Gradual Transition From Model-Based to Model-Free Actor-Critic Reinforcement Learning

Le Chen, Yunke Ao, Kaiyue Shen, Zheyu Ye
lechen,yunkao,kashen,zheyye@ethz.ch

ABSTRACT

Model-free reinforcement learning methods could achieve very good performance but typically requires a substantial amount of data collection for convergence. On the other hand, model-based methods tend to be more sample efficient but suffer from the bias of the learned models. In this work, we aim to combine the advantages of these approaches. We design a model-based learning framework that is integrated with value learning. By gradually reducing the planning horizon, it transforms into pure model-free actor-critic learning as the training goes. The experiments show that, for environments with relatively stable dynamics and lower dimensions for action space, our proposed method reaches high performance faster compared with baselines. The code is publicly available.

1 INTRODUCTION

Reinforcement learning (RL) algorithms is a powerful framework for robots to acquire a wide range of skills from experience, for example, playing chess [15], calibrating sensors [2] and so on. RL algorithms could be divided into two main categories: model-based RL and model-free RL. Model-free RL algorithms directly optimize the policy based on the gathered interaction experience, while model-based ones additionally learn the dynamic and reward functions of the environment [1]. Both approaches have their advantages as well as limitations. Generally, model-free RL could handle arbitrary dynamical systems with minimal bias and achieve higher performance for different tasks, but typically with millions of trials for convergence. For model-based RL algorithms, much fewer samples are required for learning a decent policy since they can use each sample to better learn to predict the system dynamics and obtain near-optimal behavior by planning through the dynamics. So model-based algorithms tend to be substantially more efficient [4, 14]. However, the performance of model-based methods is limited by the planning algorithms. In addition, the existence of asymptotic bias in the learned model usually results in highly sub-optimal policy and relatively lower final performance compared with model-free RL algorithms [11].

In this project, we propose our method that could gradually transform a model-based RL training framework to a model-free actor-critic architecture such as Deep Deterministic Policy Gradient (DDPG) [16]. The whole training process goes through 3 stages: model-free pre-training, model-based imitation and model-free fine-tuning. The framework of the second stage, which is the most creative part of our work, is shown in figure 1. Our principle contributions are:

- In the second stage, our approach uses a negative value function as the terminal cost to approximate the discounted sum of future costs from the last predicted state to the end for model-based planning, which is more reasonable in theory for model-based control.
- By incorporating value learning and gradual reduction of the planning horizon, our approach further extends the strategy of model-based RL with model-free fine-tuning to actor-critic RL architecture.
- The evaluation shows that with our approach, the modified model-based learning enables better initialization for model-free actor-critic learning and leads to faster convergence and competitive final performance compared with existing methods.

https://github.com/clthegoat/GT-Mb-Mf

2 MODELS AND METHODS

2.1 Related Works

Existing works have sought to integrate model-based learning with model-free policy learning in different ways. The Dyna-type algorithms [3, 10] use the interaction experiences to train one or an ensemble of neural networks to learn the dynamics model, which are utilized to generate additional imagined samples for model-free policy learning [8]. Although higher sample efficiency is achieved, they still have limited performance for tasks with high dimensional action space since the synthetic samples would degrade with modeling errors [11]. Another algorithm MBMF [1] uses the learned model to predict the cost function of the current policy, which provides priors for the intertwined model-free optimization. However, the cost function is modeled by the Gaussian process, which is prohibitive for the higher-dimensional systems or policies. Method [5] introduce the Model-based Value Expansion (MVE) which incorporates learned dynamics models to help the value learning. By simulating a fixed short-term horizon, it succeeds to reduce the value estimation error, but does not directly help the policy learning process. Approach [14] proposed model-based RL with model-free fine-tuning (Mb-Mf), where a model-based learner is trained to imitate a model predictive control (MPC) policy and then utilized to initialize a model-free learner, but it could not be integrated with widely applied off-policy actor-critic frameworks [6, 9, 16]. Our method, Gradual Transition from Model-based to Model-free RL (GT-Mb-Mf), further extends the idea of Mb-Mf. Our approach integrates a model-based learner with the current learned target value function as the terminal cost for open-loop planning, and gradually transform it to pure model-free actor-critic learning by reducing the planning horizon.

2.2 Preliminaries

Basic reinforcement learning is modeled as a Markov decision process, where we have sets of agent states $\mathcal{S}$ and actions $\mathcal{A}$, and probability model of state transition under certain action and reward function r. In this paper, we only consider the deterministic

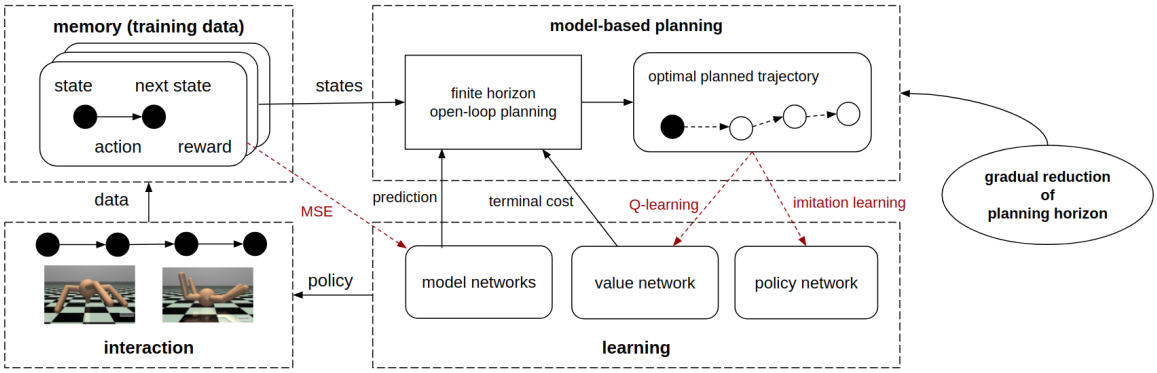

Figure 1: Framework of the second training stage of our method. The interaction data is recorded in a replay buffer, which is used for training the neural networks that learn the dynamics model and reward model. The data pairs are also extended to longer trajectories using model-based prediction and planning, which can guide policy learning and value learning. The updated policy is again utilized for generating new interaction data. The planning horizon is gradually reduced during training.

situation, so we formulate the transition as the dynamics model $f : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$. The goal at each time step t is to learn the policy that maximizes the expected cumulative reward, given by $\sum_{t'=t}^{\infty} \gamma^{t'-t} r(s_{t'}, a_{t'})$, where $\gamma \in [0, 1]$ is a discount factor that weigh more rewards in the near future.

2.3 Overview of GT-Mb-Mf

In our approach, the models for the agent to learn include the policy μ_ϕ, the Q function Q_θ, the dynamics model $\hat{f}_\alpha$ and the reward model $\hat{r}_\beta$. Besides, there are also target networks $\mu_{\phi'}$ and $Q_{\theta'}$ that update slowly following μ_ϕ and Q_θ to stabilize training [13]. The whole GT-Mb-Mf algorithm actually goes through three stages. 1. Model-free pre-training, when random actions and learned policy are first executed to quickly acquire a large amount of usable data for dynamics, reward, and value learning. 2. Model-based imitation learning, during which well-learned models are used to guide policy learning. 3. Model-free fine-tuning like normal DDPG.

The framework of the second stage is shown in Figure 1. It consists of four parts: interaction, training data generation, model-based planning, and learning. The training data comes from the simulated environment in the interaction step. They will be utilized for model learning, Q-learning, and policy learning in the learning module. To speed up the policy learning, we take advantage of the MPC controller in the finite horizon model-based planning module to provide expert actions for imitation. The updated policy is again utilized for interaction.

As the times of looping increase, the planning horizon of the MPC controller will be reduced gradually to zero following a predefined rule. In this way, finally, the whole interaction and learning become again a classical model-free framework with only value learning and policy learning, which guarantees a comparable final performance with pure model-free methods.

The whole algorithm is based on the assumption that Model learning is easier than value learning, which means the models (including the dynamics model and reward model) are well-learned much earlier than the Q function. Here, "well-learned" means sufficiently close to the true value. This assumption has been verified in previous works [5, 14]. We detail our model-based planning in Sec 2.4, design and training of models in Sec 2.5.

2.4 MPC with Value Function as Terminal Cost

In a traditional MPC framework, the open-loop planning is solving a constrained optimization problem:

$$\begin{aligned} min \quad & \sum_{t=0}^{H-1} I(s_t, a_t) + I_f(s_H) \\ s.t. \quad & s_{t+1} = f(s_t, a_t), \quad t = 0, 1, \cdots, H-1 \\ & s_t \in \mathcal{S}, a_t \in \mathcal{A}, \quad t = 0, 1, \cdots, H-1 \\ & s_0 = s, s_H \in \mathcal{S}_f \end{aligned} \tag{1}$$

where s is the current state, $I(s_t, a_t)$ denotes the stage cost, $I_f(s_H)$ is the terminal cost that approximate the "cost to go" after timestep H. $f(s_t, a_t)$ is the dynamics function. $\mathcal{S}$, $\mathcal{A}$ and $\mathcal{S}_f$ are constraints for states and actions.

In our framework, the definition of the negative value function is similar to the terminal cost of MPC, and the definition of "cost" is the opposite of "reward". Therefore, given learned dynamics model $\hat{f}_\alpha$, reward model $\hat{r}_\beta$, value model Q_θ and policy model μ_ϕ, the open-loop planning optimization actually solved in our case is:

$$\begin{aligned} max \quad & \sum_{t=0}^{H-1} \gamma^t \hat{r}_\beta(s_t, a_t) + \gamma^H Q_{\theta'}(s_H, \mu_{\phi'}(s_H)) \\ s.t. \quad & s_{t+1} = \hat{f}_\alpha(s_t, a_t), \quad t = 0, 1, \cdots, H-1 \\ & s_t \in \mathcal{S}, a_t \in \mathcal{A}, \quad t = 0, 1, \cdots, H-1 \\ & s_0 = s, s_H \in \mathcal{S}_f \end{aligned} \tag{2}$$

where H is the planning horizon, γ is the discount factor, θ' and ϕ' are parameters of target value and target policy network. Calculating the exact optimum for such a non-linear optimization problem can be difficult, so we use a simple but generally effective random shooting method. We randomly generate K candidate action sequences and use the learned dynamics and reward model to predict the corresponding state sequences and rewards, the target policy and value model to predict the terminal costs. Finally, we pick

the action sequence with the highest expected cumulative reward. Following the MPC procedure, we only keep the first action $\hat{a}$ for policy learning, which will be detailed in the next section.

Note that the introduction of the value model enables us to largely shorten the planning horizon. In the normal MPC case, better performance often needs a larger horizon. In our case, by taking into account the "cost to go" term computed by the value model, the cumulative reward can be precise even when the planning horizon is small, as long as the value model is good enough.

2.5 Algorithm Design and Training

Learning dynamics and reward functions: Dynamics and reward models are continuously trained before the transition into the third stage. We parameterize the dynamics model and reward model as neural networks: $\hat{s}_{t+1} = \hat{f}_\alpha(s_t, a_t)$, $\hat{r}_t = \hat{r}_\beta(s_t, a_t)$. Given a batch of training data $\{s_t^i, a_t^i, s_{t+1}^i, r_t^i, d_t^i\}_{i=1}^N$, where d_t^i indicate whether the terminal state is entered, both models are trained using stochastic gradient decent (SGD) to minimize the mean squared errors:

$$\epsilon_{dyn} = \frac{1}{N}\sum_{i=1}^{N} \|s_{t+1}^i - \hat{s}_{t+1}^i\|^2, \tag{3}$$

$$\epsilon_{reward} = \frac{1}{N}\sum_{i=1}^{N} \|r_t^i - \hat{r}_t^i\|^2 \tag{4}$$

A higher model prediction accuracy is fundamental to the performance of the MPC controller.

Learning value function: The Q-value function is trained using the same rule in all three stages. Given a batch of data sampled from the replay buffer $\{s_t^i, a_t^i, s_{t+1}^i, r_t^i, d_t^i\}_{i=1}^N$, the model is trained by minimizing the one-step temporal difference loss:

$$\frac{1}{N}\sum_{i=1}^{N} \|r_t^i + (1-d_t^i)\gamma Q_{\theta'}(s_{t+1}^i, \mu_{\phi'}(s_{t+1}^i)) - Q_\theta(s_t^i, a_t^i)\|^2 \tag{5}$$

which is widely applied in actor-critic algorithms. Here we adopt one-step Q-learning because it has less variance for the target and is more stable for learning compared with multi-step Q-learning.

Learning policy through imitation and policy gradient: In the second stage when dynamic and reward models have been trained relatively well and the current policy performs not as good as the MPC controller, we force the policy to imitate the MPC policy:

$$\phi \leftarrow \phi - \nabla_\phi \frac{1}{N}\sum_{i=1}^{N} \|\mu_\phi(s_t^i) - \hat{a_t}^i\|^2 \tag{6}$$

where $\hat{a_t}^i$ is the best action given by MPC controller at state s_t^i. In the first or the third training stage, we apply the gradient ascend methods to the Q-function to train the policy:

$$\phi \leftarrow \phi + \frac{1}{N}\sum_{i=1}^{N} \nabla_\phi \mu_\phi(s_t^i) \nabla_a Q_\theta(s_t^i, a)|_{a=\mu_\phi(s_t^i)} \tag{7}$$

which is named Deterministic Policy Gradient (DPG) [13]. It could be applied to off-policy data and achieve high sample efficiency.

Gradual Transition: During the model-based imitation learning status, we gradually reduce the planning horizon H, and finally, the training method will be transformed into a pure model-free actor-critic method when the planning horizon is zero. A natural idea is to reduce the H by 1 after every fixed number of interaction episodes, denoted by E_{rd}, which proves to be effective in our experiment environments. E_{rd} is a hyper-parameter to be decided. The whole algorithm is shown in Alg 1.

Algorithm 1 Gradual Transition from Model-based to Model-free Reinforcement Learning

```
//Initialization:
set maximum time step T, maximum planning steps H, number of total/ model-free pre-training/ model-based training/ reduction episodes E_m, E_p, E_mb, E_rd.
initialize models f̂_α, r̂_β, Q_θ' = Q_θ, μ_φ' = μ_φ,
initialize the reply buffer D with random interaction experiences

for episode=0 to E_m do
    //Interaction:
    reset the state and environment
    for t=0 to T do
        get current state s_t
        apply a_t = μ_φ(s_t), receive r_t, arrive at s_{t+1}, set d_t = done
        record (s_t, a_t, r_t, s_{t+1}, d_t) in D
        if d_t then break
        end if
    end for

    //Training:
    randomly sample {s_t^i, a_t^i, s_{t+1}^i, r_t^i, d^i}_{i=0}^N from D
    train f̂_α, r̂_β, by applying SGD to (3)-(4)
    train Q_θ, by applying SGD to (5)
    if E_p<episode<E_p+E_mb and H > 0 then        ▷ imitation learning
        compute {â_t^i}_{i=0}^N from {s_t^i}_{i=0}^N by solving (2)
        train μ_φ by applying (6)
    else                                          ▷ model-free training
        train μ_φ by applying (7)
    end if
    update target Q_θ', μ_φ'

    //Gradual Transition:
    if episode % E_rd = 0 and episode ≠ 0 then
        H = H − 1
    end if
end for
return f̂_α, r̂_β, Q_θ, μ_φ
```

3 RESULTS

In this section, we first evaluate different design decisions for model-based RL in the imitation learning stage. Then we compared performance of our methods with different baselines for benchmark tasks in the OpenAI Gym, including pendulum ($\mathcal{S} \in \mathbb{R}^3, \mathcal{A} \in \mathbb{R}$), lunar-lander-continuous ($\mathcal{S} \in \mathbb{R}^8, \mathcal{A} \in \mathbb{R}^2$), half-cheetah ($\mathcal{S} \in \mathbb{R}^{23}, \mathcal{A} \in \mathbb{R}^6$), hopper ($\mathcal{S} \in \mathbb{R}^{17}, \mathcal{A} \in \mathbb{R}^3$) and walker2d ($\mathcal{S} \in \mathbb{R}^{17}, \mathcal{A} \in \mathbb{R}^6$).

3.1 Experiments on MPC Controller Design

The design decisions for model-based controllers include the number of model-free pre-training episodes E_p, number of sampled trajectories for action selection of model-based controller K, whether to include terminal costs, and the planning horizon H. In previous

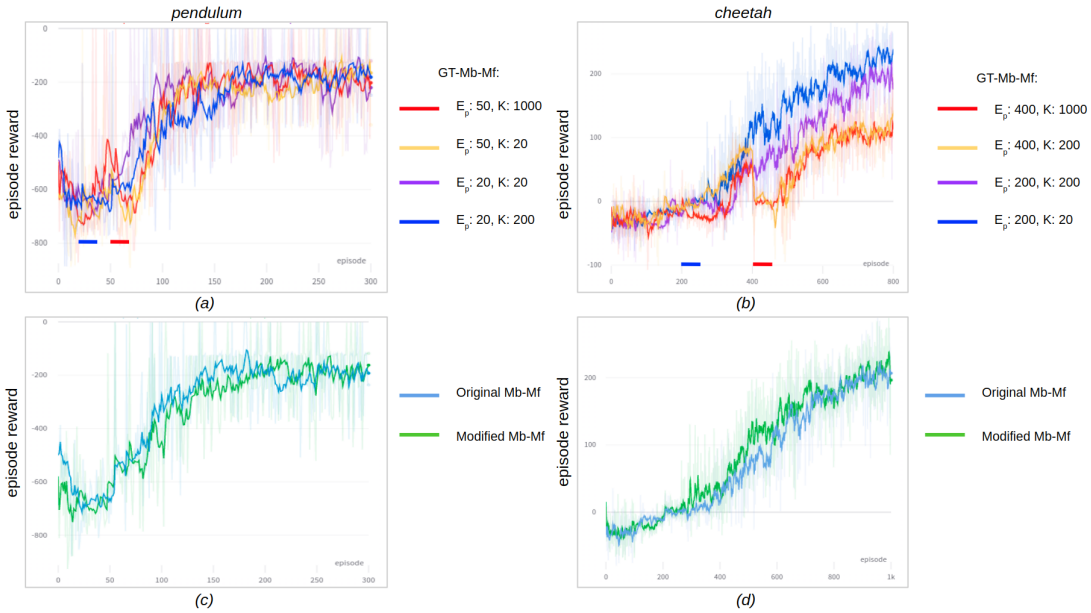

Figure 2: Experiments for analyzing the model-based controller. (a)-(b) are the performance of GT-Mb-Mf with different numbers of episodes for model-free pre-training and different shooting numbers. The blue and red line segment on the episode axis indicate the imitation learning stages for different agents. It could be noticed that there exists sudden decrease of performance at episode 50 in (a) and episode 400 in (b) for the red and yellow lines, which illustrates the importance of choosing proper E_p. (c)-(d) investigate whether including the value function as the terminal cost for the planning of MPC would improve the performance of the model-based controller. The number for model-free pre-training are kept the same, which are 20 and 200 respectively for pendulum and cheetah. The numbers of episodes for model-based training are all 25 and 50 respectively for pendulum and halfcheetah.

works [14], it has been verified that increasing planning horizon does not improve the performance much due to the bias of model learning. Therefore in this paper, we mainly investigate the former three design decisions.

Number of episodes for model-free pre-training: This design decision reflects when it is believed that the models are learned well enough for the agent to utilize the model-based controller to guide policy learning. From Figure 2 (a)-(b), it is shown that even with higher E_p, the performance of the policy during the imitation learning status does not improve much. The reason could be the existence of bias in the model and the inefficiency of shooting-based optimization for high dimensional action space. Therefore E_p should not be set to be too large.

Shooting number: This design decision is chosen by trading off computational cost and performance. Intuitively, higher K could give better performance for the model-based controller. However, as is shown in Figure 2 (a)-(b), the performance does not improve as much as is expected when K increase. This could also be caused by the existence of bias in dynamic and reward networks.

Value function as terminal cost: According to Figure 2 (c)-(d), the modified Mb-Mf (green) by additionally using negative value function as the terminal cost for open-looped planning may keep nearly the same performance as the original Mb-Mf (blue), even though it is more reasonable in theory. This is possibly because when E_p is relatively small, the target value network is still not sufficiently updated, which make less contribution to planning. Therefore, we could also substitute the original Mb-Mf with this modified Mb-Mf as the baseline to compare with our proposed algorithm.

3.2 Performance Results on Benchmark Tasks

The baselines to be compared with our method are MVE, DDPG, and modified Mb-Mf. To ensure a fair comparison between GT-Mb-Mf and modified Mb-Mf, we keep the starting episode as well as the total number of episodes for the imitation learning stage to be the same between the two methods. As for MVE, it is found that only when the imaginary horizon is 2 will the agent perform relatively well in pendulum and half-cheetah. However, for other environments with early termination conditions, the termination of imagined trajectories can not be predicted with only the learned dynamics and reward networks, which makes MVE perform poorly. Therefore we exclude the performance results of MVE in these environments.

As is shown in Figure 3 (a)-(c), for environments with relatively stable dynamics and lower dimensions for action space such as half-cheetah and pendulum, GT-Mb-Mf reaches the highest performance earlier than DDPG. One reason could be that during imitation learning stage, the policy quickly imitate a sub-optimal policy which has less difference to the optimal policy, and more data interacted with this sub-optimal policy is recorded in the replay buffer. This results in faster improvement of policy in the model-free fine-tuning stage when the policy gradient is applied to the sub-optimal policy based on plenty of data interacted with this policy. This phenomenon is also shown in [14] with Mb-Mf. Besides, compared with Mb-Mf, our model-based controller is even more sample efficient. Because as the planning horizon is continuously reduced, the dimensions of the searching spaces for the shooting-based controller also decreases, which improves the effect of imitation learning and makes

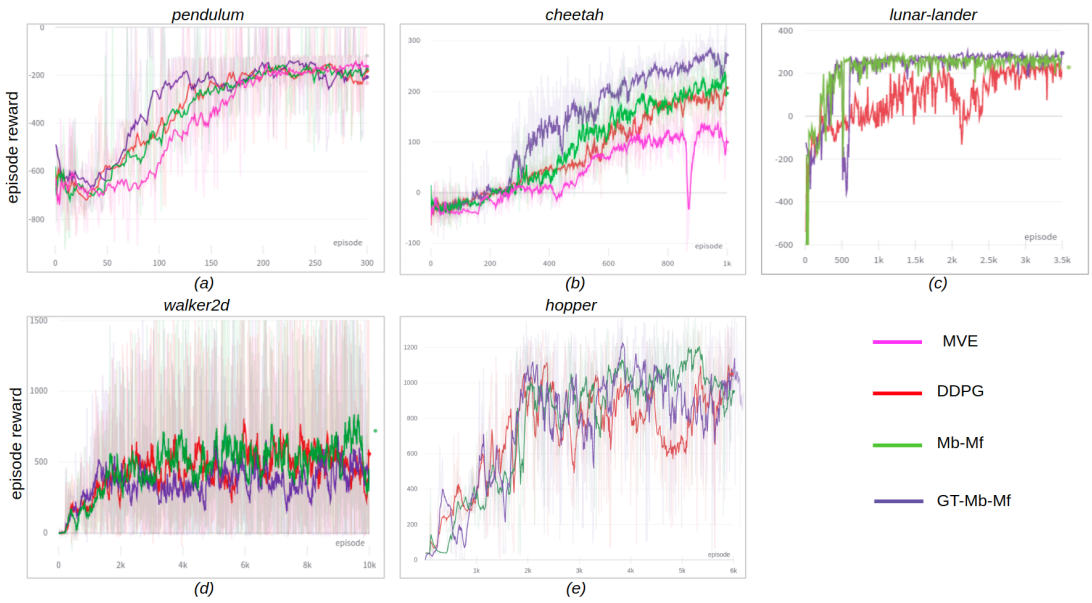

Figure 3: Learning curves for GT-Mb-Mf, MVE, DDPG and Mb-Mf on pendulum (a), cheetah (b), lunar-lander (c), walker2d (d) and hopper (e). The curves have been smoothed using exponential moving average with factors from 0.8-0.9. The dynamics and reward networks both have one hidden layer with 128 units, and are trained with 1×10^{-3} learning rate. For the model-free training stages of all the agents, we adopt the hyperparameters recommended in the original paper of DDPG. During the imitation learning stage, the learning rates for actor and critic are 1×10^{-4} and 1×10^{-3} respectively. Based on the results from Sec 3.1, we set planning horizon to be 5 and shooting numbers to be 20 for all GT-Mb-Mf and Mb-Mf agents. E_p are set as the number of episodes when the performance of the policy of the model-free learner start to reach the performance of the model-based MPC controller.

our performance converge even faster. For other environments with unstable dynamics or high dimensional action spaces such as hopper and walker2d, our GT-Mb-Mf could perform at least as good as DDPG without decreasing the final performance according to Figure 3 (d)-(e).

4 DISCUSSION

We have evaluated our approach on a range of challenging simulated tasks. Although our model-based controller cannot always reach high rewards on its own, it could improve sample efficiency when it is applied to initialize a model-free learner, compared with pure model-free algorithms. Besides, our method further extend Mb-Mf for actor-critic RL algorithms such as DDPG by including the value function in both learning and planning of the model-based learning status. The transition from model-based imitation learning to model-free training is also smoother by gradually reducing the planning horizon and results in faster convergence compared with Mb-Mf according to the experiment results.

In spite of the effectiveness and efficiency of our GT-Mb-Mf algorithm, currently, there still exist some limitations. Although the improvement of the performance of model-free learner is faster after the imitation learning status, the improvement during the imitation status is limited. This could be addressed by using a better model-based controller or a more effective policy update algorithm utilizing the learned dynamics model and reward model. Besides, the final performance of the algorithms is restricted by the model-free learner we currently adopt (DDPG). Higher final performance could be achieved when this idea is extended to more powerful actor-critic model-free algorithms.

Our GT-Mb-Mf method has wide potential applications. For example, similar ideas could be used to speed up other actor-critic RL algorithms such as soft actor critic (SAC) and twin delayed DDPG (TD3) [6, 9]. As DDPG has very limited performance in environments with higher state and action dimensions such as ant and humanoid, we could include experiments for these tasks in our future works when we extend our method for these more powerful algorithms. Except for model-free learner, various existing model-based RL learners [7, 12] could also be adopted in this framework to further improve the sample efficiency.

5 SUMMARY

We present GT-Mb-Mf, which begins with model-free learning, then goes through a model-based framework integrated with the current learned target value function as the terminal cost for open-loop planning and gradually transform to pure model-free actor-critic learning by reducing the planning horizon. The experiment results show that, for environments with relatively stable dynamics and lower dimensions for action space, our method converges faster and reaches the highest final performance than other algorithms. For other environments with unstable dynamics or high dimensional action space, our method could perform at least as good as DDPG. An interesting avenue for further improvement is to integrate our approach with other state of the art actor-critic model-free learners [6, 9]. Additionally, for model-based controller design, instead of random shooting, extensions to other planning algorithms that are more efficient for high dimensional and unstable dynamics would further improve its generality, and this is left for future work.

REFERENCES

[1] Somil Bansal, Roberto Calandra, Kurtland Chua, Sergey Levine, and Claire Tomlin. 2017. Mbmf: Model-based priors for model-free reinforcement learning. *arXiv preprint arXiv:1709.03153* (2017).

[2] Le Chen, Yunke Ao, Florian Tschopp, Andrei Cramariuc, Michel Breyer, Jen Jen Chung, Roland Siegwart, and Cesar Cadena. 2020. Learning Trajectories for Visual-Inertial System Calibration via Model-based Heuristic Deep Reinforcement Learning. In *Proceedings of the 4th Conference on Robot Learning (CoRL)*.

[3] Ignasi Clavera, Jonas Rothfuss, John Schulman, Yasuhiro Fujita, Tamim Asfour, and Pieter Abbeel. 2018. Model-based reinforcement learning via meta-policy optimization. *arXiv preprint arXiv:1809.05214* (2018).

[4] Marc Peter Deisenroth, Gerhard Neumann, and Jan Peters. 2013. *A survey on policy search for robotics*. now publishers.

[5] Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I Jordan, Joseph E Gonzalez, and Sergey Levine. 2018. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101* (2018).

[6] Scott Fujimoto, Herke Van Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477* (2018).

[7] Yarin Gal, Rowan McAllister, and Carl Edward Rasmussen. 2016. Improving PILCO with Bayesian neural network dynamics models. In *Data-Efficient Machine Learning workshop, ICML*, Vol. 4. 34.

[8] Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. 2016. Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning*. 2829–2838.

[9] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290* (2018).

[10] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. 2018. Model-ensemble trust-region policy optimization. *arXiv preprint arXiv:1802.10592* (2018).

[11] Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. 2019. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057* (2019).

[12] Sergey Levine and Vladlen Koltun. 2013. Guided policy search. In *International Conference on Machine Learning*. 1–9.

[13] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[14] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. 2018. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 7559–7566.

[15] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* 362, 6419 (2018), 1140–1144.

[16] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic policy gradient algorithms.