Live in the Moment: Learning Dynamics Model Adapted to Evolving Policy

Xiyao Wang [1] **Wichayaporn Wongkamjan** [1] **Furong Huang** [1]

Abstract

Model-based reinforcement learning (RL) 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 "global" dynamics model to fit the state-action visitation distribution for all historical policies. However, in this paper, we find that learning a global dynamics model does not necessarily benefit model prediction for the current policy since the policy in use is constantly evolving. The evolving policy during training will cause state-action visitation distribution shifts. We theoretically analyze how the distribution of historical policies affects the model learning and model rollouts. We then propose a novel model-based RL method, named *Policy-adaptation Model-based Actor-Critic (PMAC)*, which learns a policy-adapted dynamics model based on a policy-adaptation mechanism. This mechanism 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 PMAC achieves state-of-the-art asymptotic performance and almost two times higher sample efficiency than prior model-based 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 very 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 to train, 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, prior works have proposed various approaches to solve the problem. 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. To reduce the distribution mismatch between model-generated samples and real samples, 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). Instead of accurately predicting the transitions, the value equivalence model (Farahmand et al., 2017; Grimm et al., 2020; Voelcker et al., 2022) is proposed to generate the samples that can yield the same Bellman updates like the real samples. All previous works aim to learn a "global" dynamics model. 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 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, learning such a global dynamics model has an

[1]Department of Computer Science, University of Maryland, College Park, MD 20742, USA. Correspondence to: Xiyao Wang <xywang@umd.edu>.

Decision Awareness in Reinforcement Learning Workshop at the 39th International Conference on Machine Learning (ICML), Baltimore, Maryland, USA, 2022.

objective mismatch with model rollouts, since we only use the current newest policy during model rollouts to generate samples. In many real-world tasks, our goal is to identify a good policy as efficiently as possible, and the newest policy is believed to be the best we can achieve so far. Therefore, the learned dynamics model only needs to make accurate predictions for the current policy. Learning a global dynamics model may have a relatively good prediction for all historical policies, but unfortunately, it may have a high prediction error for current policy during model rollouts.

In this paper, we investigate how to learn an accurate dynamics model for model rollouts based on existing samples. To begin with, we confirm through experiments that although the global dynamics model learned by the previous methods has a low overall prediction error for all historical policies, its prediction error for the current newest policy is still very high. This leads to inaccurate model-generated samples and reduces the sample efficiency and asymptotic performance of the policy. We then theoretically analyze how the distribution of historical policies affects model learning and model rollouts. The theoretical result suggests that the distribution of historical policies used for model learning should always tilt to the newer policies rather than uniformly distribute to any policy to ensure the model prediction accuracy for model rollouts. Motivated by this insight, we propose a novel model-based RL method named *policy-adaptation model-based actor-critic (PMAC)*. Instead of learning a global dynamics model, PMAC learns a policy-adapted dynamics model based on a policy-adaptation mechanism, which adapts the learned dynamics model to the evolving policy to ensure the accuracy of the model-generated samples. We conduct systematic and extensive experiments to evaluate PMAC on a range of continuous control benchmark MuJoCo environments (Todorov et al., 2012). Experiment results show that PMAC achieves higher sample efficiency and better asymptotic performance than previous state-of-the-art model-based RL methods.

Summary of contributions: **(1)** Through detailed experimental results, we establish that learning a global dynamics model is not accurate enough for model rollouts. **(2)** We theoretically analyze how the distribution over historical policies affects model learning and model rollouts and how the dynamics model should be learned. **(3)** We propose *policy-adaptation model-based actor-critic (PMAC)*, which implements a policy adaptation mechanism that allows the model to continuously adapt to the evolving policy. **(4)** Experimental results on a range of MuJoCo environments demonstrate that PMAC achieves state-of-the-art asymptotic performance and a drastic sample efficiency improvement (more than $2\times$) compared with previous 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:

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

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. We concentrate on the Dyna-style (Sutton, 1990) model-based RL in this paper, which uses the learned dynamics model to generate samples and train the policy. Model-based RL in this paper refers to Dyna-style model-based RL.

Policy mixture During policy learning, we consider the historical policies at iteration step k as a historical policy sequence $\Pi^k = \{\pi_0, \pi_1, \pi_2, ..., \pi_k\}$. 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 w^k. Then the state-action visitation distribution of the policy mixture $\pi_{mix,k} = (\Pi^k, w^k)$ is $\rho^{\pi_{mix,k}}(s,a) = \sum_{i=0}^{k} w_i^k \rho^{\pi_i}(s,a)$ (Hazan et al., 2019; Zhang et al., 2021).

2.2. Global Dynamics Model in Model-based RL

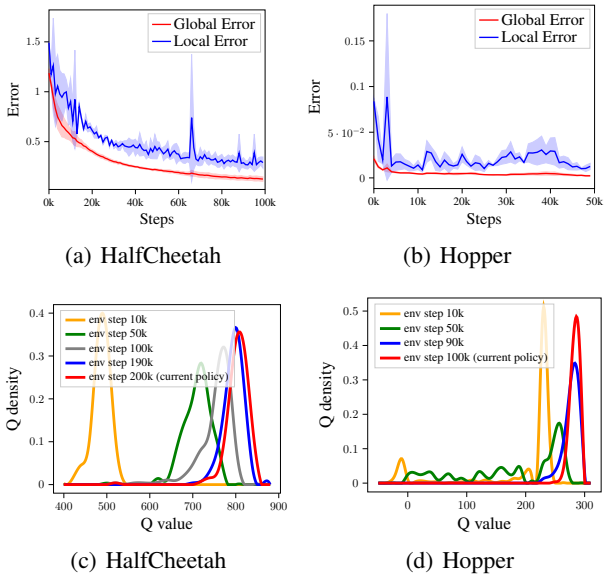
Learning a dynamics model is the most crucial part of model-based RL since the ground-truth transition dynamics is unknown and the policy must be updated based on the samples generated by the learned dynamics model. Previous works learn the dynamics model by randomly selecting training data from the 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:

$$\rho^{\pi_{mix,k}}(s,a) = \sum_{i=0}^{k} \frac{1}{k} \rho^{\pi_i}(s,a). \quad (2)$$

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

However, it is difficult to learn such a global dynamics model in many real-world tasks since the number of real samples from the environment is very limited and obtaining real samples is expensive. Meanwhile, the state-action visitation density of historical policies may have a huge shift from the current policy, which is used for model rollouts. Suppose the dynamics model adapts to the state-action visitation distribution of all historical policies. In that case, this may hurt the accuracy of the model prediction for the current policy during model rollouts, resulting in the policy getting stuck in a sub-optimal solution or even being unable to converge.

We conduct an experiment using a state-of-the-art model-based RL baseline called MBPO (Janner et al., 2019), which learns a global dynamics model on two MuJoCo (Todorov et al., 2012) environments Hopper and HalfCheetah. We present the global error curves and the local error curves during learning steps in Figure 1(a) and 1(b). Here the global error means the model prediction error for all historical policies, and the local error is the model prediction error for the current policy. We can find that although the global error has converged to be very small, the local error for the current policy is still very high, which will undoubtedly cause inaccurate model-generated samples and misleading the policy learning. Therefore, learning a global dynamics model is not an efficient way for model-based RL (especially for task-specific problems such as robotics).



(a) HalfCheetah (b) Hopper

(c) HalfCheetah (d) Hopper

Figure 1: (a) and (b) are the global error curve and the local error curve of MBPO on HalfCheetah and Hopper, respectively. (c) and (d) are the state-action visitation distribution of different history policies and the current policy expressed using the Q value.

3. Dynamics Model Adapted to the Evolving Policy

To ensure the accuracy of model-generated samples, we present a dynamics model which aims to adapt the state-action visitation density of the current policy. To motivate our method, we first analyze how the distribution of policy mixture affects the model rollout performance theoretically. We then analyze the state-action visitation distribution of the historical policies to guide us on how to adjust the distribution of policy mixture and learn the policy-adapted dynamics model.

3.1. Performance guaranteed by policy mixture distribution

As mentioned in Section 2.2, learning a dynamics model that adapts the state-action visitation distribution for all historical policies is not efficient for model-based RL. In this section, we provide a theoretical analysis of how the policy mixture distribution affects the model rollout.

First, we derive our main theorem that gives an upper bound for the performance gap between the real environment rollout and the model rollout using the current policy π.

Theorem 3.1 *Given the historical policy mixture* $\pi_{mix,k} = (\Pi^k, w^k)$ *at iteration step* k*, we denote* $\xi_{\rho_i} = D_{TV}(\rho^{\pi}_{T}(s,a)||\rho^{\pi_i}_{T}(s,a))$ *and* $\xi_{\pi_i} = \mathbb{E}_{s\sim v^{\pi_{mix}}_{\hat{T}}}[D_{TV}(\pi(a|s)||\pi_i(a|s))]$ *as the state-action visitation distribution shift and the policy distribution shift between the historical policy* π_i *and current policy* π *respectively, where* $v^{\pi_{mix}}_{\hat{T}}$ *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:*

$$
\begin{aligned}
&J(\pi,T) - J(\pi,\hat{T}) \\
\leq& 2\gamma r_{max}\mathbb{E}_{(s,a)\sim\rho^{\pi}_{T}}[D_{TV}(T(s'|s,a)||\hat{T}(s'|s,a))] \\
&+ r_{max}\sum_{i=0}^{k} w_i^k(\gamma \mathrm{Vol}(\mathcal{S})\xi_{\rho_i} + 2\xi_{\pi_i}) \\
&+ 2r_{max}D_{TV}(\rho^{\pi_{mix}}_{\hat{T}}(s,a)||\rho^{\pi}_{\hat{T}}(s,a))
\end{aligned}
\tag{3}
$$

Proof. See Appendix. B □

Remarks. We have several remarks according to this bound. **(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 the current policy induces the model rollout. **(2)** For the second term, it shows the effect of

the policy mixture distribution on model rollout. This item contains two distribution shifts: state-action visitation distribution shift ξ_{ρ_i} and policy distribution shift ξ_{π_i} between historical policy and current policy. It should be noted that ξ_{ρ_i} is induced by ξ_{π_i}, so it is reasonable to think 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 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 policy learning. To maximize sample utilization, the model-generated samples obtained by the historical policies will be used to update the current policy. Therefore, we need to adjust the model-generated sample distribution of historical policies to be close to the model-generated sample distribution of the current policy during policy learning. This 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 Eq. 3 both point out that learning a global dynamics model is not the most efficient way for model learning. The first term states that model learning should adapt as much as possible to the state-action visitation distribution of the current policy. The second term proposes that the state-action visitation distribution in the training data should be closer to the current policy by adjusting the policy mixture distribution. In the next subsection, we present a specific method for adjusting the policy mixture distribution to learn a policy-adapted dynamics model.

3.2. Policy-Adaptation Dynamics Model

This section provides a policy-adaptation mechanism to adjust the policy mixture distribution for model learning. To motivate our method, we first visualize the state-action visitation distribution of different historical policies derived from different environmental steps. We conduct this experiment using MBPO. For each historical policy, we use the policy to interact with the real environment to obtain a batch of samples, then use the critic network of a converged SAC (Haarnoja et al., 2018) (the policy part of MBPO) to estimate the Q value of these samples and obtain a state-action visitation distribution expressed by Q value. As shown in Figure 1(c) and 1(d), the policy is updated as the number of environmental steps increases. As a result, the historical policy that is closer to the current policy in the environmental steps has a smaller deviation from the state-action visitation distribution of the current policy. This phenomenon is also consistent with the goal of reinforcement learning; that is, although there may be fluctuations in the process of policy updates, the overall trend should constantly be improving.

Algorithm 1 Policy-adaptation Mechanism

1: **Input**: Historical policy sequence Π^k, current policy π, policy mixture distribution w^k, policy-adaptation rate α
2: **for** all historical policies in Π^k **do**
3: Multiply the weight w_i^k of each historical policy by α
4: **end for**
5: Calculate the weight of current policy using Eq. (4)
6: Add current policy to historical policy sequence Π^k
7: **Output**: Historical policy sequence Π^{k+1}, new policy mixture distribution w^{k+1}

Therefore, we propose a policy-adaptation mechanism which adjusts the policy mixture distribution that continuously adapts to the current evolving policy. The model learned based on the state-action visitation distribution of this policy mixture is a policy-adaptation dynamics model. The policy-adaptation mechanism is an iterative updating process that updates the policy mixture distribution as follows, and the pseudocode is shown in Algorithm 1.
(1) Before an interaction epoch, there are k policies existed in the historical policy sequence Π^k, where each policy π_i has a weight from the previous round, i.e., $\tilde{w}_i^k, \forall i \in [k]$.
(2) We update the weights of the k existing policies $w_i^{k+1} \leftarrow \alpha\tilde{w}_i^k, \forall i \in [k]$, where $\alpha < 1$ is a hyper-parameter called *policy adaptation rate*.
(3) During the interaction epoch, we get a small batch of new samples from the real environment using the current policy and assign weights for the current policy,

$$w_{k+1}^{k+1} = \frac{(1-\alpha)\sum_{i=1}^{k} w_i^{k+1}}{\alpha} \tag{4}$$

(4) The process goes to the next round: $\tilde{w}_i^{k+1} \leftarrow w_i^{k+1}$, $\forall i \in [k+1]$ and the total number of historical policy increases $k \leftarrow k+1$. Go back to **(1)**.

Remarks. **(1)** At each round, the total weights for historical policies $W_{\mathsf{old}} = \sum_{i=1}^{k} w_i^{k+1} = \alpha \sum_{i=1}^{k} \tilde{w}_i^k$ and the weight for the current policy $W_{\mathsf{new}} = w_{k+1}^{k+1} = (1-\alpha)\sum_{i=1}^{k} \tilde{w}_i^k$. Therefore, the weight ratio between historical policies and current policy is $\frac{W_{\mathsf{old}}}{W_{\mathsf{new}}} = \frac{\alpha}{1-\alpha}$. As the iteration step of policies increases, the weights of historical policies gradually decrease since $\alpha < 1$. The current newest policy always accounts for $(1-\alpha)$ portion of the total weights, getting significant attention during model learning. This enables the state-action visitation distribution induced by the historical policies always tilt toward the current policy's state-action visitation distribution. **(2)** Since the collection of the weights of historical policies remains to be a probability vector, our proposed policy-adaptation mechanism guarantees that the sum of all weights maintains constant in every round, as it can be easily verified that $\sum_{i=1}^{k+1} w_i^{k+1} = \sum_{i=1}^{k} \tilde{w}_i^k$. We

initialize weights, to sum up to 1. **(3)** According to Theorem 3.1, we should have only the current policy in the policy mixture; that is, the weights of other historical policies are set to 0. However, the number of interactions with the environment in model-based RL is very limited. We can only get a few samples belonging to the current state-action visitation distribution. It is challenging to learn the real transition dynamics of the environment based on these few samples. Therefore, we design this weight adaptation method to use the samples obtained from the historical policies closest to the current policy to make up for the shortcoming of the scarcity of training samples.

4. Policy-adaptation Model-based Actor-Critic

In this section, we introduce the practical algorithm called Policy-adaptation Model-based Actor-Critic (PMAC) which uses a policy-adaptation mechanism for model learning. The full algorithm is in Algorithm 2.

Algorithm 2 Policy-adaptation Model-based Actor-Critic

Require: Dynamics model $\hat{T}_\theta$, actor π_ϕ, critic Q_ψ, real sample buffer $\mathbb{D}_e$, model sample buffer $\mathbb{D}_m$, historical policy sequence Π^k, policy mixture distribution w^k, policy adaptation rate α, interaction epochs I, policy update steps E, number of rollout trajectories K, rollout horizon H

for I epochs **do**
  Update the historical policy mixture distribution w^k using Algorithm 1
  Sample a batch of (s_t, a_t, r, s_{t+1}) from $\mathbb{D}_e$ according to w^k
  Train dynamics model $\hat{T}_\theta$ via Eq. (5)
  for E steps **do**
    Obtain samples in the real environment using π_ϕ, add to $\mathbb{D}_e$
    for K model rollouts **do**
      Sample s_t as an initial state from $\mathbb{D}_e$ according to w^k
      Rollout H steps using dynamics model $\hat{T}_\theta$ and actor π_ϕ, add samples to $\mathbb{D}_m$
    end for
    Update actor π_ϕ and critic Q_ψ on $\mathbb{D}_m$ via Eq. (6)
  end for
end for

Model Learning. The probabilistic neural network ensemble is believed to better capture model uncertainty (Chua et al., 2018) and has shown to be effective in model learning (Janner et al., 2019; Wang & Ba, 2019; Shen et al., 2020). Thus, we use the probabilistic neural network ensemble $\hat{T}_\theta = \{\hat{T}_{\theta_1}, \hat{T}_{\theta_2}, ..., \hat{T}_{\theta_B}\}$ as our dynamics model, where B is the number of neural networks in the ensemble. Given a (s_t, a_t) 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: $\hat{T}_{\theta_b}(s_{t+1}|s_t, a_t) = \mathcal{N}(\mu_{\theta_b}(s_t, a_t), \Sigma_{\theta_b}(s_t, a_t))$, and the next state is sampled from this Multivariate Gaussian Distribution. Moreover, the parameters of neural network models in the ensemble are initialized differently. During model learning, we select the training samples according to the policy mixture distribution w^k from the real sample buffer and use the maximum likelihood to improve models:

$$\mathcal{L}(\theta_b) = \sum_{t=1}^{N} [\mu_{\theta_b}(s_t, a_t) - s_{t+1}]^\top \Sigma_{\theta_b}^{-1}(s_t, a_t)[\mu_{\theta_b}(s_t, a_t) - s_{t+1}] + \log\det \Sigma_{\theta_b}(s_t, a_t) \tag{5}$$

Model Rollouts. During model rollouts, we also sample the initial states from the real sample buffer according to the policy mixture distribution w^k. At each rollout step, we randomly select a neural network from the ensemble to predict the next state and add the model-generated sample into the model sample buffer $\mathbb{D}_m$. After getting enough samples, we then use the model sample buffer to train the policy. For the rollout horizon setting, we adopt the same mechanism as MBPO in which the rollout horizon increases linearly along with the increase of environment steps. The detail is in Appendix D.1.

Policy Learning. We use soft actor-critic (SAC) (Haarnoja et al., 2018), which trains the critic network Q_ψ using the Bellman backup operator and trains a stochastic policy π_ϕ as the actor with entropy regularization by minimizing the expected KL-divergence:

$$\mathcal{L}(\phi) = \mathbb{E}_{s\sim\mathbb{D}_m}[D_{KL}(\pi_\phi(\cdot|s)||\exp(Q_\psi^{\pi_\phi} - V^{\pi_\phi}))]. \tag{6}$$

During policy learning, training samples are a mixture of real and model-generated samples with a mixture weight ρ.

5. Experiment

In this section, we will first compare our PMAC with the previous state-of-the-art (including both model-free and model-based) baselines and demonstrate that PMAC improves SOTA sample efficiency and SOTA asymptotic performance for Model-based RL. Then we conduct systematic ablation study to study the impact of Policy-adaptation Mechanism. Lastly, we compare with three SOTA prioritized experience reply methods, PER (Goyal et al., 2018), RECALL (Schaul et al., 2016) and MaPER (Oh et al., 2022), to indicate the advantage of our policy-adaption mechanism for model learning. Due to the space limitation, we provide more ablation studies in Appendix C including the discussion of policy adaptation rate.

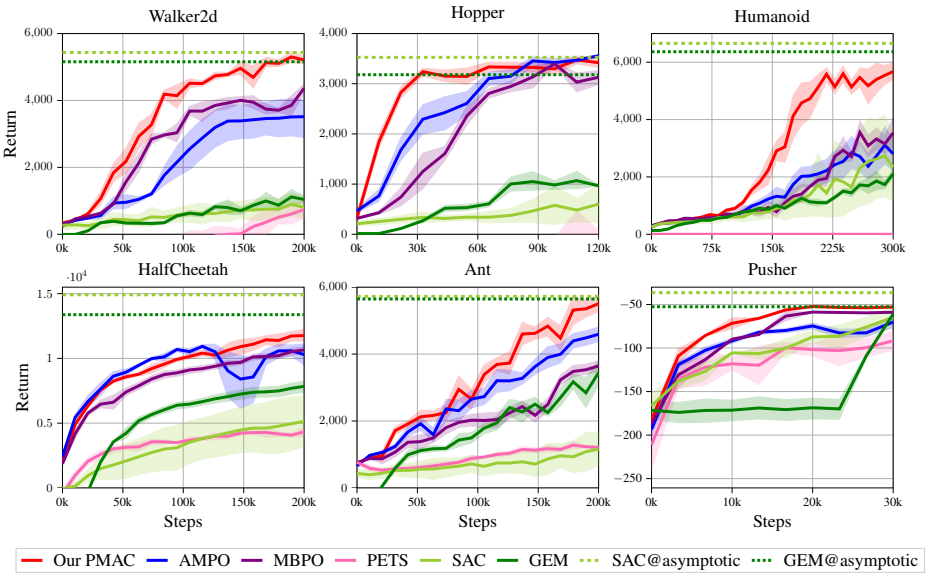


Figure 2: Performance curves for our method (PMAC) and other baseline methods on six MuJoCo environments. Our PMAC, AMPO, MBPO and PETS are model-based methods, while SAC and GEM are model-free methods. The dashed lines indicate the asymptotic performance of SAC and GEM. The solid lines indicate the mean and shaded areas indicate the standard deviation over five random seeds. We evaluate the performance every 1k interaction steps.

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 AMPO (Shen et al., 2020), MBPO (Janner et al., 2019), and PETS (Chua et al., 2018). MBPO is the SOTA model-based method and our method is building upon the architecture of PyTorch-version MBPO (Liu et al., 2020). AMPO is another SOTA model-based method that uses unsupervised model adaptation to reduce the prediction error of the learned dynamics model. PETS is a SOTA model predict control (MPC) method that uses the cross-entropy method (CEM) as the planner, and it also performs well in many continuous control tasks. These three methods use a probabilistic neural network ensemble as the architecture of dynamics model. They all try to fit the state-action visitation distribution of all historical policies to learn a global dynamics model. For model-free methods, we compare with SAC (Haarnoja et al., 2018), which is the policy part of our method and is one of the SOTA model-free methods, and GEM (Hu et al., 2021), which is another recent SOTA method. The implementation details are in Appendix D.1. We conduct experiment on six MoJoCo-v2 (Todorov et al., 2012) environments, the performance curves are shown in Figure 2.

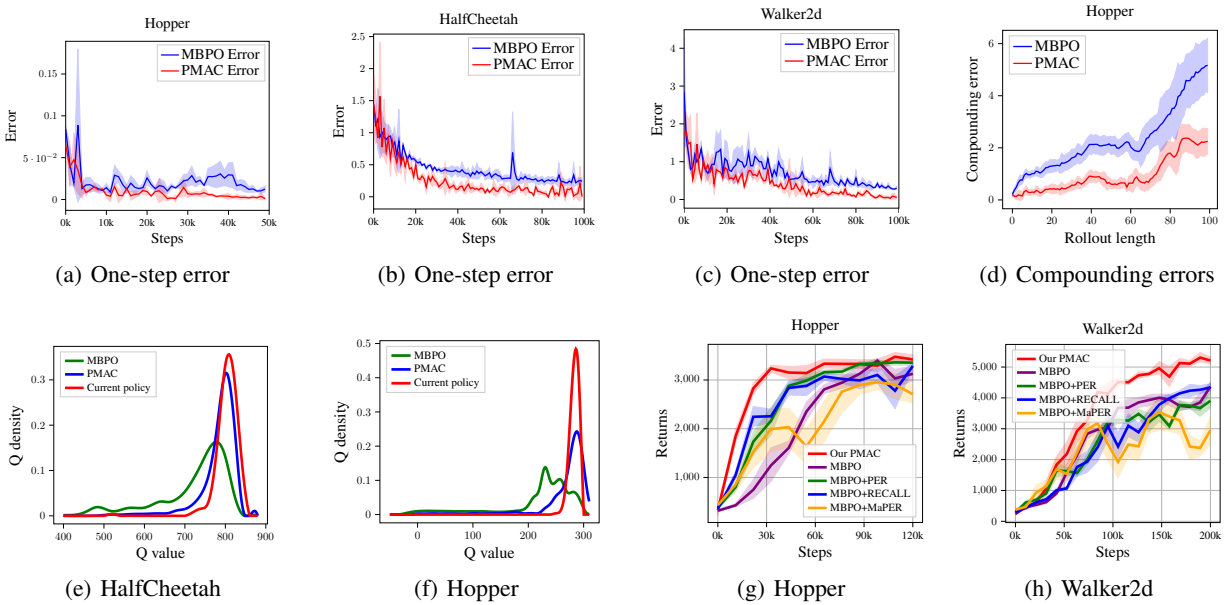
Results: (1) Improving SOTA sample efficiency. PMAC outperforms all existing state-of-the-art methods, including both model-based and model-free methods, in sample efficiency in all six environments. **(2) Improving SOTA asymptotic performance for Model-based RL.** In addition, PMAC obtains significantly better asymptotic performance compared to other state-of-the-art model-based methods, including AMPO, MBPO, and PETS. It is worth noting that the asymptotic performance of PMAC is very close to SAC in four environments (Hopper, Walker2d, Ant, and Pusher) and is even better than SAC occasionally. This indicates the effectiveness of the policy-adaptation mechanism of our proposed PMAC.

5.2. Impact of Policy-adaptation Mechanism

To further verify the impact of the policy-adaptation mechanism, we compare the one-step prediction error, and the compounding error of the policy-adaptation model learned by PMAC and the global dynamics model learned by MBPO. Besides, we also compare the state-action visitation distribution of the historical policy mixture before and after using the policy-adaptation mechanism.

One-step prediction error As shown in Figure 3(a), 3(b) and 3(c), we evaluate the model prediction error for the current policy on Hopper, HalfCheetah, and Walker2d. We evaluate every 1000 environment steps. The error curves show that the one-step prediction error for the current policy of the policy-adaptation model is much smaller than that of the global dynamics model, which means the model-generated samples of PMAC are more accurate than MBPO, so the policy induced by PMAC can perform better.

Compounding error We also compare the multi-step model rollouts compounding error of PMAC and MBPO. This directly determines the accuracy of the model-generated samples in each model rollout trajectory. Figure



(a) One-step error (b) One-step error (c) One-step error (d) Compounding errors

(e) HalfCheetah (f) Hopper (g) Hopper (h) Walker2d

Figure 3: **(a), (b) and (c)** display one-step (model-prediction) error for PMAC 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 PMAC and MBPO over 10 model rollout trajectories. **(e) and (f)** are the the state-action visitation distribution of the historical policy mixture for PMAC and MBPO. **(g) and (h)** show the comparison of prior sample distribution adjustment methods on Hopper and Walker2d. The experiments are run for 5 random seeds.

3(d) is the compounding error curves of PMAC and MBPO 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. The results demonstrate that PMAC has much a smaller compounding error than MBPO, which means the policy-adaptation model has a stronger multi-step planning capability than the global dynamics model.

State-action visitation distribution In Figure 3(e) and 3(f) we plot the state-action visitation distribution of the historical policy mixture before and after using policy-adaptation mechanism. The curve of HalfCheetah is the state-action visitation distribution at 100k environment steps, and Hopper is at 50k environment steps. We can find that the state-action visitation distribution of the historical policy mixture after using the policy-adaptation mechanism (PMAC) is much closer to the current policy's state-action visitation distribution than before using policy-adaptation mechanism (MBPO). This reveals the reason why the policy-adaptation model has a smaller model prediction error than the global dynamics model, and further proves the correctness and effectiveness of our policy-adaptation mechanism.

5.3. Other Sample Distribution Adjustment Methods

We compare with the other three prioritized experience replay methods to indicate the advantage of our policy-adaptation mechanism for model learning. 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 can induce the high-value samples 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. 3(g) and 3(h). Our PMAC outperforms significantly than all three methods on both sample efficiency and asymptotic performance. We believe this is because 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 the current policy. Therefore, the learned model cannot be adapted to the state-action visitation distribution of the current policy, and the model prediction error during model rollouts cannot be reduced. According to our theory, it is more crucial for model learning to adapt to the state-action visitation distribution of the current policy. This experiment result indicates the correctness of our theory and the effectiveness of our

method.

6. Related Work

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 non-parametric 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) is designed to directly predict the transition of an action sequence input, and unsupervised model adaptation (Shen et al., 2020) is used to reduce the potential distribution mismatch between model-generated samples and real samples.

For better model usage, a short-horizon model rollout (Janner et al., 2019) is shown to avoid the model error and improve the quality of model samples. Based on this, a bidirectional model rollout scheme (Lai et al., 2020) is proposed to avoid the model error further. Furthermore, model disagreement is used to decide when to trust the model (Pan et al., 2020) and regularize the model samples (Yu et al., 2020). To make more effective use of the model by exploiting its differentiability, model-augmented actor-critic (Clavera et al., 2020) is proposed to estimate the policy gradient more accurately by back-propagating the gradient through model rollout trajectories. Based on this work, directional derivative projection policy optimization (Li et al., 2022) provides a two-model architecture to estimate the model gradient more accurately. Recently, on-policy corrections (Froehlich et al., 2022) is proposed to use the generalization ability of the model to correct the model-generated samples according to the real samples.

In addition, model-based RL has made good progress in theoretical analysis and experimental tools. A theoretical guarantee of monotone expected reward improvement of model-based RL (Luo et al., 2018) is provided, and model-based RL is shaped as a game-theoretic framework by formulating the optimization of model and policy as a two-player game (Rajeswaran et al., 2020). To save time tuning hyperparameters, an automatic scheduling framework (Lai et al., 2021) is designed. The dynamics model capacity is systematically studied (Abbas et al., 2020) to show how it affects the model-based methods.

7. Conclusion and Discussion

In this paper, we introduce a novel model-based RL method PMAC, which learns a policy-adapted dynamics model based on a policy-adaptation mechanism. 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. PMAC 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

Abbas, Z., Sokota, S., Talvitie, E., and White, M. Selective dyna-style planning under limited model capacity. In *International Conference on Machine Learning*, pp. 1–10. PMLR, 2020.

Asadi, K., Misra, D., Kim, S., and Littman, M. L. Combating the compounding-error problem with a multi-step model. *arXiv preprint arXiv:1905.13320*, 2019.

Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in Neural Information Processing Systems*, 31, 2018.

Clavera, I., Fu, Y., and Abbeel, P. Model-augmented actor-critic: Backpropagating through paths. In *International Conference on Learning Representations*, 2020.

Deisenroth, M. and Rasmussen, C. E. 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.

Dulac-Arnold, G., Levine, N., Mankowitz, D. J., Li, J., Paduraru, C., Gowal, S., and Hester, T. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, pp. 1–50, 2021.

Eysenbach, B., Khazatsky, A., Levine, S., and Salakhutdinov, R. Mismatched no more: Joint model-policy optimization for model-based rl. *arXiv preprint arXiv:2110.02758*, 2021.

Farahmand, A.-m., Barreto, A., and Nikovski, D. Value-aware loss function for model-based reinforcement learning. In *Artificial Intelligence and Statistics*, pp. 1486–1494. PMLR, 2017.

Froehlich, L., Lefarov, M., Zeilinger, M., and Berkenkamp, F. On-policy model errors in reinforcement learning. In *International Conference on Learning Representations*, 2022.

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

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Goyal, A., Brakel, P., Fedus, W., Singhal, S., Lillicrap, T., Levine, S., Larochelle, H., and Bengio, Y. Recall traces: Backtracking models for efficient reinforcement learning. In *International Conference on Learning Representations*, 2018.

Grimm, C., Barreto, A., Singh, S., and Silver, D. The value equivalence principle for model-based reinforcement learning. *Advances in Neural Information Processing Systems*, 33:5541–5552, 2020.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. 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.

Hazan, E., Kakade, S., Singh, K., and Van Soest, A. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, pp. 2681–2691. PMLR, 2019.

Hu, H., Ye, J., Zhu, G., Ren, Z., and Zhang, C. Generalizable episodic memory for deep reinforcement learning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 4380–4390, 2021.

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

Kumar, V., Todorov, E., and Levine, S. 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.

Kurutach, T., Clavera, I., Duan, Y., Tamar, A., and Abbeel, P. Model-ensemble trust-region policy optimization. In *International Conference on Learning Representations*, 2018.

Lai, H., Shen, J., Zhang, W., and Yu, Y. Bidirectional model-based policy optimization. In *International Conference on Machine Learning*, pp. 5618–5627. PMLR, 2020.

Lai, H., Shen, J., Zhang, W., Huang, Y., Zhang, X., Tang, R., Yu, Y., and Li, Z. On effective scheduling of model-based reinforcement learning. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

Li, C., Wang, Y., Chen, W., Liu, Y., Ma, Z.-M., and Liu, T.-Y. Gradient information matters in policy optimization by back-propagating through model. In *International Conference on Learning Representations*, 2022.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *ICLR*, 2016.

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

Luo, Y., Xu, H., Li, Y., Tian, Y., Darrell, T., and Ma, T. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. In *International Conference on Learning Representations*, 2018.

Oh, Y., Shin, J., Yang, E., and Hwang, S. J. Model-augmented prioritized experience replay. In *International Conference on Learning Representations*, 2022.

Pan, F., He, J., Tu, D., and He, Q. Trust the model when it is confident: Masked model-based actor-critic. *Advances in Neural Information Processing Systems*, 2020.

Parr, R., Li, L., Taylor, G., Painter-Wakefield, C., and Littman, M. L. 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.

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

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

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

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. In *ICLR (Poster)*, 2016.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609, 2020.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

Shen, J., Zhao, H., Zhang, W., and Yu, Y. Model-based policy optimization with unsupervised model adaptation. *Advances in Neural Information Processing Systems*, 33, 2020.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

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

Sutton, R. S., Szepesvári, C., Geramifard, A., and Bowling, M. 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.

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

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

Wang, T. and Ba, J. Exploring model-based planning with policy networks. In *International Conference on Learning Representations*, 2019.

Yang, R., Zhang, M., Hansen, N., Xu, H., and Wang, X. Learning vision-guided quadrupedal locomotion end-to-end with cross-modal transformers. In *International Conference on Learning Representations*, 2022.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J. Y., Levine, S., Finn, C., and Ma, T. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

Zhang, T., Rashidinejad, P., Jiao, J., Tian, Y., Gonzalez, J. E., and Russell, S. Made: Exploration via maximizing deviation from explored regions. *Advances in Neural Information Processing Systems*, 34, 2021.

Appendix: Live in the Moment: Learning Dynamics Model Adapted to Evolving Policy

A. Useful lemma

Lemma A.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:*

$$|v_T^{\pi_1}(s') - v_{\hat{T}}^{\pi_2}(s')| \leq \gamma \mathbb{E}_{(s,a)\sim\rho_T^{\pi_1}} |T(s'|s,a) - \hat{T}(s'|s,a)| + \gamma d_{\mathcal{F}_{s'}}(\rho_T^{\pi_1}, \rho_{\hat{T}}^{\pi_2}) \tag{7}$$

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

$$v_T^{\pi}(s') = (1-\gamma)v_0(s') + \gamma \int_{(s,a)} \rho_T^{\pi}(s,a) T(s'|s,a) \mathrm{d}s\mathrm{d}a, \tag{8}$$

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

$$\begin{aligned}
&|v_T^{\pi_1}(s') - v_{\hat{T}}^{\pi_2}(s')| \\
=&\gamma \left| \int_{(s,a)} \rho_T^{\pi_1}(s,a) T(s'|s,a) - \rho_{\hat{T}(s,a)}^{\pi_2} \hat{T}(s'|s,a) \mathrm{d}s\mathrm{d}a \right| \\
=&\gamma \left| \mathbb{E}_{(s,a)\sim\rho_T^{\pi_1}}[T(s'|s,a)] - \mathbb{E}_{(s,a)\sim\rho_{\hat{T}}^{\pi_2}}[\hat{T}(s'|s,a)] \right| \\
\leq&\gamma \left| \mathbb{E}_{(s,a)\sim\rho_T^{\pi_1}}[T(s'|s,a) - \hat{T}(s'|s,a)] \right| + \gamma \left| \mathbb{E}_{(s,a)\sim\rho_T^{\pi_1}}[\hat{T}(s'|s,a)] - \mathbb{E}_{(s,a)\sim\rho_{\hat{T}}^{\pi_2}}[\hat{T}(s'|s,a)] \right| \\
\leq&\gamma \mathbb{E}_{(s,a)\sim\rho_T^{\pi_1}} |T(s'|s,a) - \hat{T}(s'|s,a)| + \gamma d_{\mathcal{F}_{s'}}(\rho_T^{\pi_1}, \rho_{\hat{T}}^{\pi_2})
\end{aligned} \tag{9}$$

□

B. Proof of main theorem

Theorem B.1 *Given the historical policy mixture $\pi_{mix,k} = (\Pi^k, w^k)$ 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:*

$$\begin{aligned}
&J(\pi, T) - J(\pi, \hat{T}) \\
\leq&2\gamma r_{max} \mathbb{E}_{(s,a)\sim\rho_T^{\pi}}[D_{TV}(T(s'|s,a)||\hat{T}(s'|s,a))] + r_{max} \sum_{i=0}^{k} w_i^k (\gamma \mathrm{Vol}(\mathcal{S}) \xi_{\rho_i} + 2\xi_{\pi_i}) \\
&+ 2r_{max} D_{TV}(\rho_{\hat{T}}^{\pi_{mix}}(s,a)||\rho_{\hat{T}}^{\pi}(s,a))
\end{aligned} \tag{10}$$

Proof.

$$
\begin{aligned}
&\left|J(\pi,T)-J(\pi,\hat{T})\right| \\
=&\left|J(\pi,T)-J(\pi_{mix},\hat{T})+J(\pi_{mix},\hat{T})-J(\pi,\hat{T})\right| \\
\leq&\underbrace{\left|\int_{(s,a)}(\rho_T^{\pi}(s,a)-\rho_{\hat{T}}^{\pi_{mix}}(s,a))r(s,a)\mathrm{d}s\mathrm{d}a\right|}_{term1}+\underbrace{\left|\int_{(s,a)}(\rho_{\hat{T}}^{\pi_{mix}}(s,a)-\rho_{\hat{T}}^{\pi}(s,a))r(s,a)\mathrm{d}s\mathrm{d}a\right|}_{term2}
\end{aligned}
\tag{11}
$$

For term 1:

$$
\begin{aligned}
&\left|\int_{(s,a)}(\rho_T^{\pi}(s,a)-\rho_{\hat{T}}^{\pi_{mix}}(s,a))r(s,a)\mathrm{d}s\mathrm{d}a\right| \\
=&\left|\int_{(s,a)}(v_T^{\pi}(s)\pi(a|s)-v_{\hat{T}}^{\pi_{mix}}(s)\pi_{mix}(a|s))r(s,a)\mathrm{d}s\mathrm{d}a\right| \\
=&\left|\int_{(s,a)}(v_T^{\pi}(s)\pi(a|s)-v_{\hat{T}}^{\pi_{mix}}(s)\pi(a|s)+v_{\hat{T}}^{\pi_{mix}}(s)\pi(a|s)-v_{\hat{T}}^{\pi_{mix}}(s)\pi_{mix}(a|s))r(s,a)\mathrm{d}s\mathrm{d}a\right| \\
\leq&\left|\int_{(s,a)}(v_T^{\pi}(s)-v_{\hat{T}}^{\pi_{mix}}(s))\pi(a|s)r(s,a)\mathrm{d}s\mathrm{d}a\right|+\left|\int_{(s,a)}(v_{\hat{T}}^{\pi_{mix}}(s)(\pi(a|s)-\pi_{mix}(a|s))r(s,a)\mathrm{d}s\mathrm{d}a\right| \\
\leq&r_{max}\int_s\left|v_T^{\pi}(s)-v_{\hat{T}}^{\pi_{mix}}(s)\right|\mathrm{d}s+2r_{max}\mathbb{E}_{s\sim v_{\hat{T}}^{\pi_{mix}}}\left[D_{TV}(\pi(a|s)||\pi_{mix}(a|s))\right]
\end{aligned}
\tag{12}
$$

For the first term of last inequality in Eq. 12, according to Lemma. A.1 we have:

$$
\begin{aligned}
&r_{max}\int_s\left|v_T^{\pi}(s)-v_{\hat{T}}^{\pi_{mix}}(s)\right|\mathrm{d}s \\
\leq&r_{max}\gamma\mathbb{E}_{(s,a)\sim\rho_T^{\pi}}\int_{s'}\left|T(s'|s,a)-\hat{T}(s'|s,a)\right|\mathrm{d}s'+r_{max}\gamma\int_{s'}d_{\mathcal{F}_{s'}}(\rho_T^{\pi},\rho_{\hat{T}}^{\pi^*})\mathrm{d}s'
\end{aligned}
\tag{13}
$$

We use total variance distance as the $\mathcal{F}_{s'}$ to measure the distance between ρ_T^{π} and $\rho_{\hat{T}}^{\pi_{mix}}$. 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 $\rho_{\hat{T}}^{\pi_{mix}}$ is approximately equal to $\rho_T^{\pi_{mix}}$. This assumption is required by many model-based RL methods (Voelcker et al., 2022). Then Eq. 13 can be expressed as:

$$
\begin{aligned}
&r_{max}\int_s\left|v_T^{\pi}(s)-v_{\hat{T}}^{\pi_{mix}}(s)\right|\mathrm{d}s \\
\leq&r_{max}\gamma\mathbb{E}_{(s,a)\sim\rho_T^{\pi}}\int_{s'}\left|T(s'|s,a)-\hat{T}(s'|s,a)\right|\mathrm{d}s'+r_{max}\gamma\int_{s'}D_{TV}(\rho_T^{\pi}||\rho_T^{\pi_{mix}})\mathrm{d}s' \\
\leq&2\gamma r_{max}\mathbb{E}_{(s,a)\sim\rho_T^{\pi}}[D_{TV}(T(s'|s,a)||\hat{T}(s'|s,a))]+\gamma\mathrm{Vol}(\mathcal{S})r_{max}D_{TV}(\rho_T^{\pi}||\rho_T^{\pi_{mix}})
\end{aligned}
\tag{14}
$$

Combined Eq. 12 with Eq. 14, we can get:

$$
\begin{aligned}
&\left|\int_{(s,a)}(\rho_T^{\pi}(s,a)-\rho_{\hat{T}}^{\pi_{mix}}(s,a))r(s,a)\mathrm{d}s\mathrm{d}a\right| \\
\leq& 2\gamma r_{max}\mathbb{E}_{(s,a)\sim\rho_T^{\pi}}[D_{TV}(T(s'|s,a)||\hat{T}(s'|s,a))]+\gamma\mathrm{Vol}(\mathcal{S})r_{max}D_{TV}(\rho_T^{\pi}(s,a)||\rho_T^{\pi_{mix}}(s,a)) \\
&+2r_{max}\mathbb{E}_{s\sim v_{\hat{T}}^{\pi_{mix}}}\left[D_{TV}(\pi(a|s)||\pi_{mix}(a|s))\right] \\
=& 2\gamma r_{max}\mathbb{E}_{(s,a)\sim\rho_T^{\pi}}[D_{TV}(T(s'|s,a)||\hat{T}(s'|s,a))]+\gamma\mathrm{Vol}(\mathcal{S})r_{max}D_{TV}(\rho_T^{\pi}(s,a)||\sum_{i=0}^{k}w_i\rho_T^{\pi_i}(s,a)) \\
&+2r_{max}\mathbb{E}_{s\sim v_{\hat{T}}^{\pi_{mix}}}\left[D_{TV}(\pi(a|s)||\sum_{i=0}^{k}w_i\pi_i(a|s))\right] \\
=& 2\gamma r_{max}\mathbb{E}_{(s,a)\sim\rho_T^{\pi}}[D_{TV}(T(s'|s,a)||\hat{T}(s'|s,a))]+\gamma\mathrm{Vol}(\mathcal{S})r_{max}\sum_{i=0}^{k}w_iD_{TV}(\rho_T^{\pi}(s,a)||\rho_T^{\pi_i}(s,a)) \\
&+2r_{max}\sum_{i=0}^{k}w_i\mathbb{E}_{s\sim v_{\hat{T}}^{\pi_{mix}}}\left[D_{TV}(\pi(a|s)||\pi_i(a|s))\right]
\end{aligned}
\tag{15}
$$

Finally, based on Eq. 15, we get:

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

and the proof is completed. □

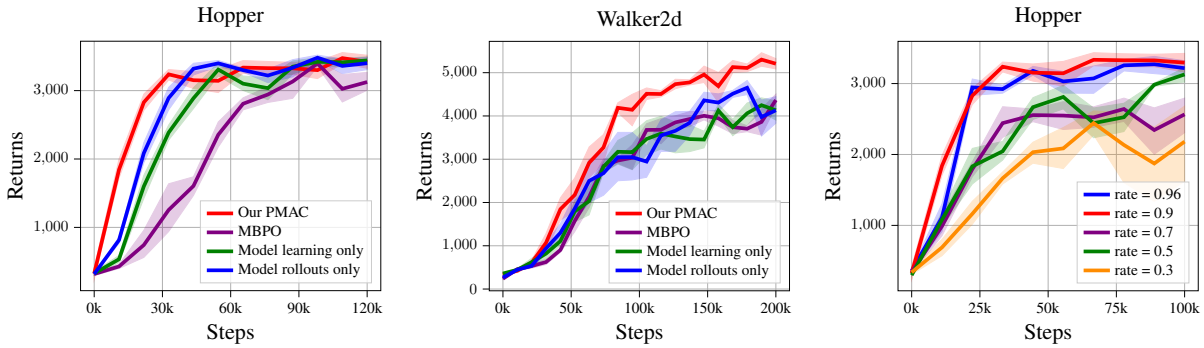
C. More experiments

C.1. Ablation Study of PMAC

As we described in Section 4, we use the historical policy mixture distribution twice after adjusting it. The first time we sample the training data from the real sample buffer according to this distribution to train the dynamics model. The second time is to sample the initial states from the the real sample buffer according to this distribution for model rollouts. In this section, we conduct ablation experiments to show the impact of the policy-adaptation mechanism in these two parts respectively. We conducted our experiments in Hopper and Walker2d, and the performance curves are shown in Figures 4(a) and 4(b). We find that using the policy-adaptation mechanism only for model learning or model rollouts significantly improves in Hopper compared to MBPO. However, the improvement in Walker2d is not significant; only the asymptotic performance of using the policy-adaptation mechanism for model rollouts is better than MBPO. We think this is because the model-generate samples are all obtained through the model rollouts. If the initial samples of the model rollouts are not in the state-action visitation distribution of the current policy, then even if the learned dynamics model is accurate, the model-generate samples are helpless for policy learning. In addition, when the initial samples of the model rollouts are sampled according to the adjusted historical policy mixture distribution, if the current dynamics model is not adapted to the current policy, it will also make inaccurate predictions, thereby misleading policy learning. Therefore, model learning and model rollout should be synergistic; that is, the training data for training the dynamics model and the initial samples of model rollouts should obey the same distribution, so that the model prediction error can be minimized.

C.2. Discussion of Policy-adaptation Rate

The experiment in this section is about the most crucial hyper-parameter of PMAC, the policy-adaptation rate. This hyper-parameter determines how much the distribution of the historical policy mixture tilts towards the current policy. We test different policy-adaptation rate rates in Hopper and find that 0.9 is the best rate. We think this is because, in the model-based RL, the number of real samples obtained by each historical policy is very limited (1000 samples in both our PMAC and MBPO implementations). If the policy-adaptation rate is low, the policy mixture distribution will be overly inclined to the current policy. Therefore, it is difficult to learn a dynamics model that is accurate enough for the current policy based only on these limited samples. Combined with our previous analysis, the distribution shift of adjacent policies in the historical policy sequence is small, so we need to adjust the policy-adaptation rate appropriately to effectively use the samples obtained from adjacent policies to learn the dynamics model.



(a) Hopper (b) Walker2d (c) Policy-adaptation Rate

Figure 4: Ablation studies.

D. Implementation

D.1. Implementation Details

We implement PMAC based on the PyTorch-version MBPO (Liu et al., 2020). We also set the ensemble size of PMAC 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 policy-adaptation rate α to be 0.9. 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 1. Other hyper-parameter settings are shown in Table 2. For AMPO[1], MBPO[2], PETS[3], SAC[4], and GEM[5], we use their open source implementations. We evaluate PMAC and other baselines on six MuJoCo-v2 continuous control environments (Todorov et al., 2012) with a maximum horizon of 1000, including HalfCheetah, Hopper, Walker2d, Ant, Humanoid, and Pusher. For Ant and 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 5.3, 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 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

[1] https://github.com/RockySJ/ampo
[2] https://github.com/Xingyu-Lin/mbpo_pytorch
[3] https://github.com/kchua/handful-of-trials
[4] https://github.com/pranz24/pytorch-soft-actor-critic
[5] https://github.com/MouseHu/GEM
[6] https://openreview.net/forum?id=WuEiafqdy9H

Table 1: Rollout horizon settings for PMAC

Walker2d	Hopper	Huamnoid	HalfCheetah	Ant	Pusher
1	[1, 15, 20, 100]	[1, 25, 20, 300]	1	[1, 25, 20, 100]	1

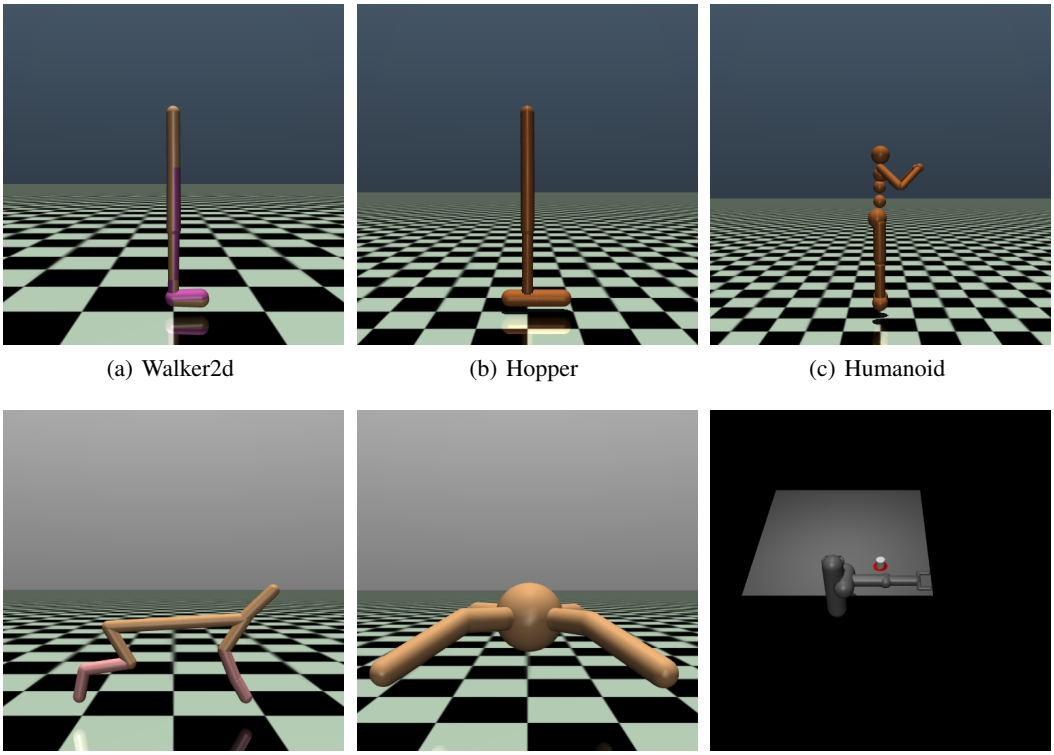
Table 2: Hyper-parameter settings for PMAC

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

bug and conduct the experiment, so the results of MaPER are lower than those reported in their paper.

D.2. Experiment environments

We visualize the six MuJoCo continuous control tasks we used in Figure 5 including Walker2d, Hopper, Humanoid, HalfCheetah, Ant, and Pusher. The first five tasks aim to keep the agent moving forward without falling, and the last task, Pusher, is designed to control the robotic arm to push the object to a fixed point.



(a) Walker2d (b) Hopper (c) Humanoid

(d) HalfCheetah (e) Ant (f) Pusher

Figure 5: Six MuJoCo-v2 continuous control tasks used in our experiment