MOBA: MULTI-TEACHER MODEL BASED REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Although reinforcement learning (RL) shines at solving decision-making problems, it not only requires collecting a large amount of environment data but also is time-consuming for training and interaction, making it hard to apply to real applications. To reduce the time-cost and improve the data efficiency, model-based reinforcement learning uses a learned system model to predict system dynamics (i.e. states or rewards) and makes a plan accordingly, and can avoid frequent interactions between the agent and the environment. Model-based methods suffer from the model-bias problem, where certain spaces of model are inaccurate, resulting in policy learning variations and system performance degradation. We propose a **M**ulti-teacher M**O**del **BA**sed Reinforcement Learning algorithm (MOBA), which leverages multi-teacher knowledge distillation theory to solve the model-bias problem. Specifically, different teachers search different spaces and learn various instances of a system. By distilling and transferring the teacher knowledge to a student, the student model is able to learn a generalized dynamic model that covers the state space. Moreover, to overcome the instability of multi-teacher knowledge transfer, we learn a set of student models and use an ensemble method to jointly predict system dynamics. We evaluate MOBA in high-dimensional control locomotion tasks. Results show that, compared with SOTA model-free methods, our method can improve the data efficiency and system performance by up to 75% and 10%, respectively. Moreover, our method outperforms other SOTA model-based approaches by up to 63.2% when exposed to high-range model-bias environments.

1 INTRODUCTION

Building on the tremendous successes of deep neural networks, the combining of classical reinforcement learning (RL) methods with deep learning has become increasingly popular throughout the years. Deep reinforcement learning approaches have shown human-level performance in domains like Atari games (Mnih et al., 2015), Go (Silver et al., 2016), and complex continuous control tasks (Lillicrap et al., 2015). Most astonishing achievements are built upon the **model-free** reinforcement learning algorithms which do not need the knowledge of environment. Model-free reinforcement learning suffers from low sample efficiency and requires huge amounts of environment data, since it needs frequent interactions with the real environment, making it hard to apply to real applications. On the contrary, **model-based** reinforcement learning algorithms use system dynamic models for policy learning. The system model can be given or learned jointly with the agent policy. By utilizing the system model, model-based reinforcement learning can avoid frequent environment interaction and potentially be more sample efficient than model-free algorithms, thus it is more suitable to real-world scenarios (Deisenroth & Rasmussen, 2011; Levine et al., 2016; Venkatraman et al., 2016).

Most existing model-based RL approaches, however, only work well on low-dimensional tasks and struggle to perform effectively in continuous control tasks. The reason is that the policy optimization tends to exploit regions with insufficient data for model training, leading to failures. The This is the well-known model-bias problem, which assume that the learned system model sufficiently accurately resembles the real environment (Schneider, 1996; Atkeson & Santamaría, 1997; Deisenroth & Rasmussen, 2011). Although existing studies try to resolve the model-bias problem by learning an ensemble models (Kurutach et al., 2018) or using a monotonic improvement approach (Janner et al.,

2019), we argue that the learned models are not diverse enough and fail to deal with environment changes.

Knowledge distillation (KD) is first introduced for model compression. Recently, this technique has been widely used in different problems such as data privacy (Papernot et al., 2017), natural language processing (NLP) (Tan et al., 2019), few-shot learning (Dvornik et al., 2019). The motivation of using KD can be concluded in two folds: (1) KD can be used to transfer different system dynamics (e.g. state transition probability) to one student model, which can be used to unseen scenario. (2) Multi-teacher KD is a way of solving model-bias problem because the student model is learned from different environment instances, which contain various state spaces. RL can be benefit from both of the above two advantages. First, RL agent is trained by the student model, which is a generalized model contains different transition probabilities. Second, KD is benefit for agent exploration, since it provides accurate state predictions.

In this paper, we propose a **M**ulti-teacher M**O**del **BA**sed Reinforcement Learning algorithm (MOBA) to learn a set of environment models, which we called teacher models. Our key observation is that the policy performance is closely related to the learned model, thus it is likely that, with enough data, the learned model will be accurate enough. Motivated by this, we improve the model learning accuracy by using multiple teacher models, where different teachers contain different aspects of an environment. In contrast to existing methods that directly learn a model from one environment, we show that distilling multi-teacher model knowledge not only suppresses the model bias but also avoids frequent interactions between the agent and environment, thus reducing the time-cost and improving the data efficiency. Furthermore, we use an ensemble of student networks to predict system dynamics and improve the stability of learned student model and policy. More importantly, our proposed framework is algorithm agnostic, which can be easily combined with other model-free methods to improve the sample efficiency accordingly.

The major contributions of this work are summarized as follows:

- We introduce a multi-teacher knowledge distillation approach to solve the model bias problem. The learned dynamic student models have a high model prediction accuracy, enabling robustness to environment changes.
- We propose an ensemble student prediction method to improve the stability in student model and policy learning.

2 Related Work

2.1 Model-based Reinforcement Leaning

Model-based reinforcement leaning algorithms are famous for solving real-world sequential decision-making problems due to their data efficiency (Kaelbling et al., 1996). Usually, one cannot directly obtain the system model, thus there are plenty of ways to approximate the dynamic model by treating the system model as a black box. Among all the approximators, Neural Networks (NNs) based approximators are widely used in the model-based reinforcement learning, due to its asymptotic performance of high-capacity approximate function (Schrittwieser et al., 2019; Kaiser et al., 2020). Moreover, NNs are able to scale to high dimensional control problems with better sample efficiency (Nagabandi et al., 2018; Chua et al., 2018; Wahlström et al., 2015). The learned models may be inaccurate in high-dimensional dynamic systems. A major challenge when using high-capacity dynamics models lies in policies exploiting model inaccuracies where less data is provided, which is also known as the model-bias problem.

Previous work has tried to alleviate the model-bias problem by learning a distribution of models (Depeweg et al., 2017; Chua et al., 2018; Kurutach et al., 2018; Rajeswaran et al., 2017) or by learning adaptive models (Clavera et al., 2018a; Fu et al., 2016; Gu et al., 2016). In Clavera et al. (2018b), authors use an ensemble of learned dynamic models to overcome the model inaccuracy problem. To solve the bias of the model generated data, Janner et al. (2019) study the role of model usage in the policy optimization. Nagabandi et al. (2019) also propose a method to alleviate the model-bias problem by learning a global model that is accurate through the entire state space. In contrast to the above-mentioned approaches, we use the multi-teacher knowledge distillation to learn a generalized dynamics model. Specifically, each teacher learns an instance of the system, which

only differs in environment settings. We argue that the pure ensemble method still struggles in generating similar trajectories, which cannot make enough difference in model learning. Besides, we also use ensemble learning ideas in student training as it can stabilize both model and policy training (Clavera et al., 2018b; Kurutach et al., 2018).

2.2 MULTI-TEACHER KNOWLEDGE DISTILLATION

Knowledge distillation is first proposed by (Hinton et al., 2015), and aims to help the training process of a smaller student network under the supervision of a larger teacher network. FitNets (Romero et al., 2014) encourage an intermediate layer of student network to have the ability to match the outputs of some intermediate layers of teacher network. The relationships among different neural layers and neurons (Yim et al., 2017; Lee & Song, 2019) are also considered as knowledge distilled by teacher models. To better transfer knowledge, multiple teacher networks have been introduced in the knowledge distillation framework where a student network can simultaneously receive knowledge distilled from multiple teacher networks.

Average ensemble of logits is a commonly-used approach in multi-teacher knowledge distillation. In such a setting, a student network is encouraged to learn the average softened output of multiple teacher networks' logits via minimizing the cross-entropy loss, and the average softened output serves as the incorporation of multiple teacher networks in the output layer. (Tarvainen & Valpola, 2017; Papernot et al., 2016; You et al., 2017b). (Yuan et al., 2021) formulate the teacher selection problem under a RL framework, where each teacher network is assigned an appropriate weight based on various training samples and the outputs of teacher networks.

To this end, we utilize multi-teacher knowledge distillation to solve the model-bias problem, where each teacher learns an instance of the system. In this way, the teacher networks can simultaneously guide the student network for a better system dynamics prediction. Moreover, the data and time efficiency are improved by reducing interactions between the agent and environment. Different from existing works, we also consider a set of student networks to improve the stability of learned network and policy.

3 BACKGROUND

3.1 MODEL-BASED REINFORCEMENT LEARNING

A discrete-time finite Markov decision process (MDP) is defined by the tuple $< S, A, p, r, \gamma >$. Here, S is the states space, A is the action space, $p : S \times A \times S$ is a state transition function, $r : S \times A \to \mathbb{R}$ is a reward function, γ is the discount factor. The goal of reinforcement learning is to learn an agent policy π that can collect the largest expected return $\eta(\theta) = \mathbb{E}[\sum_t^T \gamma^t r_t]$, which is denoted by η.

In contrast to model-free RL algorithms that don't model system dynamics, model-based RL methods explicitly learn the state transition function p. The system dynamics can be treated as a black box and learned using various ways. One of the methods we adopt in this paper is neural network function approximator $f_\phi(s_{t+1}|s_t, a_t)$. We maximize the log-likelihood of the state transition distribution by training the parameters ϕ. In this paper, the learned system dynamic functions are represented deterministically.

3.2 KNOWLEDGE DISTILLATION

To improve the performance of learning a student network, (Hinton et al., 2015) introduces knowledge distillation framework, which uses the supervision knowledge distilled from teacher networks. The student network is trained to have similar output distribution with regard to teacher networks. Outputs of student network is regulated to be close to the ground truth labels as well as outputs of teacher networks.

$$\mathcal{L}_{KD}(\theta_S) = \mathcal{H}(y, f_{\theta_S}) + \alpha \mathcal{H}(f_{\theta_T}, f_{\theta_S}),$$

where $\mathcal{H}(\cdot, \cdot)$ is the cross-entropy, α is a hyper-parameter that regularize the second term and y stands for the ground-truth label. In addition, f_{θ_S} and f_{θ_T} are student network and teacher network respectively.

4 METHODOLOGY

In this section, we introduce our model-based RL framework, which learns a generalized system model using multi-teacher knowledge distillation. The whole architecture is depicted in Figure 1.

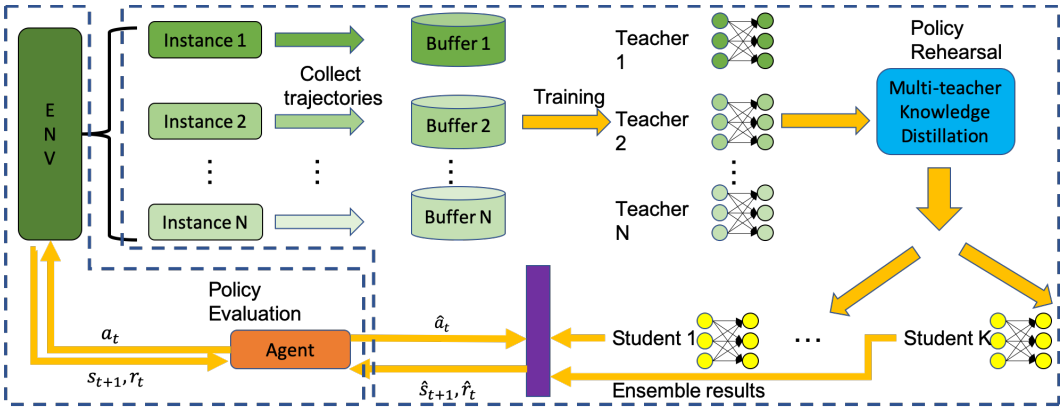

Figure 1: The architecture of our proposed method.

4.1 SYSTEM MODEL LEARNING

One of the vital component of this work is the system model learning, which learns system dynamics in a black box manner. In particular, we learn two system dynamics, which are state transition function and reward function. Furthermore, we tweak the environment settings to have different instances of the same environment[1] We train a dynamic model as a teacher for each environment instance. Then, we distill the knowledge from multiple teachers to one student model. The student model will be used to train the policy.

4.1.1 THE TEACHER MODEL

In this work, we assume the state transition function to be a deterministic function of the state s_t and action a_t. We use three hidden layer neural networks to approximate this function. For each model, we learn two models: state transition function $f_{\phi_k^T}(s_t, a_t)$ and reward function $f_{\eta_k^T}(s_t, a_t)$, where $k \in [1, \ldots, K]$ and K is the number of instances. The learned models generate an imaginary trajectory to teach one single policy network. In our work, the reward function remains the same across tasks while the dynamics vary. Therefore, each teacher model constitutes a different belief about what the dynamics in the true environment could be and minimizes the loss:

$$\mathcal{L}_T = \sum_{k=1}^{K} \sum_{(s_t, a_t, s_{t+1}, r_t) \in \mathcal{D}_k} [||s_{t+1} - f_{\phi_k^T}(s_t, a_t)||_2^2 + ||r_t - f_{\eta_k^T}(s_t, a_t)||_2^2]. \tag{1}$$

4.1.2 THE STUDENT MODEL

To distill the knowledge from multi-teacher model, we adopt the idea from (You et al., 2017a) and construct two parts in the student model loss function, which are ground truth loss and knowledge distillation loss.

$$\mathcal{L}_S = \sum_{k=1}^{K} \sum_{(s_t, a_t, s_{t+1}) \in \mathcal{D}_k} [||s_{t+1} - f_{\phi^S}(s_t, a_t)||_2^2 + ||f_{\phi_k^T}(s_t, a_t) - f_{\phi^S}(s_t, a_t)||_2^2], \tag{2}$$

where f_{ϕ^S} is the student network parameterized by ϕ^S.

[1] See in appendix

4.2 POLICY REHEARSAL

We leverage the learned student model with policy learning. In this step, we train the policy using generated trajectories, which we call policy rehearsal. We name generated trajectories rehearsal trajectories. At each time-step t, the student model computes the hypothetical state $\hat{s}_{t+1}$ and reward r_t. This step mirrors the structure of the underlying MDP model $\mathcal{M}$ and computes an approximate MDP $\hat{\mathcal{M}}$ with expected reward and state. Given such a model, we can help the agent rehearse future actions based on the current state, which could be seen as a way of planning (Schrittwieser et al., 2019). The goal of the agent is maximizing the future return and updating the corresponding policy with respect to the approximate MDP $\hat{\mathcal{M}}$: $\hat{\eta}(\theta; \phi_S) = \mathbb{E}_{\hat{\tau}}[\sum_{t=0}^{T} r(s_t, a_t)]$, where $\hat{\tau} = (s_0, a_0, \ldots), s_0 \sim \rho_0(\cdot), a_t \sim \pi_\theta(\cdot|s_t)$ and $s_{t+1} = f_{\phi^S}$.

Specifically, we adopt PPO algorithm as our policy training strategy. In addition, we adopt the idea from Kurutach et al. (2018) for early stopping. We will keep the policy learning with rehearsal trajectories over a period of time until $\hat{\eta}$ no longer improves.

4.3 POLICY EVALUATION

After policy rehearsal, We evaluate the policy in environment. Specifically, we perform a shallow trail, which rollout the environment with fixed timesteps the using current policy, and calculate the collected return. In the real MDP $\mathcal{M}$, the shallow trail is defined as $\eta(\theta) = \mathbb{E}][\sum_{t=0}^{T} r(s_t, a_t)]$, where T is a small number of steps. The policy learning continues as long as the current policy still improves:

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{1}[\eta(\theta_{new}) > \eta(\theta_{old}) + C], \tag{3}$$

where C is a threshold value that measures the relative improvements. We perform T times of shallow trail and compute the average value. The iteration continues as long as this ratio exceeds a certain threshold. In our implementation, we find 0.7 is a good threshold. The algorithm is described in Algorithm 1.

Algorithm 1 Vanilla Multi Teacher Model Based Reinforcement Learning (Vanilla-MOBA)

Require: Inner and outer step size α, β
Initialize the policy π_θ, the models $f_{\phi_1}, \ldots, f_{\phi_k}$ and $\mathcal{D} \leftarrow 0$

for T episodes **do**
  Sample trajectories from the real environment with policies $\pi_{\theta_1}, \ldots, \pi_{\theta_k}$.
  Add them to $\mathcal{D}_1, \ldots, \mathcal{D}_k$.
  Train model f_{ϕ_k} using $\mathcal{D}_k$.
  for Model $k \leftarrow 1$ to K **do**
    Optimise ϕ_k using Eqn. 1
  end for
  while Performance $\hat{\eta}$ still improves **do**
    Training student model ϕ_S using Eqn. 2
    Sample imaginary trajectories from student model
    Update the policy according to the imaginary trajectories
  end while
  Terminate if the policy evaluation ratio (using Eqn. 3) below the threshold.
end for

4.4 STUDENT MODEL ENSEMBLE

During the evaluation, we find that the algorithm proposed in Algorithm 1 suffered from high variance training performance, which is caused by the unstable multi-teacher student distillation (Wang & Yoon, 2020). To solve this issue, we propose a student model ensemble solution.

Instead of training one single student model, we learn a set of student models $f_{\phi_1^S}, \ldots, f_{\phi_k^S}$ using the same teacher models. All of student models are trained parallel via same loss function Eqn. 2. Fur-

Algorithm 2 Ensembled Multi Teacher Model Based Reinforcement Learning (MOBA)

Require: Inner and outer step size α, β
Initialize the policy π_θ, the models $f_{\phi_1}, \ldots, f_{\phi_k}$ and $\mathcal{D} \leftarrow 0$

1: **for** N episodes **do**
2: Follow steps 2-7 in Algorithm 1
3: Training K student model ϕ_k^S using Eqn. 2
4: Follow steps 8-9 in Algorithm 1
5: Sample imaginary trajectories from student models using Eqn. 4
6: Follow steps 11-14 in Algorithm 1
7: **end for**

thermore, to enlarge the discrepancies among these student models, we uniformly sample different parameters' weights $\phi_k^S, k \in K$ as the model initialization.

States and Rewards Ensemble Since our evaluation environments are continuous, it is no harm to simply average the learned states and rewards. This approach not only avoids student model overfitting but also stabilizes policy learning. We define the predicted next state as $\hat{s}_{t+1}$ and $\hat{r}_t$ and the predicted reward as $\hat{r}_t$. The averaged predictions are given as below:

$$\hat{s}_{t+1} = \frac{1}{K}\sum_{k=1}^{K}[f_{\phi_k^S}(s_t, a_t)],\ \hat{r}_t = \frac{1}{K}\sum_{k=1}^{K}[f_{\eta_k^S}(s_t, a_t)]. \tag{4}$$

Although we don't evaluate in discrete states or rewards environments, we propose a solution to generate rehearsal trajectories. In every step, we use the majority vote to decide the next state and reward. If predictions are different, we randomly choose one of them as the next state and reward. The ensemble MOBA is described in Algorithm 2.

5 Evaluation

5.1 Environment Setup

We evaluate our algorithms in continuous control benchmark tasks in PyBullet environment (Coumans & Bai, 2016–2021). Specifically, we choose six standard benchmark tasks: Ant, Half Cheetah, Humanoid, Walker, Inverted Double Pendulum, and Hopper. We create several instances, which only differ in environment parameters for each task, and each instance has a corresponding teacher model.

5.2 Comparison to SOTA: Model-Free Algorithms

We compare our method with three state-of-the-art model-free RL algorithms: (1) PPO: proximal policy optimization (Schulman et al., 2017) is one of the state-of-the-art policy update algorithms that has been widely used in continuous control problems . (2) SAC: soft actor-critic (Haarnoja et al., 2018) is an algorithm that optimizes a an off-policy algorithm, forming a bridge between stochastic policy optimization and DDPG-style approaches. (3) TD3: twin delayed DDPG (Fujimoto et al., 2018) is an algorithm that addresses the overestimate Q-value issue by introducing three critical tricks: clipped double-Q learning, "delayed" policy updates and target policy smoothing.

The results in Figure 2 show that our method can achieve asymptotic performance with a better convergence rate. Overall, in all the locomotion tasks our method MOBA can outperform other model-free methods in terms of convergence rate and mean episode return. Specifically, MOBA reduce an average 64.0% of the training time in most of the tasks, 75% in pendulum task, and outperform other algorithms in walker2D. Since we adopt PPO as our policy learning algorithm, our method can effectively improve the PPO performance in challenging tasks: hopper, walker2D half cheetah, which PPO is hard to converge to the optimal solution.

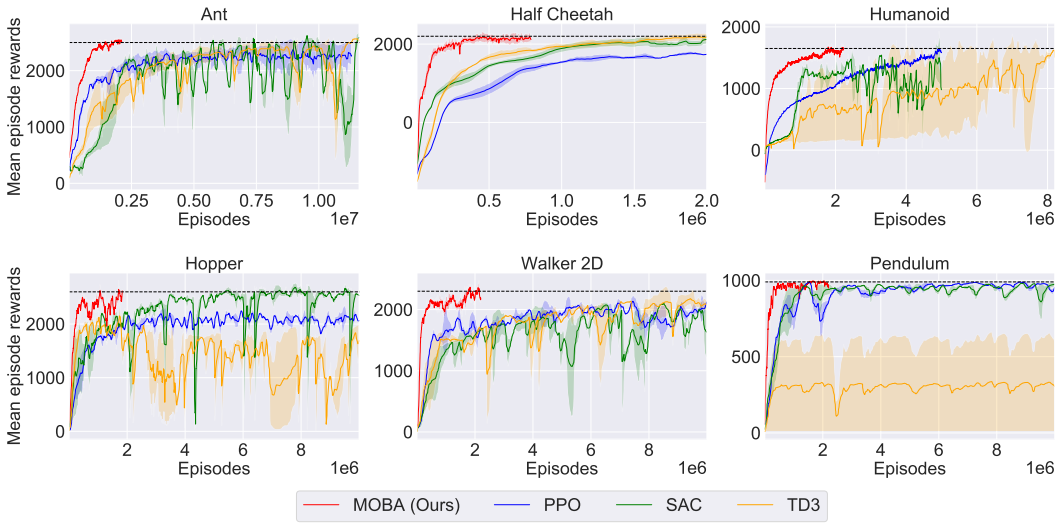

Figure 2: Learning curves of our method versus state-of-the-art model-free algorithms. Each learning curve is computed in three runs with different random seeds. The dash line depicts the desired best reward. MOBA ("Ours") achieves faster convergence rate and achieves better performance than model-free methods.

5.3 Comparison to SOTA: Model-Based Algorithms

To further understand the advantages of multi-teacher knowledge distillation in improving data efficiency, we compare our method with two state-of-the-art model-based RL algorithms: (1) GrBAL: gradient-based adaptive learning (Nagabandi et al., 2019), which uses a gradient-based method to learn a online adaptation of models in dynamic environments. (2) MBPO: model-based policy optimization (Janner et al., 2019), which incorporates a linear approximation of model generalization into the analysis and justifies using the model for truncated rollouts.

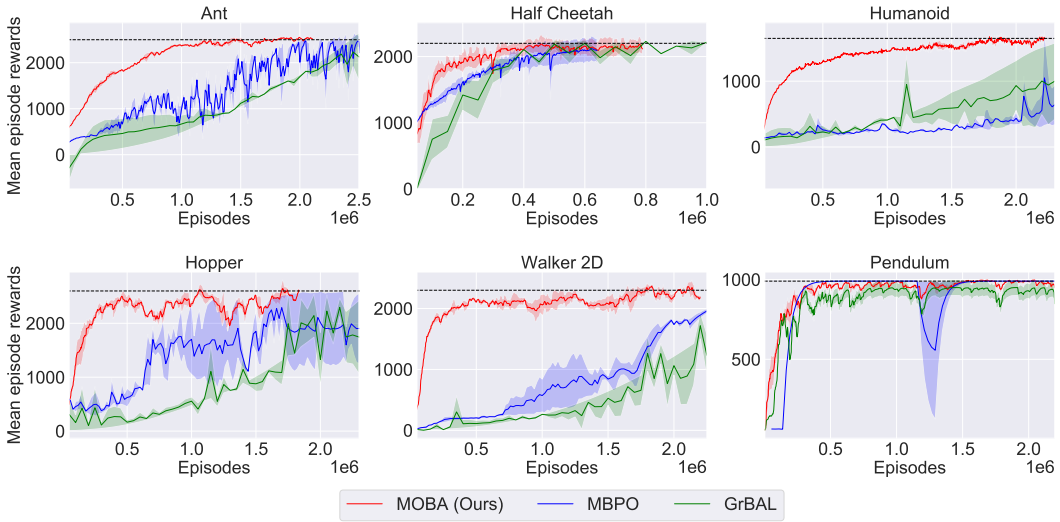

Figure 3: Learning curves of our method versus state-of-the-art model-based algorithms. Each learning curve is computed in three runs with different random seeds. MOBA ("Ours") achieves faster convergence rate than other model-based methods.

The results are shown in Figure 3. As we can see, our MOBA can still outperform the model-based methods in terms of sample efficiency in all tasks. The reasons are concluded as follows: (1) MBPO learns an ensemble of system models, which is trained from one instance of the environment and

unable to cover the different aspects of state space. (2) Although GrBAL can rapidly adapt to the environment changes, it fails to learn a model that covers all the state space and suffers model-bias problem. (3) Our MOBA uses multi-teacher knowledge distillation mechanism for model learning, which enhances the model accuracy. The learned student model covers different aspects of state space, which ensures the accuracy of predicted states during agent exploration. Additionally, we use the planning idea to rehearsal the agent trajectories with current policy and calculate its corresponding return. The agent learns to make the best decision by planning the rehearse trajectories and updates its policy. After the policy rehearsal, the agent interacts with the real environment and collects a real reward as well as new environment trajectories, which, as a result, helps to improve the policy rehearsal in the next iteration.

5.4 Dealing with The Model-bias Problem

To illustrate the model-bias problem and how it affects policy learning, we empirically measure the policy performance under different dynamic systems with biased noise. We adopt the idea from (Clavera et al., 2018b), which add biased Gaussian noise $\mathcal{N}(b, 0.1^2)$ to the next state prediction, where $b \sim \sqcap(0, b_{max})$ is sampled from a uniform distribution between 0 and b_{max}. In Figure 4 We show the policy performance under different ranges of adding bias.

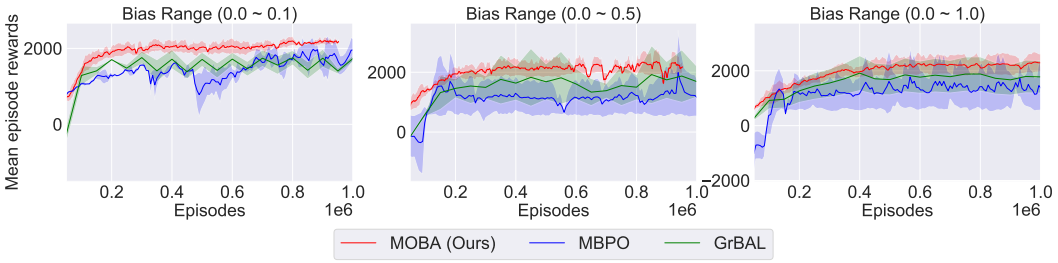

Figure 4: Learning curves of our methods versus SOTA model-based algorithms using three different bias ranged dynamic models in the half cheetah environment.

Our results show that MOBA consistently outperforms other two model-based algorithms when system model has some biased noises. Especially, when exposed to the strong bias environments i.e. $b_{max} = 0.5$ and $b_{max} = 1.0$, MBPO and GrBAL show large variance in the learning curve and fail to converge to an optimal solution. On the country, MOBA manage to solve the model-bias problem in all three different bias range cases, which shows its effectiveness. Specifically, our MOBA increases the mean episode rewards by up to 63.2% in strong bias environments.

5.5 Ablation Study

In this section, we analyze the effect of the student model ensemble component. Figure 5 shows the results of MOBA andVanilla-MOBA.

The results indicate that our proposed method can achieve better stability and lower variance by adding the student ensemble component. This phenomenon is even more noticeable in the Pendulum environment, which shows both MOBA and Vanilla-MOBA (in the best case) can find the optimal policy. The mean value of the Vanilla-MOBA is lower than MOBA, which shows the effectiveness of the student ensemble component.

6 Conclusion

In this paper, we study the model-bias problem in model-based RL, which often occurs in high-dimensional dynamic systems due to the policy exploration in inaccuracy model spaces. A major challenge lies in how to cover different spaces of the system dynamic model. To conquer this challenge, we propose an Ensembled Multi Teacher Model-Based Reinforcement Learning algorithm. MOBA learns a generalized dynamic model through the knowledge distillation from multiple teacher models, which learn from different instances of environments with various settings. To the

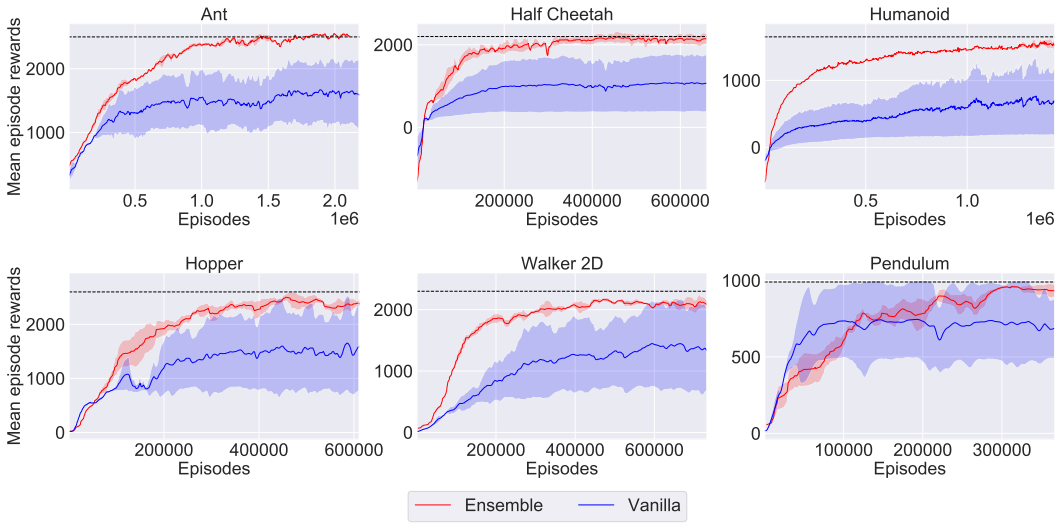

Figure 5: Learning curves of MOBA andVanilla-MOBA.

best of our knowledge, this is the first multi-teacher knowledge distillation approach for model learning. Evaluation results show that our method can outperform SOTA model-free and model-based methods in high-dimensional control locomotion tasks. Moreover, our method achieves the best performance when exposed to high range model-bias environments. The ablation study shows the effectiveness of leveraging the student model ensemble component to stabilize the policy and model learning.

REFERENCES

Ossama Ahmed, Frederik Träuble, Anirudh Goyal, Alexander Neitz, Manuel Wuthrich, Yoshua Bengio, Bernhard Schölkopf, and Stefan Bauer. Causalworld: A robotic manipulation benchmark for causal structure and transfer learning. In *ICLR*. OpenReview.net, 2021.

Christopher G. Atkeson and Juan Carlos Santamaría. A comparison of direct and model-based reinforcement learning. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation, Albuquerque, New Mexico, USA, April 20-25, 1997*, pp. 3557–3564. IEEE, 1997. doi: 10.1109/ROBOT.1997.606886. URL https://doi.org/10.1109/ROBOT.1997.606886.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *NeurIPS*, pp. 4759–4770, 2018.

Ignasi Clavera, Anusha Nagabandi, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt: Meta-learning for model-based control. *CoRR*, abs/1803.11347, 2018a.

Ignasi Clavera, Jonas Rothfuss, John Schulman, Yasuhiro Fujita, Tamim Asfour, and Pieter Abbeel. Model-based reinforcement learning via meta-policy optimization. In *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, volume 87 of *Proceedings of Machine Learning Research*, pp. 617–629. PMLR, 2018b. URL http://proceedings.mlr.press/v87/clavera18a.html.

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021.

Marc Peter Deisenroth and Carl Edward Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In Lise Getoor and Tobias Scheffer (eds.), *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pp. 465–472. Omnipress, 2011. URL https://icml.cc/2011/papers/323_icmlpaper.pdf.

Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. Learning and policy search in stochastic dynamical systems with bayesian neural networks. In *ICLR (Poster)*. OpenReview.net, 2017.

Nikita Dvornik, Julien Mairal, and Cordelia Schmid. Diversity with cooperation: Ensemble methods for few-shot classification. In *ICCV*, pp. 3722–3730. IEEE, 2019.

Justin Fu, Sergey Levine, and Pieter Abbeel. One-shot learning of manipulation skills with online dynamics adaptation and neural network priors. In *IROS*, pp. 4019–4026. IEEE, 2016.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1582–1591. PMLR, 2018. URL http://proceedings.mlr.press/v80/fujimoto18a.html.

Shixiang Gu, Timothy P. Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 2829–2838. JMLR.org, 2016.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1856–1865. PMLR, 2018. URL http://proceedings.mlr.press/v80/haarnoja18b.html.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015. URL http://arxiv.org/abs/1503.02531.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 12498–12509, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/5faf461eff3099671ad63c6f3f094f7f-Abstract.html.

Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *J. Artif. Intell. Res.*, 4:237–285, 1996.

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H. Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model based reinforcement learning for atari. In *ICLR*. OpenReview.net, 2020.

Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=SJJinbWRZ.

Seunghyun Lee and Byung Cheol Song. Graph-based knowledge distillation by multi-head attention network. In *BMVC*, pp. 141. BMVA Press, 2019.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.*, 17:39:1–39:40, 2016. URL http://jmlr.org/papers/v17/15-522.html.

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. *arXiv preprint arXiv:1509.02971*, 2015.

Volodymyr Mnih, Koray Kavukcuoglu, and et.al David Silver. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 1928–1937. JMLR.org, 2016.

Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *ICRA*, pp. 7559–7566. IEEE, 2018.

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 *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=HyztsoC5Y7.

Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755*, 2016.

Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian J. Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. In *ICLR*. OpenReview.net, 2017.

Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning robust neural network policies using model ensembles. In *ICLR (Poster)*. OpenReview.net, 2017.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

Jeff G. Schneider. Exploiting model uncertainty estimates for safe dynamic control learning. In *NIPS*, pp. 1047–1053. MIT Press, 1996.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy P. Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *CoRR*, abs/1911.08265, 2019. URL http://arxiv.org/abs/1911.08265.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

David Silver, Aja Huang, Chris J. Maddison, and et.al Arthur Guez. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

Xu Tan, Yi Ren, Di He, Tao Qin, Zhou Zhao, and Tie-Yan Liu. Multilingual neural machine translation with knowledge distillation. In *ICLR (Poster)*. OpenReview.net, 2019.

Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *arXiv preprint arXiv:1703.01780*, 2017.

Arun Venkatraman, Roberto Capobianco, Lerrel Pinto, Martial Hebert, Daniele Nardi, and J. Andrew Bagnell. Improved learning of dynamics models for control. In Dana Kulic, Yoshihiko Nakamura, Oussama Khatib, and Gentiane Venture (eds.), *International Symposium on Experimental Robotics, ISER 2016, Tokyo, Japan, October 3-6, 2016*, volume 1 of *Springer Proceedings in Advanced Robotics*, pp. 703–713. Springer, 2016. doi: 10.1007/978-3-319-50115-4_61. URL https://doi.org/10.1007/978-3-319-50115-4_61.

Niklas Wahlström, Thomas B. Schön, and Marc Peter Deisenroth. From pixels to torques: Policy learning with deep dynamical models. *CoRR*, abs/1502.02251, 2015.

Lin Wang and Kuk-Jin Yoon. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *CoRR*, abs/2004.05937, 2020.

Junho Yim, Donggyu Joo, Ji-Hoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *CVPR*, pp. 7130–7138. IEEE Computer Society, 2017.

Shan You, Chang Xu, Chao Xu, and Dacheng Tao. Learning from multiple teacher networks. In *KDD*, pp. 1285–1294. ACM, 2017a.

Shan You, Chang Xu, Chao Xu, and Dacheng Tao. Learning from multiple teacher networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1285–1294, 2017b.

Fei Yuan, Linjun Shou, Jian Pei, Wutao Lin, Ming Gong, Yan Fu, and Daxin Jiang. Reinforced Multi-Teacher Selection for Knowledge Distillation. In *AAAI*, pp. 14284–14291. AAAI Press, 2021.

A APPENDIX

A.1 DETAILS OF PYBULLET ENVIRONMENTS' INSTANCES

In this section, we explain how we implement instances for Pybullet tasks. In all the locomotion tasks (i.e., Ant, Half Cheetah, Humanoind, Hopper, and Walker2D), we modify the initial position, initial orientation, and goal state. Specifically, the initial position is chosen according to three standard position numbers suggested in code base[2]. As for initial orientation, the value (0, 0, 3) forces all the robots to face a random direction but stay straight. We also change the robot pitch and yaw to force the robot to start with a more challenging position. Moreover, we change the goal state, which is the distance the robot is required to walk. The detailed settings is shown in Table. 1.

Task	Number	Init Position	Init Orientation (raw, pitch, yaw)	Goal State (km)
Ant	1	(0, 0, 1.4)	(0, 1.57, 0)	500
	2	(0, 0, 0.25)	(0, 0, 3)	1500
	3	(0, 0, 0.45)	(0, 1.57, 3)	2000
Half Cheetah	1	(0, 0, 1.4)	(0, 1.57, 0)	500
	2	(0, 0, 0.25)	(0, 0, 3)	1500
	3	(0, 0, 0.45)	(0, 1.57, 3)	2000
Humanoid	1	(0, 0, 1.4)	(0, 1.57, 0)	500
	2	(0, 0, 0.25)	(0, 0, 3)	1500
	3	(0, 0, 0.45)	(0, 1.57, 3)	2000
Hopper	1	(0, 0, 1.4)	(0, 1.57, 0)	500
	2	(0, 0, 0.25)	(0, 0, 3)	1500
	3	(0, 0, 0.45)	(0, 1.57, 3)	2000
Walker2D	1	(0, 0, 1.4)	(0, 1.57, 0)	500
	2	(0, 0, 0.25)	(0, 0, 3)	1500
	3	(0, 0, 0.45)	(0, 1.57, 3)	2000
Pendulum	1	(-0.1, 0)	-	-
	2	(0.1, 0)	-	-
	3	(0, 0)	-	-

Table 1: Environment parameters for each instances of environments implemented for evaluation. Moreover, since pendulum task is not locomotion task that doesn't apply orientation and goal state. We only modify its initial position.

A.2 BASELINES MODIFICATION

Since the baselines are not designed for multiple instances of environment, to make a fair comparison, we use the multi-processing method to train baseline algorithms. This is first proposed in synchronized advantage actor-critic method (Mnih et al., 2016), which trains a vector of the same environments using the same algorithm. During the training, each process copies the same networks and train the corresponding environment. At the end of each iteration, the agent collects different networks parameters and average them. Then, the averaged parameters push back to the copied networks. In our case, instead of vectorizing the same environments, we stack the different instances of environments and train them using the multi-processing method.

A.3 COMPARE MOBA VANILLA-MOBA ENSEMBLE WITH MODEL-BASED METHODS

To illustrate the advantage of multi-teacher knowledge distillation, we compare the Vanilla MOBA with other model-based algorithms in Figure 6.

As we can see from Figure 6, in spite of the large variance, the Vanilla-MOBA can still outperform other SOTA model-based methods in three out of six environments (i.e. Humanoid, Hopper, and Walker 2D). In the Ant environment, we can see that Vanilla-MOBA converges faster than MBPO and GrBAL. However, it cannot perform well in Half Cheetah and Pendulum. From these results, we can see that Vanilla-MOBA shows advantages in complex locomotion tasks (i.e. Humanoid,

[2]https://github.com/bulletphysics/bullet3/blob/master/examples/pybullet/gym/pybullet_envs/robot_locomotors.py

Hopper, and Walker 2D), which is hard to train by using ensemble model-based methods (GrBAL and MBPO).

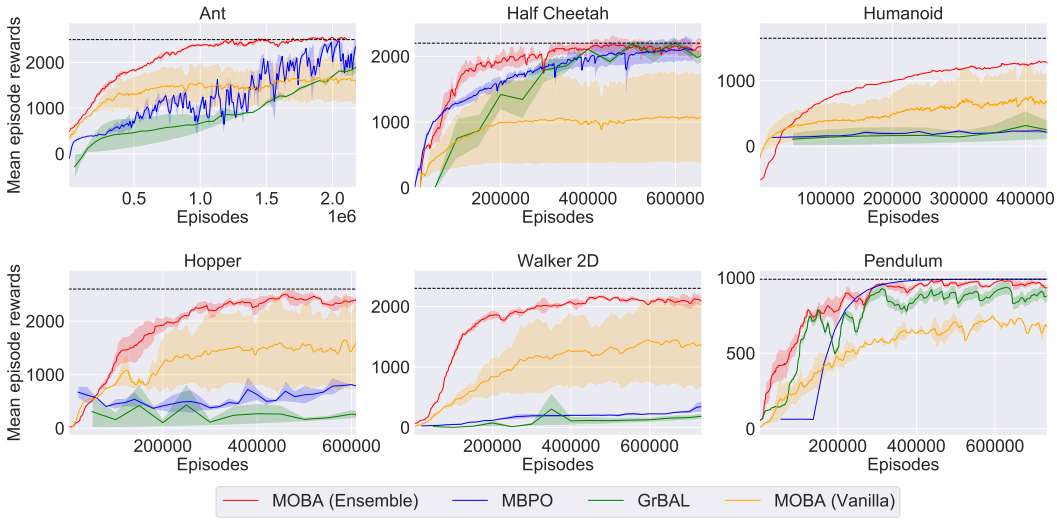

Figure 6: Learning curves of MOBA, Vanilla-MOBA, MBPO, and GrBAL

A.4 Evaluation Results on Other Environments

In this part, we evaluate the performance of the proposed MOBA method on different teacher datasets. In the RL problem, different datasets correspond to different tasks, where each task is drawn from the same task distribution. To evaluate, we adopt the environment CausalWorld (Ahmed et al., 2021) that contains various features including task generation.

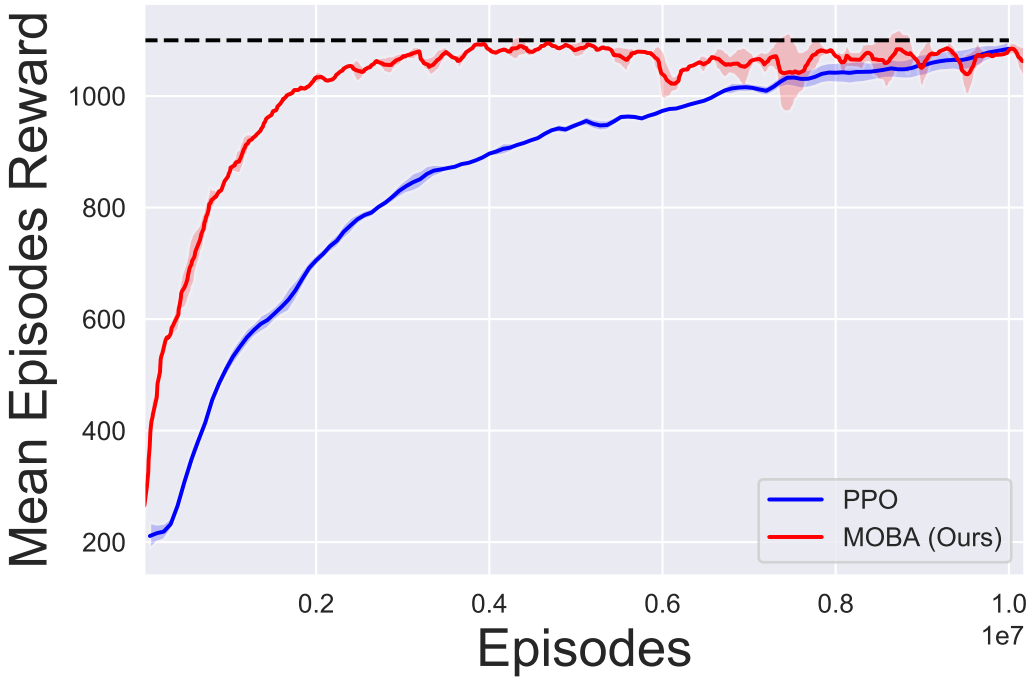

Figure 7: Learning curves of MOBA and PPO on CausalWorld Environment

The results are shown in Figure 7.

A.5 HYPER-PAREMTERS

We choose the PPO algorithm policy learning method. We list the important hyper-parameters in Table. 2. We also provide the complete code in the supplementary material.

Table 2: Hyper-parameter values for PPO training

Parameter	Value
Discount factor (γ)	0.9995
n_steps	5000
Entropy coefficiency	0
Learning rate	0.00025
Maximum gradient norm	10
Value coefficiency	0.5
Experience buffer size	1e6
Minibatch size	128
clip parameter (ϵ)	0.3
Activation function	ReLU
Optimizer	Adam