution over the policy sequence as $\boldsymbol{w}^k = [w_1^k \dots, w_k^k]$. Then the state-action visitation distribution of the policy mixture $\pi_{\text{mix},k} = (\Pi^k, \boldsymbol{w}^k)$ is $\rho^{\pi_{\text{mix},k}}(s,a) = \sum_{i=1}^k w_i^k \rho^{\pi_i}(s,a)$ (Hazan et al., 2019; Zhang et al., 2021).

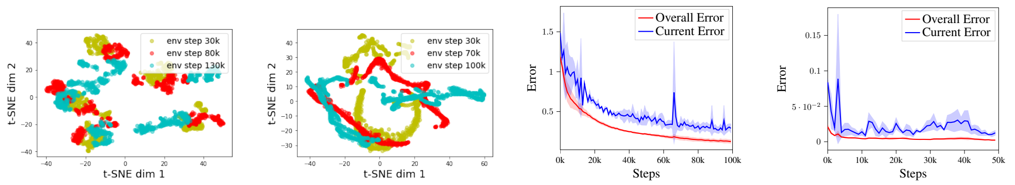
2.2 Dynamics Model Learning in Model-based RL

Learning a dynamics model is the most crucial part of model-based RL since the ground-truth transition dynamics is unknown and the policy must be updated based on the samples generated by the learned dynamics model. Previous works learn the dynamics model by randomly selecting training data from the samples obtained by the historical policy sequence Π^k, which means the distribution of policy mixture is a random distribution: $w_i^k = \frac{1}{k}$. The learned dynamics model is trained based on the following state-action visitation distribution

$$\rho^{\pi_{\text{mix},k}}(s,a) = \sum_{i=1}^{k} \frac{1}{k} \rho^{\pi_i}(s,a). \tag{2}$$

This model tries to fit all the samples obtained by sampling the state-action visitation distribution corresponding to all policies in the historical policy sequence, so the learned dynamics model is (hopefully) able to predict the transition for any state-action input.

However, as shown in Figure 1a and 1b, since the policy is constantly evolving, the state-action visitation distribution of historical policies may have a huge shift from the current policy. There is little overlap between the state-action visitation distribution of policies at different environment steps. The state-action pairs visited by historical policies may not appear in the state-action visitation distribution of the current policy. During model rollouts, we only use the current policy to interact with the learned dynamics model to generate samples. Thus, learning these samples may not benefit model rollouts. When the model capacity is not large enough, learning these samples may even be detrimental to the learning of the samples collected by the current policy.



(a) HalfCheetah (b) Hopper (c) HalfCheetah (d) Hopper

Figure 1: **(a)** and **(b)**: visualization of the state-action visitation distribution of different historical policies and the current policy using t-SNE. Env step 130k and env step 100k are the current policy. More details are shown in Appendix F.2. **(c)** and **(d)**: the overall error curves and current error curves of MBPO on HalfCheetah and Hopper, respectively.

We conduct an experiment using a state-of-the-art model-based RL method called MBPO (Janner et al., 2019) on four MuJoCo (Todorov et al., 2012) environments HalfCheetah, Hopper, Walker2d, and Ant. MBPO first trains a model based on the real samples and then uses the model to roll out multiple samples for policy learning. The architecture of the dynamics model is a 4-layer neural network with a hidden size of 200, which is a very common architecture used in many recent model-based methods (Yao et al., 2021; Froehlich et al., 2022; Li et al., 2022). We present the overall error curves and the current error curves during learning steps on HalfCheetah and Hopper in Figure 1c and 1d. Here the overall error means the model prediction error for all historical policies during training. It is evaluated on an evaluation dataset which contains 1000 $\times N$ samples from the real environment. N is the number of historical policies in the historical policy sequence. The current error is the model prediction error for the current policy, which is evaluated using L2 error on the 1000 samples obtained by the current policy from the real environment. The error curves for more environments can be found in Appendix F.1.

From Figure 1c and 1d, we observe that there is a gap between the overall error and the current error. This means although the agent can learn a dynamics model which is good enough for all samples obtained by historical policies, this is at the expense of the prediction accuracy for the samples induced by current policy. Since we only use the current policy during model rollouts, this will lead to inaccurate model-generated samples and misleading policy learning.

Therefore, learning a dynamics model that adapts the state-action visitation distribution for all historical policies, in other words, a random historical policy mixture distribution used for model learning, is not the most efficient way for model-based RL (especially for task-specific problems). In the next section, we will analyze how the policy mixture distribution affects the performance of model-based RL.

3 Performance Gap Influenced by Policy Mixture Distribution

In this section, we provide a theoretical analysis of how the policy mixture distribution affects the performance of model-based RL. First, we derive a theorem that upper bounds the performance gap between the real environment rollouts and the model rollouts under any current policy π.

Theorem 3.1 *Given the historical policy mixture $\pi_{mix,k} = (\Pi^k, \boldsymbol{w}^k)$ at iteration step k, we denote $\xi_{\rho_i} = D_{TV}(\rho_T^{\pi}(s,a)||\rho_T^{\pi_i}(s,a))$ as the state-action visitation distribution shift and $\xi_{\pi_i} = \mathbb{E}_{s\sim v_{\hat{T}}^{\pi_{mix}}}[D_{TV}(\pi(a|s)||\pi_i(a|s))]$ as the policy distribution shift between the historical policy π_i and current policy π respectively, where $v_{\hat{T}}^{\pi_{mix}}$ is the state visitation distribution of the policy mixture under the learned dynamics model $\hat{T}$. r_{max} is the maximum reward the policy can get from the real environment, γ is the discount factor, and $\mathrm{Vol}(\mathcal{S})$ is the volume of state space. Then the performance gap between the real environment rollout $J(\pi, T)$ and the model rollout $J(\pi, \hat{T})$ can be bounded as:*

$$\begin{aligned} J(\pi,T) - J(\pi,\hat{T}) \leq\ & 2\gamma r_{max}\mathbb{E}_{(s,a)\sim\rho_T^{\pi}}[D_{TV}(T(s'|s,a)||\hat{T}(s'|s,a))] \\ & + r_{max}\sum_{i=1}^{k} w_i^k(\gamma \mathrm{Vol}(\mathcal{S})\xi_{\rho_i} + 2\xi_{\pi_i}) \\ & + 2r_{max}D_{TV}(\rho_{\hat{T}}^{\pi_{mix}}(s,a)||\rho_{\hat{T}}^{\pi}(s,a)) \end{aligned} \tag{3}$$

Proof. See Appendix D. □

Remarks.
(1) The first term is about **model prediction error**. This term suggests that the model needs to adapt to the state-action visitation distribution of the *current policy* to reduce the model prediction error, since this term is the expectation of prediction error of the learned dynamics model $\hat{T}$ under the current policy state-action visitation distribution ρ_T^{π}.
(2) The second term shows the effect of the policy mixture distribution on **model rollout**. This item contains two distribution shifts: **(2a)** state-action visitation distribution shift ξ_{ρ_i} and **(2b)** policy distribution shift ξ_{π_i} between the historical policy and current policy. It should be noted that ξ_{ρ_i} is induced by ξ_{π_i}, so it is reasonable to believe that a historical policy with a larger ξ_{π_i} will have a larger ξ_{ρ_i}. Both ξ_{ρ_i} and ξ_{π_i} are fixed since historical policies and the current policy are immutable during model learning and model rollout. Therefore, to reduce this term, we can only adjust the policy mixture distribution $\boldsymbol{w}^k$. Since the distribution shift varies across historical policies and the current policy, it is obvious that the random distribution $w_i^k = \frac{1}{k}$ is not the best choice.
(3) The last term is related to the model sample buffer, which is used for **policy learning**. To maximize sample utilization, the model-generated samples obtained by the historical policies will be maintained in the model sample buffer until they are replaced by the new samples generated by the current policy. Therefore, the distribution of simulated samples in the model buffer is not exactly the same as the simulated sample distribution of the current policy, but is often mixed with the simulated sample distribution of the historical policies. This makes it necessary to adjust the sample distribution in the model sample buffer to make it close to the simulated sample distribution of the current policy during the policy learning process. This has been studied in many model-based and model-free methods (Schaul et al., 2016; Liu et al., 2021; Huang et al., 2021; Mu et al., 2021) and is out of the scope of this paper, and we focus on reducing the first two terms related to model learning.

The first two items on the right-hand side of Equation (3) provide useful insights on model learning. This first term points out the goal of model learning: to make accurate predictions for the current policy. The second item further demonstrates that to achieve this goal, we should adjust the policy mixture distribution to reduce the distribution shift between the historical policy mixture and the current policy. According to the second term, we have the following proposition.

Proposition 3.2 *The performance gap can be reduced if the weight w_i^k of each policy π_i in the historical policy sequence Π^k is negatively related to state action visitation distribution shift ξ_{ρ_i} and the policy distribution shift ξ_{π_i} between the historical policy π_i and current policy π instead of an average weight $w_i^k = \frac{1}{k}$.*

The proof is in Appendix E. Proposition 3.2 illustrates how we should adjust the policy distribution to help the learned dynamics model adapt to the current policy. This naturally motivates our method, which is described in the next section.

4 Policy-adapted Dynamics Model Learning

In this section, we introduce our model learning method called *Policy-adapted Dynamics Model Learning* (PDML). PDML is designed to reduce the model prediction error during model rollouts, and it contains two parts. The first part is adjusting the policy mixture distribution into a non-uniform distribution, and the second part is learning the dynamics model based on this non-uniform distribution. The pseudo-code is in Algorithm 1.

Algorithm 1 Policy-adapted Dynamics Model Learning (PDML)

```
Require: current policy proportion hyperparameter $\alpha$, interaction epochs $I$
 1: Initialize historical policy sequence $k \leftarrow 0, \Pi^k \leftarrow \emptyset$
 2: for $I$ epochs do
 3:     Interact with the environment using current policy $\pi_c$, add samples into real sample buffer $\mathbb{D}_e$
 4:     Add current policy $\pi_c$ into historical policy sequence: $\pi_k \leftarrow \pi_c$, $\Pi^k \leftarrow \{\Pi^{k-1}, \pi_k\}$
 5:     Adjust the historical policy mixture distribution $\boldsymbol{w}^k = [w_1^k, \ldots, w_k^k]$ via Equation (4) and (5)
 6:     Normalize $\boldsymbol{w}_k \leftarrow \boldsymbol{w}_k / \|\boldsymbol{w}_k\|$
 7:     Sample a training data batch of $(s_n, a_n, r, s_{n+1})$ from $\mathbb{D}_e$ according to $\boldsymbol{w}^k$
 8:     Train dynamics model $\hat{T}_\theta$ via Equation (7), use current policy $\pi_c$ to perform model rollouts
 9:     $k \leftarrow k + 1$
10: end for
```

4.1 Policy Mixture Distribution Adjustment

In this section, we introduce a mechanism to adjust the policy mixture distribution. According to our Theorem 3.1, to minimize the performance gap, one may set the weight of the policy with the smallest ξ_{ρ_i} and ξ_{π_i} to be 1 and the weights of other policies in the historical policy sequence to be 0. However, this is not the best approach in practice since each policy can only interact with the environment for very few steps in model-based RL. This means each policy can provide very limited samples for model learning. If we only use a small number of samples from just one policy, it is difficult to learn accurate transition dynamics for the current policy.

Weights design for historical policies. In order to maximize the use of limited samples to estimate the transition dynamics, inspired by Proposition 3.2, we design the weight of each policy in the historical policy sequence $\Pi^k = \{\pi_1, \pi_2, ..., \pi_k\}$ except for the current policy π_c (i.e., $\pi_k \in \Pi^k$) as follows:

$$w_i^k = \frac{{\xi_{\pi_i}}^{-1}}{\sum_{n=1}^k {\xi_{\pi_n}}^{-1}}, \quad \xi_{\pi_i} = \mathbb{E}_{s \sim v_{\hat{T}}^{\pi_{\text{mix}}}} \left[D_{TV}(\pi_c(\cdot|s) || \pi_i(\cdot|s)) \right], \quad \forall i \in [k-1], \tag{4}$$

where ξ_{π_i} is the policy distribution shift between historical policy π_i^k and the current policy π_c; it is also one of the distribution shifts in the second term of Equation (3). We use $[k-1] := \{1, \ldots, k-1\}$ to denote the integers from 1 to $k-1$. We only use the policy distribution shift ξ_{π_i} (and not the state-action visitation distribution shift ξ_{ρ_i}) because estimating the state-action visitation distribution shift using limited real samples is difficult, and thus the estimation may be inaccurate. Besides, as mentioned in the remarks of Theorem 3.1, state-action visitation distribution is induced by the policy, so it is reasonable to believe that a historical policy with a larger ξ_{π_i} will have a larger ξ_{ρ_i}.

Weight design for the current policy. In model-based RL, the current policy becomes a historical policy after interacting with the environment and is added to the historical policy sequence (see Algorithm 1). The total variation distance between the current policy and itself is 0, so Equation (4) cannot be used to calculate the weight of the current policy. For the weight of the current policy w_k^k, we use the following equation:

$$w_k^k = \begin{cases} \alpha \sum_{i=1}^{k-1} w_i^k, & \text{if} \quad \alpha \sum_{i=1}^{k-1} w_i^k > \underset{i\in[k-1]}{max}\{w_i^k\} \\ \underset{i\in[k-1]}{max}\{w_i^k\}, & \text{if} \quad \alpha \sum_{i=1}^{k-1} w_i^k \leq \underset{i\in[k-1]}{max}\{w_i^k\} \end{cases} \tag{5}$$

where α is a hyperparameter to control the proportion of the weight of the current policy to the total weight over the historical policy sequence. Equation (5) ensures that the weight of the current policy w_k^k is always the largest in the historical policy sequence. Before each model learning iteration, we adjust the policy mixture distribution according to Equation (4) and Equation (5) and normalize the weights $\boldsymbol{w}^k = [w_1^k, ..., w_k^k]$ to make sure they sum to 1. The details are illustrated in Algorithm 1.

Estimation of the policy distribution shift ξ_{π_i} $\forall i \in [k-1]$. Given a state s_n, we define the output of policy π_i as a multivariate Gaussian distribution $\mathcal{N}(\mu_{\pi_i^n}, \Sigma_{\pi_i^n})$. In order to make the empirical estimation more accurate, we use each historical policy to traverse all N samples in the real sample buffer and output the action distribution corresponding to each state. Then we use the inequality between KL divergence and total variation distance to estimate ξ_{π_i}:

$$\begin{aligned} \xi_{\pi_i} &= \frac{1}{N}\sum_{n=1}^{N} D_{TV}(\pi_c(\cdot|s_n)||\pi_i(\cdot|s_n)) \\ &\leq \frac{1}{2N}\sum_{n=1}^{N}\sqrt{tr(\Sigma_{\pi_c^n}^{-1}\Sigma_{\pi_i^n} - I) + (\mu_{\pi_c^n} - \mu_{\pi_i^n})^{\top}\Sigma_{\pi_i^n}^{-1}(\mu_{\pi_c^n} - \mu_{\pi_i^n}) - \log\det(\Sigma_{\pi_c^n}^{-1}\Sigma_{\pi_i^n})} \end{aligned} \tag{6}$$

Novelty of PDML compared to prioritized experience replay proposed in model-free RL. In model-free RL, prioritized experience replay methods only need to consider how to improve the policy based on existing samples. Therefore, it is only necessary to select the sample that can bring the greatest improvement to the policy, and a weighting is designed for each sample. In model-based RL, the policy is learned based on model-generated samples, and the accuracy of these model-generated samples determines the sub-optimality of the policy. Thus, in the model-learning part, we focus on the model prediction accuracy. Our theoretical analysis shows that we should consider whether the state-action visitation distribution that generates the samples is close to the current policy when reweighting samples. Although a sample can bring a great improvement to the current policy (the TD value is high), if this sample is not in the state-action visitation distribution of the current policy, this sample will not be encountered during model rollouts. Then learning this sample will not bring any benefit to model learning and policy learning. Therefore, we reweight the state-action visitation distribution that generates a batch of samples according to ξ_{π_i}, rather than a single sample as in model-free RL.

4.2 Dynamics Model Learning

After adjusting the policy mixture distribution, we learn the dynamics model based on this adjusted distribution. Although our method can be applied to learn any type of dynamic model, here we choose to use the current state-of-the-art structure probabilistic dynamics model ensemble Chua et al. (2018): $\{\hat{T}_\theta^1, ..., \hat{T}_\theta^B\}$. θ is the parameters of each dynamics model in the ensemble, and B is the ensemble size. Given a (s_n, a_n) pair as an input, the output $\hat{T}_\theta^b$ of each network b in the ensemble is the Multivariate Gaussian Distribution of the next state: $\hat{T}_\theta^b(s_{n+1}|s_n, a_n) = \mathcal{N}(\mu_\theta^b(s_n, a_n), \Sigma_\theta^b(s_n, a_n))$ Before each model learning iteration, we sample the training data batch from the real sample buffer according to the adjusted policy mixture distribution $\boldsymbol{w}^k$, and train the dynamics model using maximum likelihood:

$$\mathcal{L}(\theta) = \sum_{n=1}^{N}[\mu_\theta^b(s_n, a_n) - s_{n+1}]^\top {\Sigma_\theta^b}^{-1}(s_n, a_n)[\mu_\theta^b(s_n, a_n) - s_{n+1}] + \log\det\Sigma_\theta^b(s_n, a_n) \tag{7}$$

During model rollouts, we use the current policy π_c as the rollout policy and sample the initial states from the real sample buffer according to the adjusted policy mixture distribution $\boldsymbol{w}^k$.

5 Experiment

In this section, we will first compare our method with the previous state-of-the-art (including both model-free and model-based) baselines. We demonstrate that after combining with SOTA model-based method, PDML improves SOTA sample efficiency and SOTA asymptotic performance for

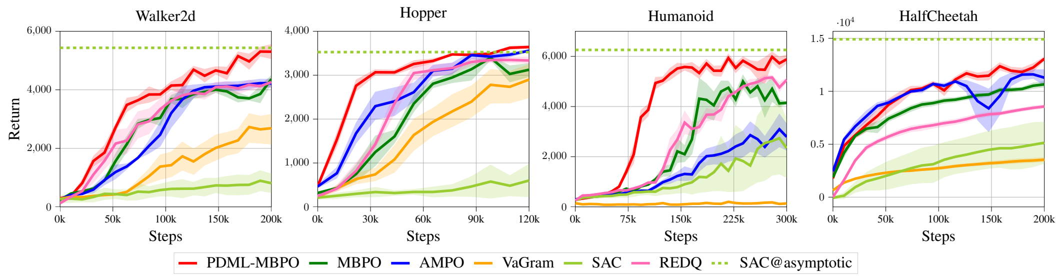


Figure 2: Performance curves for our method (PDML-MBPO) and other baseline methods on four MuJoCo environments. Our method, AMPO, MBPO and VaGram are model-based methods, while SAC and REDQ are model-free methods. The dashed line indicates the asymptotic performance of SAC. The solid lines indicate the mean over 8 seeds and shaded regions correspond to the 95% confidence interval among seeds. We evaluate the performance every 1k interaction steps.

model-based RL. Then we compare our method with three SOTA prioritized experience replay methods to indicate the advantage of our distribution adjustment method for model learning. Lastly, we conduct a systematic ablation study to analyze the model errors of PDML.

5.1 Comparison with State-of-the-arts

In this section, we compare our method with several previous state-of-the-art (SOTA) baselines. For model-based methods, we choose MBPO (Janner et al., 2019), AMPO (Shen et al., 2020), and VaGraM (Voelcker et al., 2022). MBPO is the SOTA model-based method, and our method is combined with MBPO for the model learning part. We name our method PDML-MBPO and we provide the pseudo code in Appendix A. AMPO is another SOTA model-based method that uses unsupervised model adaptation during model learning to reduce the prediction error. VaGraM is a SOTA value equivalence model-based method. Instead of accurately learning each dimension in the dynamics, it aims to learn the dimensions which impact policy learning most. In other words, this method also learns a locally accurate model. Both AMPO and VaGraM are implemented based on MBPO. PDML-MBPO, AMPO, and VaGraM share the same model architecture and policy part; only the model learning part is different. For model-free methods, we compare with two methods. The first one is SAC (Haarnoja et al., 2018), which is the policy part of all model-based and model-free baselines we used and is one of the SOTA model-free methods. The second one is REDQ (Chen et al., 2020), which improves the Update-To-Data (UTD) ratio of the model-free method and achieves higher sample efficiency than SAC. The implementation details of our method are in Appendix G.1. We conduct experiment on four complex MoJoCo-v2 (Todorov et al., 2012) environments, the performance curves are shown in Figure 2.

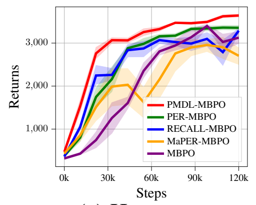
Results: (1) Improving SOTA sample efficiency. PDML-MBPO outperforms all existing state-of-the-art methods, including model-based and model-free, in sample efficiency in all four environments. In Hopper, Walker2d, and Humanoid, PDML-MBPO achieves very impressive sample efficiency improvements, up to a $2\times$ improvement in Hopper and Humanoid compared to the SOTA model-based methods. For example, our method using only 30k steps to achieve 3000 while other model-based methods need almost 60k steps. Besides, its sample efficiency is also higher than REDQ which is modified for sample-efficient model-free RL. **(2) Improving SOTA asymptotic performance for Model-based RL.** In addition, PDML-MBPO obtains significantly better asymptotic performance compared to other state-of-the-art model-based methods. It is worth noting that the asymptotic performance of PDML-MBPO is very close to SAC in three environments (Hopper, Walker2d, and Humanoid) and is even better than SAC occasionally. Furthermore, our method achieves impressive improvement in the most complex environment Humanoid. These indicate the effectiveness of our proposed model learning method.

Discussion of computational cost. PDML requires saving all historical policies as well as computing their distances to the current policy for adjusting their weights (as shown in Equation 6). This creates an additional memory overhead of storing historical policy networks ($k\times$policy network size) and an additional computational overhead of computing the distances, for each iteration k. In PDML-MBPO, we observe storing historical policy networks costs a memory overhead of no more than $k\times 1$ MB, compared with the high memory occupied by the model sample buffer, this cost is very small. Besides, Compared to MBPO, the training time of PDML-MBPO does not increase significantly, we provide the training time on four environments in Appendix F.8.

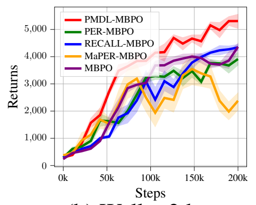
5.2 Comparison with Model-free Experience Replay Methods

We compare with the other three prioritized experience replay methods in model-free RL to indicate the advantage of our PDML. The first one is Prioritized Experience Replay (PER) (Schaul et al., 2016), which weighs the samples according to their TD-error. The second method is RECALL (Goyal et al., 2018), which chooses the top k highest value sample. They use this to recall the samples that induce the high-value trajectories and train the policy. We implement this by choosing the top 25% highest Q value samples to train the model and as model rollout initial states. The third method is Model-augmented Prioritized Experience Replay (MaPER) (Oh et al., 2022), which is an extension of PER using both TD-error and model prediction error to weight the samples for model learning.

The experiment results are shown in Figure 3a and 3b. Our PDML significantly outperforms all three methods on both sample efficiency and asymptotic performance. We believe these methods adjust the weights for each sample in the training data rather than each policy. This will cause the samples belonging to the same state-action visitation distribution to have different weights, and the samples with higher weights may not necessarily appear in the state-action visitation distribution of current policy. Therefore, the learned model cannot be adapted to current policy's state-action visitation distribution, and the model prediction error during model rollouts cannot be reduced. In the model learning process, it is crucial to adapt to current policy's state-action visitation distribution according to our theory. This experiment result indicates our theory's correctness and our method's effectiveness.

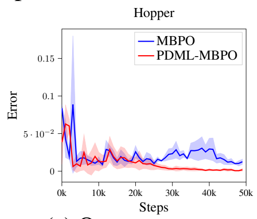


(a) Hopper

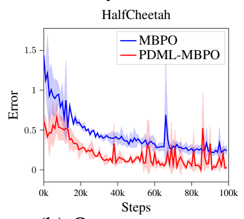


(b) Walker2d

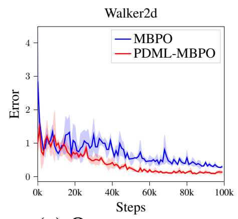
Figure 3: The comparison of model-free experience replay methods on Hopper and Walker2d. The experiments are run for 8 random seeds.



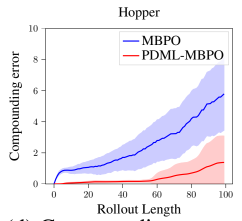
(a) One-step error



(b) One-step error



(c) One-step error



(d) Compounding errors

Figure 4: **(a), (b) and (c)** display one-step (model-prediction) error for PDML-MBPO and MBPO. **(d)** demonstrates the compounding error (i.e., the difference between the h-step state in the model rollout trajectory and the real environment rollout trajectory) of PDML-MBPO and MBPO over 20 model rollout trajectories.

5.3 Model Error Analysis

To further verify the impact of PDML, we compare the one-step prediction error and the compounding error of the policy-adapted model learned by PDML-MBPO and the original dynamics model learned by MBPO.

One-step prediction error. As shown in Figure 4a, 4b and 4c, we evaluate the model prediction error for the current policy on Hopper, HalfCheetah, and Walker2d. We evaluate the learned model every 1000 environment steps using L2 error on the 1000 samples obtained by the current policy from the real environment. The error curves show that the one-step prediction error for the current policy of the policy-adapted model is much smaller than that of the original dynamics model, which means the model-generated samples of PDML-MBPO are more accurate than MBPO, so the policy induced by PDML-MBPO can perform better.

Compounding error. We also compare the multi-step model rollouts compounding error of the policy-adapted model and the original dynamics model. This directly determines the accuracy of the model-generated samples in each model rollout trajectory. Figure 4d shows the compounding error curves of the policy-adapted model and the original dynamics model on Hopper. We calculate the h-step compounding error as the difference between the state at each rollout step h in the model rollout trajectory and the real environment rollout trajectory using L2 error. The results demonstrate that the policy-adapted model has much a smaller compounding error than the original dynamics model, which means the policy-adapted model has a more robust multi-step planning capability than the original dynamics model learned by MBPO.

6 Related Work

Model adaptation. Several adaptive control approaches (Sastry & Isidori, 1989; Pastor et al., 2011; Meier et al., 2016) aim to train a dynamics model that can adapt online. However, scaling such methods to complex tasks is exponentially difficult. Adaptive learning in the dynamics model has also been studied in inverse dynamics learning tasks. A drifting Gaussian process (GP) keeps a history of a constant number of recently observed data points and updates its hyper-parameters at each time step (Meier & Schaal, 2016). The drifting Gaussian process (GP) predicts the local dynamics errors to control the learning rate (Meier et al., 2016), resulting in more online hyperparameter learning and adaptive function approximator robustness. Our method is different from these works that we learn a forward model which can always adapt to the evolving policy. Some studies focus on an adaptive model predictive control for constrained linear systems (Tanaskovic et al., 2013) and guaranteeing safety, robustness, and convergence in a quadrotor helicopter testbed (Aswani et al., 2012). Our work closely relates to a model adaptation in forward models from Fu et al. (2016); Nagabandi et al. (2018a;b); Lee et al. (2020); Guo et al. (2022). These methods use meta-learning to train a dynamics model as a prior and then combine it with recent data to rapidly adapt to the new task. However, these works are mainly about model transfer under different dynamics. Different from their works, we study learning an accurate dynamics model for policy learning under a fixed transition dynamics, and we also provide theoretical analysis to motivate our method. More related works about model-based RL are provided in Appendix B.

Prioritized experience replay. Another related line of work is prioritized experience replay in reinforcement learning. This solves a classic issue in model-free RL. Previous work Katharopoulos & Fleuret (2018) claimed that emphasizing essential samples in the replay buffer can benefit off-policy RL algorithms. Prioritized Experience Replay (PER) Schaul et al. (2016) measured the importance of sample by temporal-difference (TD) error. Based on this work, many methods are proposed to perform prioritized sampling. Some methods (Brittain et al., 2019; Lee et al., 2019; Fujimoto et al., 2020; Jiang et al., 2021; Liu et al., 2021; Lahire et al., 2021; Oh et al., 2022) extend or explain PER from different perspectives, and others (Novati & Koumoutsakos, 2019; Fedus et al., 2020) propose to prioritize samples according to their age. Our work is different from experience replay works in model-free RL in the following points: **(1)** In model learning, we re-weight the state-action visitation distribution that generates a batch of samples, rather than a single sample as in model-free RL. **(2)** During weighting, we use the distance between the policy distribution that each sample generated from and the policy distribution of the current policy as a metric, rather than how much improvement each sample can bring to the policy. **(3)** We provide very detailed theoretical result to analyze how to reweight the samples for model learning.

7 Conclusion and Discussion

In this paper, we introduce a novel dynamics model learning method for model-based RL called PDML, which learns a policy-adapted dynamics model based on a dynamically adjusted historical policy mixture distribution. This policy-adapted dynamics model can continually adapt to the state-action visitation distribution of the evolving policy. This makes it more accurate than the previous dynamics model when making predictions during model rollouts. We also provide theoretical analysis and experimental results to motivate our method. After combining with the state-of-the-art model-based method MBPO, PDML achieves better asymptotic performance and higher sample efficiency than previous state-of-the-art model-based methods in MuJoCo. We believe our work takes an important step toward more sample-efficient RL. One limitation of our work is that the generalization ability of the policy-adapted dynamics model may not be strong enough because we focus on fitting the samples induced by the evolving policy to improve the convergence speed of the policy. Therefore, our method is efficient for task-specific problems but may not perform well for some exploration-oriented tasks. We leave this direction to future work.

REFERENCES

Zaheer Abbas, Samuel Sokota, Erin Talvitie, and Martha White. Selective dyna-style planning under limited model capacity. In *International Conference on Machine Learning*, pp. 1–10. PMLR, 2020.

Kavosh Asadi, Dipendra Misra, Seungchan Kim, and Michel L Littman. Combating the compounding-error problem with a multi-step model. *arXiv preprint arXiv:1905.13320*, 2019.

Anil Aswani, Patrick Bouffard, and Claire Tomlin. Extensions of learning-based model predictive control for real-time application to a quadrotor helicopter. In *2012 American Control Conference (ACC)*, pp. 4661–4666. IEEE, 2012.

Marc Brittain, Josh Bertram, Xuxi Yang, and Peng Wei. Prioritized sequence experience replay. *arXiv preprint arXiv:1905.12726*, 2019.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith W Ross. Randomized ensembled double q-learning: Learning fast without a model. In *International Conference on Learning Representations*, 2020.

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

Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472. Citeseer, 2011.

Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, pp. 1–50, 2021.

Benjamin Eysenbach, Alexander Khazatsky, Sergey Levine, and Ruslan Salakhutdinov. Mismatched no more: Joint model-policy optimization for model-based rl. *arXiv preprint arXiv:2110.02758*, 2021.

Amir-massoud Farahmand. Iterative value-aware model learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Amir-massoud Farahmand, Andre Barreto, and Daniel Nikovski. Value-aware loss function for model-based reinforcement learning. In *Artificial Intelligence and Statistics*, pp. 1486–1494. PMLR, 2017.

William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. In *International Conference on Machine Learning*, pp. 3061–3071. PMLR, 2020.

Lukas Froehlich, Maksym Lefarov, Melanie Zeilinger, and Felix Berkenkamp. On-policy model errors in reinforcement learning. In *International Conference on Learning Representations*, 2022.

Justin Fu, Sergey Levine, and Pieter Abbeel. One-shot learning of manipulation skills with online dynamics adaptation and neural network priors. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4019–4026. IEEE, 2016.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pp. 1587–1596. PMLR, 2018.

Scott Fujimoto, David Meger, and Doina Precup. An equivalence between loss functions and non-uniform sampling in experience replay. *Advances in neural information processing systems*, 33: 14219–14230, 2020.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Anirudh Goyal, Philemon Brakel, William Fedus, Soumye Singhal, Timothy Lillicrap, Sergey Levine, Hugo Larochelle, and Yoshua Bengio. Recall traces: Backtracking models for efficient reinforcement learning. In *International Conference on Learning Representations*, 2018.

Christopher Grimm, André Barreto, Satinder Singh, and David Silver. The value equivalence principle for model-based reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 5541–5552, 2020.

Jixian Guo, Mingming Gong, and Dacheng Tao. A relational intervention approach for unsupervised dynamics generalization in model-based reinforcement learning. *arXiv preprint arXiv:2206.04551*, 2022.

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

Elad Hazan, Sham Kakade, Karan Singh, and Abby Van Soest. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, pp. 2681–2691. PMLR, 2019.

Hao Hu, Jianing Ye, Guangxiang Zhu, Zhizhou Ren, and Chongjie Zhang. Generalizable episodic memory for deep reinforcement learning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 4380–4390, 2021.

Wenzhen Huang, Qiyue Yin, Junge Zhang, and Kaiqi Huang. Learning to reweight imaginary transitions for model-based reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 7848–7856, 2021.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in Neural Information Processing Systems*, 32:12519–12530, 2019.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning*, pp. 4940–4950. PMLR, 2021.

Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*, pp. 2525–2534. PMLR, 2018.

Vikash Kumar, Emanuel Todorov, and Sergey Levine. Optimal control with learned local models: Application to dexterous manipulation. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 378–383. IEEE, 2016.

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

Thibault Lahire, Matthieu Geist, and Emmanuel Rachelson. Large batch experience replay. *arXiv preprint arXiv:2110.01528*, 2021.

Hang Lai, Jian Shen, Weinan Zhang, and Yong Yu. Bidirectional model-based policy optimization. In *International Conference on Machine Learning*, pp. 5618–5627. PMLR, 2020.

Hang Lai, Jian Shen, Weinan Zhang, Yimin Huang, Xing Zhang, Ruiming Tang, Yong Yu, and Zhenguo Li. On effective scheduling of model-based reinforcement learning. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

Kimin Lee, Younggyo Seo, Seunghyun Lee, Honglak Lee, and Jinwoo Shin. Context-aware dynamics model for generalization in model-based reinforcement learning. In *International Conference on Machine Learning*, pp. 5757–5766. PMLR, 2020.

Su Young Lee, Choi Sungik, and Sae-Young Chung. Sample-efficient deep reinforcement learning via episodic backward update. *Advances in Neural Information Processing Systems*, 32, 2019.

Chongchong Li, Yue Wang, Wei Chen, Yuting Liu, Zhi-Ming Ma, and Tie-Yan Liu. Gradient information matters in policy optimization by back-propagating through model. In *International Conference on Learning Representations*, 2022.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *ICLR*, 2016.

Xu-Hui Liu, Zhenghai Xue, Jingcheng Pang, Shengyi Jiang, Feng Xu, and Yang Yu. Regret minimization experience replay in off-policy reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Yuting Liu, Jiahao Xu, and Yiming Pan. [re] when to trust your model: Model-based policyoptimization. *ReScience C*, 6(2), 2020. Accepted at NeurIPS 2019 Reproducibility Challenge.

Ângelo G Lovatto, Thiago P Bueno, Denis D Mauá, and Leliane N Barros. Decision-aware model learning for actor-critic methods: when theory does not meet practice. 2020.

Yuping Luo, Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. In *International Conference on Learning Representations*, 2018.

Franziska Meier and Stefan Schaal. Drifting gaussian processes with varying neighborhood sizes for online model learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 264–269. IEEE, 2016.

Franziska Meier, Daniel Kappler, Nathan Ratliff, and Stefan Schaal. Towards robust online inverse dynamics learning. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4034–4039. IEEE, 2016.

Yao Mu, Yuzheng Zhuang, Bin Wang, Guangxiang Zhu, Wulong Liu, Jianyu Chen, Ping Luo, Shengbo Li, Chongjie Zhang, and Jianye Hao. Model-based reinforcement learning via imagination with derived memory. *Advances in Neural Information Processing Systems*, 34:9493–9505, 2021.

Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *International Conference on Learning Representations*, 2018a.

Anusha Nagabandi, Chelsea Finn, and Sergey Levine. Deep online learning via meta-learning: Continual adaptation for model-based rl. In *International Conference on Learning Representations*, 2018b.

Guido Novati and Petros Koumoutsakos. Remember and forget for experience replay. In *International Conference on Machine Learning*, pp. 4851–4860. PMLR, 2019.

Youngmin Oh, Jinwoo Shin, Eunho Yang, and Sung Ju Hwang. Model-augmented prioritized experience replay. In *International Conference on Learning Representations*, 2022.

Feiyang Pan, Jia He, Dandan Tu, and Qing He. Trust the model when it is confident: Masked model-based actor-critic. *Advances in Neural Information Processing Systems*, 2020.

Ronald Parr, Lihong Li, Gavin Taylor, Christopher Painter-Wakefield, and Michael L Littman. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pp. 752–759, 2008.

Peter Pastor, Ludovic Righetti, Mrinal Kalakrishnan, and Stefan Schaal. Online movement adaptation based on previous sensor experiences. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 365–371. IEEE, 2011.

Athanasios S Polydoros and Lazaros Nalpantidis. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86(2):153–173, 2017.

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

CE Rasmussen and M Kuss. Gaussian processes in reinforcement learning. *Advances in Neural Information Processing Systems*, pp. 751–759, 2004.

Sosale Shankara Sastry and Alberto Isidori. Adaptive control of linearizable systems. *IEEE Transactions on Automatic Control*, 34(11):1123–1131, 1989.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *ICLR (Poster)*, 2016.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

Jian Shen, Han Zhao, Weinan Zhang, and Yong Yu. Model-based policy optimization with unsupervised model adaptation. *Advances in Neural Information Processing Systems*, 33, 2020.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990*, pp. 216–224. Elsevier, 1990.

Richard S Sutton, Csaba Szepesvári, Alborz Geramifard, and Michael Bowling. Dyna-style planning with linear function approximation and prioritized sweeping. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, pp. 528–536, 2008.

Marko Tanaskovic, Lorenzo Fagiano, Roy Smith, Paul Goulart, and Manfred Morari. Adaptive model predictive control for constrained linear systems. In *2013 European Control Conference (ECC)*, pp. 382–387. IEEE, 2013.

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

Claas A Voelcker, Victor Liao, Animesh Garg, and Amir massoud Farahmand. Value gradient weighted model-based reinforcement learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=4-D6CZkRXxI.

Ruihan Yang, Minghao Zhang, Nicklas Hansen, Huazhe Xu, and Xiaolong Wang. Learning vision-guided quadrupedal locomotion end-to-end with cross-modal transformers. In *International Conference on Learning Representations*, 2022.

Yao Yao, Li Xiao, Zhicheng An, Wanpeng Zhang, and Dijun Luo. Sample efficient reinforcement learning via model-ensemble exploration and exploitation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4202–4208. IEEE, 2021.

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

Tianjun Zhang, Paria Rashidinejad, Jiantao Jiao, Yuandong Tian, Joseph E Gonzalez, and Stuart Russell. Made: Exploration via maximizing deviation from explored regions. *Advances in Neural Information Processing Systems*, 34, 2021.

Supplementary Material

A Pseudo Code of PDML-MBPO

In Algorithm 2, we demonstrate the pseudo code of PDML-MBPO.

Algorithm 2 PDML-MBPO

```
Require: current policy proportion hyperparameter $\alpha$, interaction epochs $I$, rollout horizon $h$
 1: Initialize historical policy sequence $k \leftarrow 0, \Pi^k \leftarrow \emptyset$
 2: for $I$ epochs do
 3:     Interact with the environment using current policy $\pi_c$, add samples into real sample buffer $\mathbb{D}_e$
 4:     Add current policy $\pi_c$ into historical policy sequence: $\pi_k \leftarrow \pi_c$, $\Pi^k \leftarrow \{\Pi^{k-1}, \pi_k\}$
 5:     Adjust the historical policy mixture distribution $\boldsymbol{w}^k = [w_1^k, \dots, w_k^k]$ via Equation (4) and (5)
 6:     Normalize $\boldsymbol{w}_k \leftarrow \boldsymbol{w}_k / \|\boldsymbol{w}_k\|$
 7:     Sample a training data batch of $(s_n, a_n, r, s_{n+1})$ from $\mathbb{D}_e$ according to $\boldsymbol{w}^k$
 8:     Train dynamics model $\hat{T}_\theta$ via Equation (7)
 9:     for $M$ model rollouts do
10:         Sample initial rollout states from real sample buffer $\mathbb{D}_e$ according to $\boldsymbol{w}^k$
11:         Use current policy $\pi_c$ to perform $h$-step model rollouts, add model-generated samples into
            model sample buffer $\mathbb{D}_m$
12:     end for
13:     for $G$ gradient updates do
14:         Update current policy $\pi_c$ using model-generated samples from model sample buffer $\mathbb{D}_m$
15:     end for
16:     $k \leftarrow k + 1$
17: end for
```

B Additional Related Work

Model-based reinforcement learning. Model-based RL is proposed as a solution to reduce the sample complexity of model-free RL by learning a dynamics model. Current model-based RL mainly focuses on better model learning and better model usage. To learn a model with more accuracy, many model architectures have been proposed, such as linear models Parr et al. (2008); Sutton et al. (2008); Kumar et al. (2016) and nonparametric Gaussian processes Rasmussen & Kuss (2004); Deisenroth & Rasmussen (2011). With the rapid development of deep learning, neural networks have become a popular choice of model architecture in recent years Kurutach et al. (2018); Chua et al. (2018). Moreover, to reduce the model error, a multi-step model Asadi et al. (2019) was designed to directly predict the transition of an action sequence input, and Shen et al. (2020) used unsupervised model adaptation to reduce the potential data distribution mismatch. For better model usage, Janner et al. (2019) proved that short model rollouts could avoid the model error and improve the quality of model samples. Based on this, Lai et al. (2020) proposed a bidirectional model rollout scheme to avoid the model error further. Furthermore, model disagreement was used to decide when to trust the model Pan et al. (2020) and regularize the model samples Yu et al. (2020). Besides, Luo et al. (2018) provided a theoretical guarantee of monotone expected reward improvement of model-based RL. Rajeswaran et al. (2020) cast model-based RL as a game-theoretic framework by formulating the optimization of model and policy as a two-player game. To save time tuning hyperparameters, Lai et al. (2021) designed an automatic scheduling framework. Abbas et al. (2020) systematically studied how the model capacity affects the model-based methods.

Value-equivalence dynamics model. Value-equivalence dynamics model has been noted by several authors in recent years. Since learning an accurate dynamics model of the world remains challenging and often requires computationally costly and data-hungry models (Lovatto et al., 2020), Farahmand et al. (2017) proposed value-aware model learning which aims to learn a value-equivalence model that induces the same Bellman operator as the real environment, rather than accurately predicting transitions. However, they replaced the value function with the supremum over a function space, and it is difficult to find a supremum for a function space parameterized by complex function approximators like neural networks. Based on this work, Farahmand (2018) proposed Iterative Value-Aware Model Learning (IterVAML) which replaced the supremum over a value function space with the value function at current iteration. Besides, Grimm et al. (2020) introduced value equivalence principle and analysed how the space of possible solutions on model learning is impacted by the choice of policies and functions. However, despite very detailed theoretical guarantees, there is still a performance gap between the value-equivalence dynamics model in the practical implementation and the model trained by the maximum likelihood estimate (Lovatto et al., 2020). Eysenbach et al. (2021) introduced a novel objective to jointly train the model and the policy. Voelcker et al. (2022) proposed Value-Gradient weighted Model loss (VaGraM) which approximated the value-aware model loss function with a Taylor expansion of value function and achieved SOTA performance across all value-aware model learning methods. Like our method, VaGraM also tries to learn a locally accurate dynamics model. The difference is that our method aims to learn the samples that the current policy may encounter as accurately as possible, while VaGraM is to learn the dimensions in the state that can bring the greatest improvement to policy learning. Experimental results demonstrate that our method outperforms VaGraM in practice.

C USEFUL LEMMA

Lemma C.1 *Shen et al. (2020) Assume the initial state distributions of the real dynamics T and the learned dynamics model $\hat{T}$ are the same. For any state s', assume $\mathcal{F}_{s'}$ is a class of real-valued bounded measurable functions on state-action space, such that $\hat{T}(s'|\cdot,\cdot) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is in $\mathcal{F}_{s'}$. Then the gap between two different state visitation distributions $v_T^{\pi_1}(s')$ and $v_{\hat{T}}^{\pi_2}(s')$ can be bounded as follows:*

$$|v_T^{\pi_1}(s') - v_{\hat{T}}^{\pi_2}(s')| \leq \gamma \mathbb{E}_{(s,a)\sim\rho_T^{\pi_1}} |T(s'|s,a) - \hat{T}(s'|s,a)| + \gamma d_{\mathcal{F}_{s'}}(\rho_T^{\pi_1}, \rho_{\hat{T}}^{\pi_2}) \tag{8}$$

Proof. For any state visitation distribution v_T^π, we have:

$$v_T^\pi(s') = (1-\gamma)v_0(s') + \gamma \int_{(s,a)} \rho_T^\pi(s,a) T(s'|s,a) \mathrm{d}s\mathrm{d}a, \tag{9}$$

where v_0 is the probability of the initial state being the state s'. Then the gap between two different state visitation distributions is:

$$\begin{aligned}
&|v_T^{\pi_1}(s') - v_{\hat{T}}^{\pi_2}(s')| \\
=&\gamma \left| \int_{(s,a)} \rho_T^{\pi_1}(s,a) T(s'|s,a) - \rho_{\hat{T}(s,a)}^{\pi_2} \hat{T}(s'|s,a) \mathrm{d}s\mathrm{d}a \right| \\
=&\gamma \left| \mathbb{E}_{(s,a)\sim\rho_T^{\pi_1}}[T(s'|s,a)] - \mathbb{E}_{(s,a)\sim\rho_{\hat{T}}^{\pi_2}}[\hat{T}(s'|s,a)] \right| \\
\leq&\gamma \left| \mathbb{E}_{(s,a)\sim\rho_T^{\pi_1}}[T(s'|s,a) - \hat{T}(s'|s,a)] \right| + \gamma \left| \mathbb{E}_{(s,a)\sim\rho_T^{\pi_1}}[\hat{T}(s'|s,a)] - \mathbb{E}_{(s,a)\sim\rho_{\hat{T}}^{\pi_2}}[\hat{T}(s'|s,a)] \right| \\
\leq&\gamma \mathbb{E}_{(s,a)\sim\rho_T^{\pi_1}} |T(s'|s,a) - \hat{T}(s'|s,a)| + \gamma d_{\mathcal{F}_{s'}}(\rho_T^{\pi_1}, \rho_{\hat{T}}^{\pi_2})
\end{aligned} \tag{10}$$

□

D PROOF OF MAIN THEOREM

Theorem D.1 *Given the historical policy mixture $\pi_{mix,k} = (\Pi^k, \boldsymbol{w}^k)$ at iteration step k, we denote $\xi_{\rho_i} = D_{TV}(\rho_T^\pi(s,a)||\rho_T^{\pi_i}(s,a))$ and $\xi_{\pi_i} = \mathbb{E}_{s\sim v_{\hat{T}}^{\pi_{mix}}}[D_{TV}(\pi(a|s)||\pi_i(a|s))]$ as the state-action visitation distribution shift and the policy distribution shift between the historical policy π_i and current policy π respectively, where $v_{\hat{T}}^{\pi_{mix}}$ is the state visitation distribution of policy mixture under the learned dynamics model. r_{max} is the maximum reward the policy can get from the real environment, γ*

is the discount factor, and $\mathrm{Vol}(\mathcal{S})$ *is the volume of state space. Then the performance gap between the real environment rollout* $J(\pi, T)$ *and the model rollout* $J(\pi, \hat{T})$ *can be bounded as follows:*

$$\begin{aligned} J(\pi, T) - J(\pi, \hat{T}) \leq 2\gamma r_{max} \mathbb{E}_{(s,a)\sim\rho_T^\pi}[D_{TV}(T(s'|s,a)||\hat{T}(s'|s,a))] \\ + r_{max} \sum_{i=0}^{k} w_i^k (\gamma \mathrm{Vol}(\mathcal{S}) \xi_{\rho_i} + 2\xi_{\pi_i}) \\ + 2r_{max} D_{TV}(\rho_{\hat{T}}^{\pi_{mix}}(s,a)||\rho_{\hat{T}}^{\pi}(s,a)) \end{aligned} \tag{11}$$

Proof.

$$\begin{aligned} &\left|J(\pi, T) - J(\pi, \hat{T})\right| \\ =&\left|J(\pi, T) - J(\pi_{\text{mix}}, \hat{T}) + J(\pi_{\text{mix}}, \hat{T}) - J(\pi, \hat{T})\right| \\ \leq&\underbrace{\left|\int_{(s,a)} (\rho_T^\pi(s,a) - \rho_{\hat{T}}^{\pi_{\text{mix}}}(s,a)) r(s,a) \mathrm{d}s\mathrm{d}a\right|}_{term1} + \underbrace{\left|\int_{(s,a)} (\rho_{\hat{T}}^{\pi_{\text{mix}}}(s,a) - \rho_{\hat{T}}^{\pi}(s,a)) r(s,a) \mathrm{d}s\mathrm{d}a\right|}_{term2} \end{aligned} \tag{12}$$

For term 1:

$$\begin{aligned} &\left|\int_{(s,a)} (\rho_T^\pi(s,a) - \rho_{\hat{T}}^{\pi_{\text{mix}}}(s,a)) r(s,a) \mathrm{d}s\mathrm{d}a\right| \\ =&\left|\int_{(s,a)} (v_T^\pi(s)\pi(a|s) - v_{\hat{T}}^{\pi_{\text{mix}}}(s)\pi_{\text{mix}}(a|s)) r(s,a) \mathrm{d}s\mathrm{d}a\right| \\ =&\left|\int_{(s,a)} (v_T^\pi(s)\pi(a|s) - v_{\hat{T}}^{\pi_{\text{mix}}}(s)\pi(a|s) + v_{\hat{T}}^{\pi_{\text{mix}}}(s)\pi(a|s) - v_{\hat{T}}^{\pi_{\text{mix}}}(s)\pi_{\text{mix}}(a|s)) r(s,a) \mathrm{d}s\mathrm{d}a\right| \\ \leq&\left|\int_{(s,a)} (v_T^\pi(s) - v_{\hat{T}}^{\pi_{\text{mix}}}(s))\pi(a|s) r(s,a) \mathrm{d}s\mathrm{d}a\right| + \left|\int_{(s,a)} (v_{\hat{T}}^{\pi_{\text{mix}}}(s)(\pi(a|s) - \pi_{\text{mix}}(a|s)) r(s,a) \mathrm{d}s\mathrm{d}a\right| \\ \leq& r_{\text{max}} \int_s \left|v_T^\pi(s) - v_{\hat{T}}^{\pi_{\text{mix}}}(s)\right| \mathrm{d}s + 2r_{\text{max}} \mathbb{E}_{s\sim v_{\hat{T}}^{\pi_{\text{mix}}}} [D_{TV}(\pi(a|s)||\pi_{\text{mix}}(a|s))] \end{aligned} \tag{13}$$

For the first term of last inequality in Eq. 13, according to Lemma. C.1 we have:

$$\begin{aligned} &r_{\text{max}} \int_s \left|v_T^\pi(s) - v_{\hat{T}}^{\pi_{\text{mix}}}(s)\right| \mathrm{d}s \\ \leq& r_{\text{max}} \gamma \mathbb{E}_{(s,a)\sim\rho_T^\pi} \int_{s'} \left|T(s'|s,a) - \hat{T}(s'|s,a)\right| \mathrm{d}s' + r_{\text{max}} \gamma \int_{s'} d_{\mathcal{F}_{s'}}(\rho_T^\pi, \rho_{\hat{T}}^{\pi^*}) \mathrm{d}s' \end{aligned} \tag{14}$$

We use total variance distance as the $\mathcal{F}_{s'}$ to measure the distance between ρ_T^π and $\rho_{\hat{T}}^{\pi_{\text{mix}}}$. Suppose we can learn a dynamics model that can perfectly adapt the state-action visitation distribution of π_{mix}, which means the difference between the model prediction and the environment next state s' is very small, and the state-action visitation density induced by the learned dynamics model $\rho_{\hat{T}}^{\pi_{\text{mix}}}$ is approximately equal to $\rho_T^{\pi_{\text{mix}}}$. This assumption is required by many model-based RL methods Voelcker et al. (2022). Then Eq. 14 can be expressed as:

$$\begin{aligned} &r_{\text{max}} \int_s \left|v_T^\pi(s) - v_{\hat{T}}^{\pi_{\text{mix}}}(s)\right| \mathrm{d}s \\ \leq& r_{\text{max}} \gamma \mathbb{E}_{(s,a)\sim\rho_T^\pi} \int_{s'} \left|T(s'|s,a) - \hat{T}(s'|s,a)\right| \mathrm{d}s' + r_{\text{max}} \gamma \int_{s'} D_{TV}(\rho_T^\pi||\rho_T^{\pi_{\text{mix}}}) \mathrm{d}s' \\ \leq& 2\gamma r_{\text{max}} \mathbb{E}_{(s,a)\sim\rho_T^\pi}[D_{TV}(T(s'|s,a)||\hat{T}(s'|s,a))] + \gamma \mathrm{Vol}(\mathcal{S}) r_{\text{max}} D_{TV}(\rho_T^\pi||\rho_T^{\pi_{\text{mix}}}) \end{aligned} \tag{15}$$

Combined Eq. 13 with Eq. 15, we can get:

$$\begin{aligned}
&\left|\int_{(s,a)}(\rho_T^{\pi}(s,a)-\rho_{\hat{T}}^{\pi_{\text{mix}}}(s,a))r(s,a)\mathrm{d}s\mathrm{d}a\right| \\
\leq& 2\gamma r_{\max}\mathbb{E}_{(s,a)\sim\rho_T^{\pi}}[D_{TV}(T(s'|s,a)||\hat{T}(s'|s,a))]+\gamma\mathrm{Vol}(\mathcal{S})r_{\max}D_{TV}(\rho_T^{\pi}(s,a)||\rho_T^{\pi_{\text{mix}}}(s,a)) \\
&+2r_{\max}\mathbb{E}_{s\sim v_{\hat{T}}^{\pi_{\text{mix}}}}[D_{TV}(\pi(a|s)||\pi_{\text{mix}}(a|s))] \\
=& 2\gamma r_{\max}\mathbb{E}_{(s,a)\sim\rho_T^{\pi}}[D_{TV}(T(s'|s,a)||\hat{T}(s'|s,a))]+\gamma\mathrm{Vol}(\mathcal{S})r_{\max}D_{TV}(\rho_T^{\pi}(s,a)||\sum_{i=0}^{k}w_i\rho_T^{\pi_i}(s,a)) \\
&+2r_{\max}\mathbb{E}_{s\sim v_{\hat{T}}^{\pi_{\text{mix}}}}\left[D_{TV}(\pi(a|s)||\sum_{i=0}^{k}w_i\pi_i(a|s))\right] \\
=& 2\gamma r_{\max}\mathbb{E}_{(s,a)\sim\rho_T^{\pi}}[D_{TV}(T(s'|s,a)||\hat{T}(s'|s,a))]+\gamma\mathrm{Vol}(\mathcal{S})r_{\max}\sum_{i=0}^{k}w_iD_{TV}(\rho_T^{\pi}(s,a)||\rho_T^{\pi_i}(s,a)) \\
&+2r_{\max}\sum_{i=0}^{k}w_i\mathbb{E}_{s\sim v_{\hat{T}}^{\pi_{\text{mix}}}}[D_{TV}(\pi(a|s)||\pi_i(a|s))]
\end{aligned} \tag{16}$$

Finally, based on Eq. 16, we get:

$$\begin{aligned}
&\left|J(\pi,T)-J(\pi,\hat{T})\right| \\
\leq& 2\gamma r_{\max}\mathbb{E}_{(s,a)\sim\rho_T^{\pi}}[D_{TV}(T(s'|s,a)||\hat{T}(s'|s,a))]+\gamma\mathrm{Vol}(\mathcal{S})r_{\max}\sum_{i=0}^{k}w_i^kD_{TV}(\rho_T^{\pi}(s,a)||\rho_T^{\pi_i}(s,a)) \\
&+2r_{\max}\sum_{i=0}^{k}w_i^k\mathbb{E}_{s\sim v_{\hat{T}}^{\pi_{\text{mix}}}}[D_{TV}(\pi(a|s)||\pi_i(a|s))]+2r_{\max}D_{TV}(\rho_{\hat{T}}^{\pi_{\text{mix}}}(s,a)||\rho_{\hat{T}}^{\pi}(s,a)) \\
\leq& 2\gamma r_{\max}\mathbb{E}_{(s,a)\sim\rho_T^{\pi}}[D_{TV}(T(s'|s,a)||\hat{T}(s'|s,a))]+r_{\max}\sum_{i=0}^{k}w_i^k(\gamma\mathrm{Vol}(\mathcal{S})\xi_{\rho_i}+2\xi_{\pi_i}) \\
&+2r_{\max}D_{TV}(\rho_{\hat{T}}^{\pi_{\text{mix}}}(s,a)||\rho_{\hat{T}}^{\pi}(s,a)),
\end{aligned} \tag{17}$$

and the proof is completed. □

E Proof of Proposition 3.2

Proposition E.1 *The performance gap can be reduced if the weight w_i^k of each policy π_i in the historical policy sequence Π^k is negatively related to state action visitation distribution shift ξ_{ρ_i} and the policy distribution shift ξ_{π_i} between the historical policy π_i and current policy π instead of an average weight $w_i^k = \frac{1}{k}$:*

$$\sum_{i=1}^{k}w_i^k(\gamma\mathrm{Vol}(\mathcal{S})\xi_{\rho_i}+2\xi_{\pi_i})\leq\sum_{i=1}^{k}\frac{1}{k}(\gamma\mathrm{Vol}(\mathcal{S})\xi_{\rho_i}+2\xi_{\pi_i}) \tag{18}$$

Proof. Each policy π_i in the historical policy sequence Π^k corresponds to a distribution shift pair $(\xi_{\rho_i},\xi_{\pi_i})$, and these pairs form a distribution shift sequence $\{(\xi_{\rho_1},\xi_{\pi_1}),(\xi_{\rho_2},\xi_{\pi_2}),......,(\xi_{\rho_k},\xi_{\pi_k})\}$, assuming that this sequence decreases as i increases (this is a reasonable assumption, because we can always arrange the historical policy sequence into a distribution shift decreasing sequence according to the magnitude of the shift). As the weight of each policy is negatively related to state action visitation distribution shift ξ_{ρ_i} and the policy distribution shift ξ_{π_i}, w_i^k increases with k.

Since $\sum\limits_{i=1}^{k} w_i^k = \sum\limits_{i=1}^{k} \frac{1}{k} = 1$, there exists a k_0 that for all $i > k_0, w_i^k > \frac{1}{k}$.

Then we have:

$$\begin{aligned}
0 &\leqslant \sum_{i=k_0}^{k} (w_i^k - \frac{1}{k})(\gamma \mathrm{Vol}(\mathcal{S})\xi_{\rho_i} + 2\xi_{\pi_i}) \\
&\leqslant \sum_{i=k_0}^{k} (w_i^k - \frac{1}{k})(\gamma \mathrm{Vol}(\mathcal{S})\xi_{\rho_{k_1}} + 2\xi_{\pi_{k_1}}),
\end{aligned} \tag{19}$$

where $k_1 \in [k_0, k]$

$$\begin{aligned}
0 &\geqslant \sum_{i=1}^{k_0-1} (w_i^k - \frac{1}{k})(\gamma \mathrm{Vol}(\mathcal{S})\xi_{\rho_{k_2}} + 2\xi_{\pi_{k_2}}) \\
&\geqslant \sum_{i=1}^{k_0-1} (w_i^k - \frac{1}{k})(\gamma \mathrm{Vol}(\mathcal{S})\xi_{\rho_i} + 2\xi_{\pi_i}),
\end{aligned} \tag{20}$$

where $k_2 \in [0, k_0)$

Based on these two equations:

$$\begin{aligned}
&\sum_{i=1}^{k}(w_i^k - \frac{1}{k})(\gamma \mathrm{Vol}(\mathcal{S})\xi_{\rho_i} + 2\xi_{\pi_i}) \\
&= \sum_{i=1}^{k_0-1} (w_i^k - \frac{1}{k})(\gamma \mathrm{Vol}(\mathcal{S})\xi_{\rho_i} + 2\xi_{\pi_i}) + \sum_{i=k_0}^{k} (w_i^k - \frac{1}{k})(\gamma \mathrm{Vol}(\mathcal{S})\xi_{\rho_i} + 2\xi_{\pi_i}) \\
&\leqslant \sum_{i=1}^{k_0-1} (w_i^k - \frac{1}{k})(\gamma \mathrm{Vol}(\mathcal{S})\xi_{\rho_{k_2}} + 2\xi_{\pi_{k_2}}) + \sum_{i=k_0}^{k} (w_i^k - \frac{1}{k})(\gamma \mathrm{Vol}(\mathcal{S})\xi_{\rho_{k_1}} + 2\xi_{\pi_{k_1}}) \\
&= \sum_{i=1}^{k_0-1} (w_i^k - \frac{1}{k})(\gamma \mathrm{Vol}(\mathcal{S})\xi_{\rho_{k_2}} + 2\xi_{\pi_{k_2}}) - \sum_{i=1}^{k_0-1} (w_i^k - \frac{1}{k})(\gamma \mathrm{Vol}(\mathcal{S})\xi_{\rho_{k_1}} + 2\xi_{\pi_{k_1}}) \\
&\quad + \sum_{i=1}^{k_0-1} (w_i^k - \frac{1}{k})(\gamma \mathrm{Vol}(\mathcal{S})\xi_{\rho_{k_1}} + 2\xi_{\pi_{k_1}}) + \sum_{i=k_0}^{k} (w_i^k - \frac{1}{k})(\gamma \mathrm{Vol}(\mathcal{S})\xi_{\rho_{k_1}} + 2\xi_{\pi_{k_1}}) \\
&= \sum_{i=1}^{k_0-1} (w_i^k - \frac{1}{k})[(\gamma \mathrm{Vol}(\mathcal{S})\xi_{\rho_{k_2}} + 2\xi_{\pi_{k_2}}) - (\gamma \mathrm{Vol}(\mathcal{S})\xi_{\rho_{k_1}} + 2\xi_{\pi_{k_1}})] + \sum_{i=1}^{k}(w_i^k - \frac{1}{k})(\gamma \mathrm{Vol}(\mathcal{S})\xi_{\rho_{k_1}} + 2\xi_{\pi_{k_1}})
\end{aligned} \tag{21}$$

Since distribution shift sequence $\{(\xi_{\rho_1}, \xi_{\pi_1}), (\xi_{\rho_2}, \xi_{\pi_2}), ..., (\xi_{\rho_k}, \xi_{\pi_k})\}$ decreases as i increases, and $k_2 < k_1$, the first term will be less than 0. Meanwhile, the second term will be equal to 0 because $\sum\limits_{i=1}^{k} w_i^k = \sum\limits_{i=1}^{k} \frac{1}{k} = 1$. Therefore, we can get:

$$\begin{aligned}
&\sum_{i=1}^{k}(w_i^k - \frac{1}{k})(\gamma \mathrm{Vol}(\mathcal{S})\xi_{\rho_i} + 2\xi_{\pi_i}) \\
&\leqslant \sum_{i=1}^{k_0-1}(w_i^k - \frac{1}{k})[(\gamma \mathrm{Vol}(\mathcal{S})\xi_{\rho_{k_2}} + 2\xi_{\pi_{k_2}}) - (\gamma \mathrm{Vol}(\mathcal{S})\xi_{\rho_{k_1}} + 2\xi_{\pi_{k_1}})] + \sum_{i=1}^{k}(w_i^k - \frac{1}{k})(\gamma \mathrm{Vol}(\mathcal{S})\xi_{\rho_{k_1}} + 2\xi_{\pi_{k_1}}) \\
&\leqslant 0
\end{aligned} \tag{22}$$

The proof is finished. $\square$

Proposition 3.2 illustrate that after adjusting the policy mixture distribution according to the distribution shifts, the performance bound will be tighter than learning a global dynamics model ($w_i^k = \frac{1}{k}$). This provides a guidance for our proposed method, that the weight w_i^k of each policy π_i in the historical policy sequence Π^k should be negatively related to its state action visitation distribution shift ξ_{ρ_i} and the policy distribution shift ξ_{π_i}.

F More experiments

F.1 More Error Curves for Dynamics Model Learned by MBPO

In this section, we provide the local error curves for global dynamics model in four MoJoCo environments: Hopper, HalfCheetah, Walker2d, and Humanoid. The curves are shown in Figure 5.

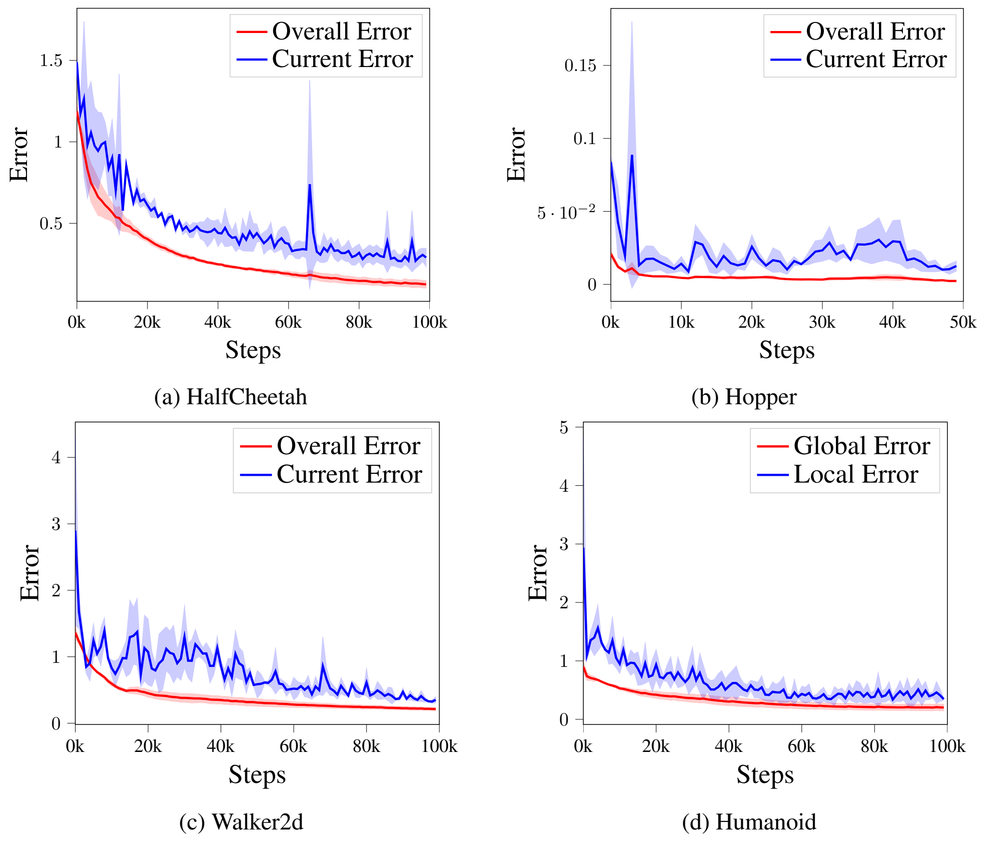
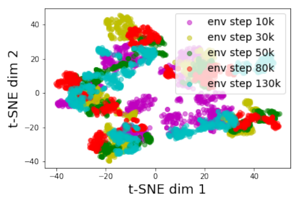


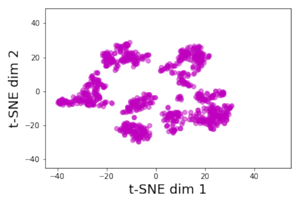
Figure 5: The global error curve and the local error curve of MBPO in four MuJoCo environments.

F.2 Visualization of State-Action Visitation Distribution of Different Historical Policies

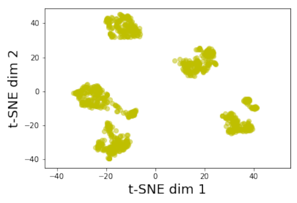
Due to the limited space of main paper, we provide detailed visualization of the state-action visitation distribution of policies under different environment steps in this section. We conduct the experiment on HalfCheetah and Hopper, the results are shown in Figure 6 and Figure 7. (a) in each figure is the comparison of different policies in the same figure, from (b) to (f) are the figures presenting the state-action visitation distribution of each policy individually. We can see that the state-action visitation distribution of policies under different environment steps is very different.



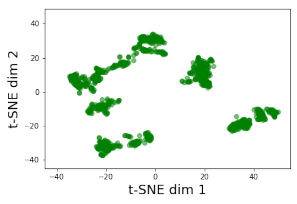
(a)



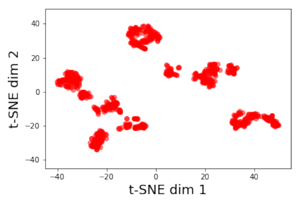
(b) Environment step 10k



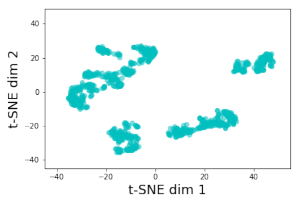
(c) Environment step 30k



(d) Environment step 50k

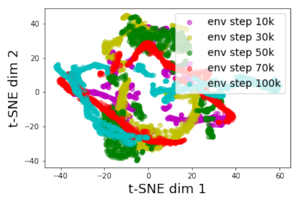


(e) Environment step 80k

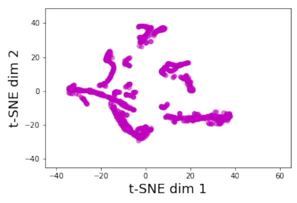


(f) Environment step 130k

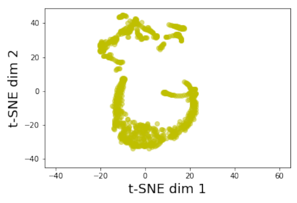
Figure 6: Visualization of state-action visitation distribution of policies at different environment steps in HalfCheetah.



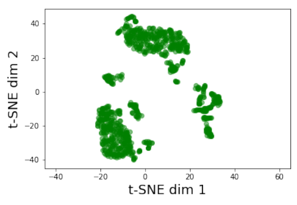
(a)



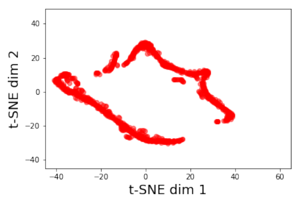
(b) Environment step 10k



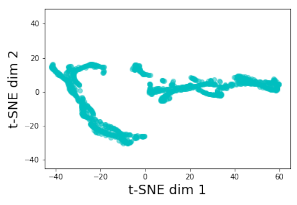
(c) Environment step 30k



(d) Environment step 50k



(e) Environment step 70k

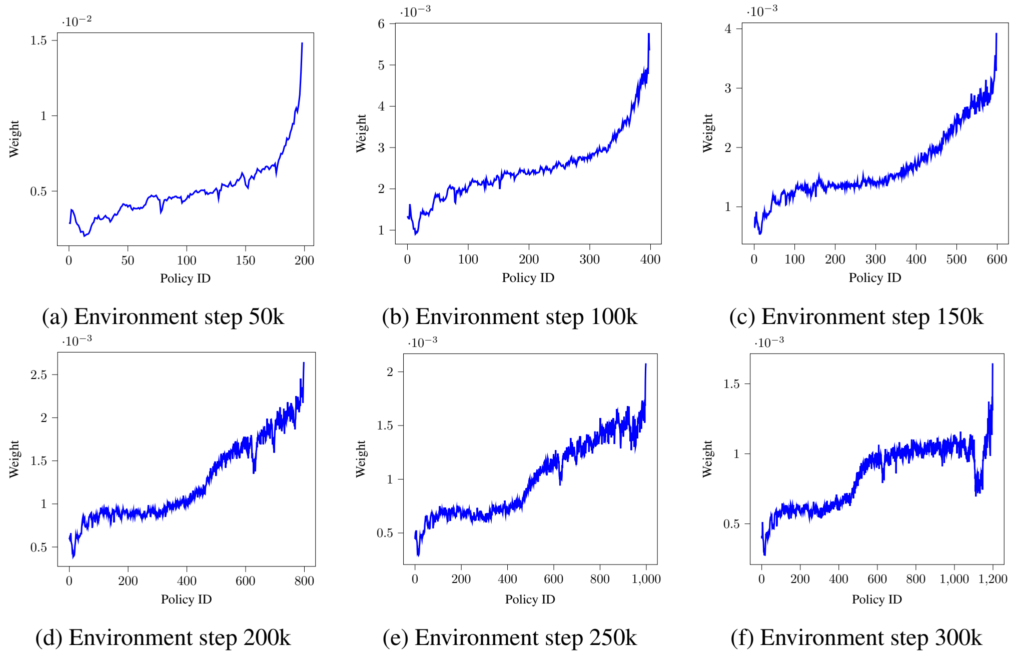


(f) Environment step 100k

Figure 7: Visualization of state-action visitation distribution of policies at different environment steps in Hopper.

F.3 Visualization of Adjusted Policy Mixture Distribution

To provide a further understanding of out method, we visualize the adjusted policy mixture distribution at different training steps on Humanoid in Figure 8. We take Figure 8a as an example to explain the origin and meaning of the policy ID on the horizontal axis. Each of policies interacts with the environment for 250 steps, so a 50k environment step has 200 historical policies. The policy ID of 0 indicates the oldest policy. The larger the ID, the newer the policy. We can see that the policy mixture distribution is totally different at different training steps. The weight of policy is not a simple exponentially decay or linearly decay, which indicates our proposed method is non-trivial.



(a) Environment step 50k (b) Environment step 100k (c) Environment step 150k

(d) Environment step 200k (e) Environment step 250k (f) Environment step 300k

Figure 8: Visualization of adjusted policy mixture distribution at different training steps on Humanoid.

F.4 More Benchmark Results

Due to the limited space, we put more results on MuJoCo in this section. Compared with Figure 2, we add the results on Ant and Pusher in Figure 9. In Ant and Pusher, our method achieves a significant performance improvement compared with original MBPO. Besides, our method also achieves comparable or better results than previous sample efficient model-based or model-free methods.

F.5 Comparison with Simple Exponentially Decay Prioritization

To further demonstrate the effectiveness of our method, we compare with an exponentially decay method. The weight of the historical policy exponentially decays as it lifetime increases. To ensure a fair comparison, the weight of current policy is also compute using Eq. 5. The hyperparameter α of exponentially decay method is the same as PDML which is given in Appendix G.1. We conduct the experiment on three MoJoCo environments: Hopper, Walker2d, and Humanoid. The experiment results are given in Figure 10. We can see that after using exponentially decay method, the performance in three environments is slightly improved, but it is much lower than PDML. Besides, the model error of exponentially decay method is higher than PDML. Combined with the analysis of distribution visualization in Appendix F.3, this further demonstrates that our method is non-trivial and effective.

F.6 Ablation Study of PDML

As we described in Section 4, we use the adjusted policy mixture distribution for both model learning and sampling initial states for model rollouts. In this section, we provide the ablation study to show the impact of the adjusted policy mixture distribution in these two parts respectively. We conducted our experiments in Hopper and Walker2d, and the performance curves are shown in Figures 11a and 11b.

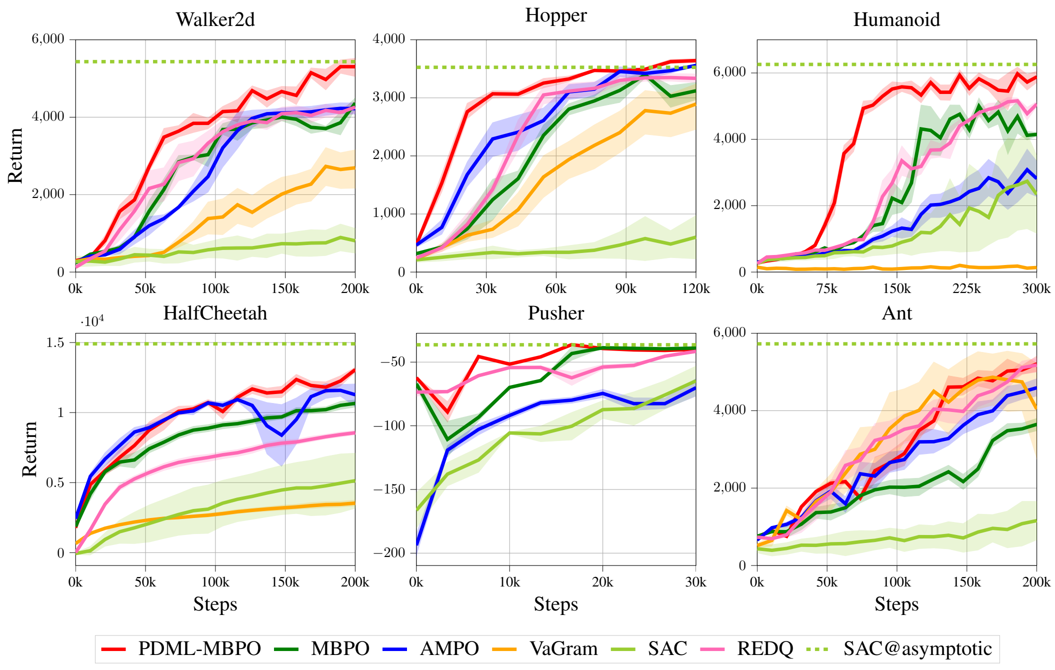
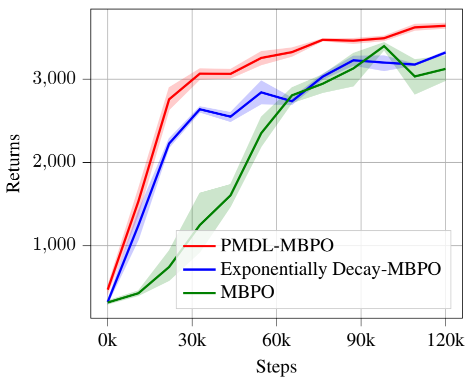
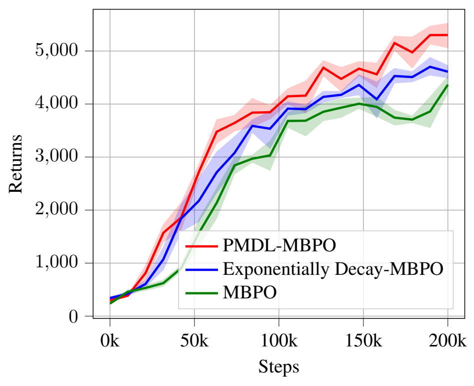


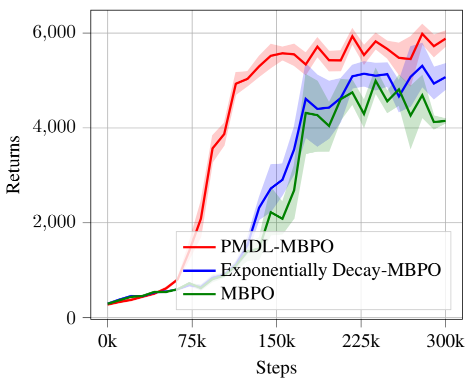
Figure 9: Benchmark comparison on six MuJoCo environments.



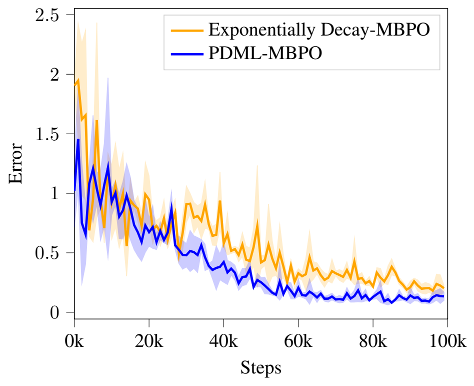
(a) Hopper



(b) Walker2d

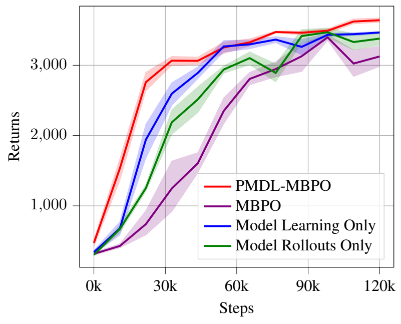


(c) Humanoid

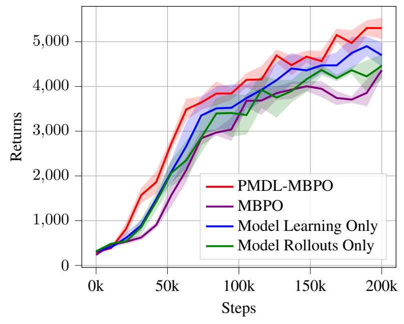


(d) Walker2d one-step error

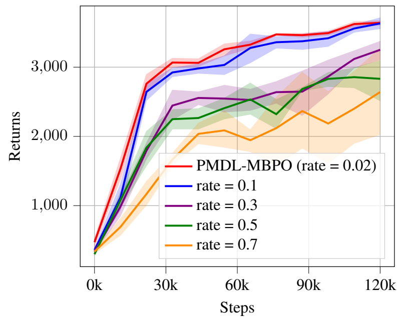
Figure 10: Comparison with exponentially decay prioritization.



(a) Hopper



(b) Walker2d



(c) Hopper

Figure 11: (a) and (b): Ablation study of adjusted policy mixture distribution on model learning and sampling initial states for model rollouts. (c): Ablation study of current policy proportion rate.

We find that using the adjusted policy mixture distribution only for model learning or model rollouts initial states sampling both improves the performance in Hopper and Walker2d compared to MBPO. However, the improvement of only using the adjusted policy mixture distribution for model rollouts initial states sampling in Walker2d is not very significant. Besides, the improvement of using the adjusted policy mixture distribution for model learning is better than using that for model rollouts initial states sampling, but both of them are worse than PDML. This indicates two things. First, model learning is more important than model rollouts initial states sampling, because even the initial state distribution obeys the state-action visitation distribution of current policy, the model-generated samples will still be inaccurate if the learned dynamics model is not accurate enough for the current policy. Second, to achieve the best performance, sample distribution for model learning and sample distribution for model rollouts initial states should be synergistic; that is, the training data for training the dynamics model and the initial states of model rollouts should obey the same distribution, so that the model prediction error can be minimized.

F.7 Ablation Study of Current Policy Proportion Rate

We conducted experiments to explore the impact of current policy proportion rate on the performance of our method. The α in Eq. 5 equals to current policy proportion rate divided by 1 minus current policy proportion rate. As shown in Figure 11c, when the current policy proportion rate is small (0.02 and 0.1), the policy mixture distribution will not be too inclined to the current policy, so the model can learn a good transition dynamics. When the current policy proportion rate is too large (0.3, 0.5, and 0.7), the learned dynamics capture information about the underlying transition too locally, resulting in performance decrease. Therefore, we recommend that the selection of the current policy proportion rate should not be greater than 0.1.

F.8 Training Time of PDML-MBPO and MBPO

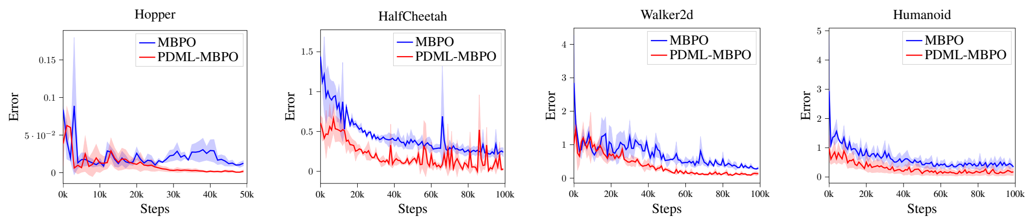
In this section, we present the training time of PDML-MBPO and MBPO in four different environments. As shown in Table 1, after using PDML, the training time doesn't increase significantly. In the most complex environment Humanoid, the training time for 300k steps increases by only one hour. In other environments, the training time of PDML-MBPO is almost the same as that of MBPO.

Table 1: Training time of PDML-MBPO and MBPO in different environments. The results are averaged over 8 random seeds.

	Walker2d	Hopper	Humanoid	HalfCheetah
MBPO	58.6 h	35.5 h	70.8 h	60.2 h
PDML-MBPO	59.2 h	35.7 h	72.0 h	60.9 h

F.9 One-step Error in Four Environments

As an extension of Section 5.3, we provide the one-step model prediction error curve in this section. The results are shown in Figure 12.



(a) (b) (c) (d)

Figure 12: One-step error curves in Hopper, Walker2d, HalfCheetah, and Humanoid.

G Implementation

G.1 Implementation Details

We implement PDML-MBPO based on the PyTorch-version MBPO Liu et al. (2020). We also set the ensemble size of PDML-MBPO to be the same as MBPO, which is 7. The warm-up samples are collected through interaction with the real environment for 5000 steps using a randomly chosen policy. After the warm-up, we train the dynamics model and update the lifetime weight every 250 interaction steps. We set the current policy proportion to be 0.02 and α equals $0.02/0.98$. One thing that needs to be noticed is the rollout horizon setting. As introduced in MBPO Janner et al. (2019), the rollout horizon should start at a short horizon and increase linearly with the interaction epoch. $[a, b, x, y]$ denotes a thresholded linear function, $i.e.$ at epoch e, rollout horizon is $h = \min(\max(x + \frac{e-a}{b-a}(y - x), x), y)$. We set the rollout horizon to be the same as used in the MBPO paper, as shown in Table 2. Other hyper-parameter settings are shown in Table 3. For MBPO[1], AMPO[2], VaGraM[3] SAC[4], and REDQ[5], we use their open source implementations. We evaluate PDML-MBPO and other baselines on four MuJoCo-v2 continuous control environments Todorov et al. (2012) with a maximum horizon of 1000, including HalfCheetah, Hopper, Walker2d, and Humanoid. For Humanoid, we use the modified version introduced by MBPO Janner et al. (2019). All experiments are conducted using a single NVIDIA TITAN X Pascal GPU.

For the experiment of MaPER in Sec 5.2, we use their open-source code in the supplementary material on openreview [6]. However, we find a bug in their code that comes from the PyTorch-version MBPO implementation, i.e., the same environment is used for policy training and policy evaluation. This

[1]https://github.com/Xingyu-Lin/mbpo_pytorch
[2]https://github.com/RockySJ/ampo
[3]https://github.com/pairlab/vagram
[4]https://github.com/pranz24/pytorch-soft-actor-critic
[5]https://github.com/watchernyu/REDQ
[6]https://openreview.net/forum?id=WuEiafqdy9H

Table 2: Rollout horizon settings for PDML

Walker2d	Hopper	Humanoid	HalfCheetah
1	[1, 15, 20, 100]	[1, 25, 20, 300]	1

Table 3: Hyper-parameter settings for PDML

Parameter	Value
Dynamics model ensemble size	7
Dynamics model layers	4
Actor and critic layers	3
Dynamics model hidden units	200
Actor and critic hidden units	256
Learning rate	$3 \cdot 10^{-4}$
Batch size	256
Optimizer	Adam
Activation function	ReLU
Real sample buffer size	10^6
Model sample buffer size	10^6
Real sample ratio	0.05
Policy updates per environment step	20
Environment steps between model training	250

causes the rollout length to exceed the 1000-step limit during evaluation, resulting in the performance of the policy being much higher than the 1000-step performance. We fix this bug and conduct the experiment, so the results of MaPER are lower than those reported in their paper.