Neurocomputing xxx (xxxx) xxx

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Deterministic policy optimization with clipped value expansion and long-horizon planning

Shiqing Gao [a,b], Haibo Shi [c], Fang Wang [d], Zijian Wang [e], Siyu Zhang [a], Yunxia Li [b,*], Yaoru Sun [a,*]

[a] Department of Computer Science and Technology, Tongji University, Shanghai 201804, China
[b] Department of Neurology, Tongji Hospital, School of Medicine, Tongji University, Shanghai 200092, China
[c] School of Statistics and Management, Shanghai University of Finance and Economics, Shanghai 200433, China
[d] Department of Computer Science, Brunel University, Uxbridge UB8 3PH, United Kingdom
[e] School of Computer Science and Technology, Donghua University, Shanghai 200000, China

A R T I C L E I N F O

Article history:
Received 15 November 2021
Revised 20 January 2022
Accepted 3 February 2022
Available online xxxx
Communicated by Zidong Wang

Keywords:
Model-based reinforcement learning
Policy gradient
Sample efficiency
Planning
Imitation learning

A B S T R A C T

Model-based reinforcement learning (MBRL) approaches have demonstrated great potential in handling complex tasks with high sample efficiency. However, MBRL struggles with asymptotic performance compared to model-free reinforcement learning (MFRL). In this paper, we present a long-horizon policy optimization method, namely model-based deterministic policy gradient (MBDPG), for efficient exploitation of the learned dynamics model through multi-step gradient information. First, we approximate the dynamics of the environment with a parameterized linear combination of an ensemble of Gaussian distributions. Moreover, the dynamics model is equipped with a memory module and trained on a multi-step prediction task to reduce cumulative error. Second, successful experience is used to guide the policy at the early stage of training to avoid ineffective exploration. Third, a clipped double value network is expanded in the learned dynamics to reduce overestimation bias. Finally, we present a deterministic policy gradient approach in the model that backpropagates multi-step gradient along the imagined trajectories. Our method shows higher sampling efficiency than the state-of-the-art MFRL methods while maintaining better convergence performance and time efficiency compared to the SOAT MBRL.

1. Introduction

Reinforcement learning (RL) can be divided into two classes: model-free RL (MFRL), which learns a policy directly from the interaction with the environment without knowing its dynamics, and model-based RL (MBRL), which optimizes a policy with a learned dynamics model of the environment. MFRL has shown an excellent capability to handle complex tasks in unknown environments, including the game of Go [1,2], Atari games [3,4], control tasks [5–7], and images segmentation [8]. However, the low sampling efficiency of MFRL limits its applications to control problems in the real world, especially for tasks with high data collection costs, such as robotics [9–11]. Compared with model-free methods, MBRL requires orders-of-magnitude fewer samples [12–14] since the simulation data can be gathered without interacting with the environment. Another advantage of MBRL is that the value function can be estimated more accurately through the returns of long-horizon rollouts in the dynamics model [15,16].

MBRL algorithms improve the sampling efficiency roughly through three mechanisms. First, Dyna-style algorithms [13,17] use the learned dynamics to generate imaginary data and learn policy through a model-free approach, which do not exploit the gradient information provided by the dynamics. Second, shooting algorithms plan action sequences based on the trajectories sampling from the learned model, such as random shooting [18,19], or CEM [20], which have low convergence performance as multiple complete Monte Carlo experiments are required. Contrary to the above two categories, model-based policy gradient algorithms [12,21–23] exploit the differentiability of the model and calculate the gradient of the objective function with respect to the policy, which can be optimized directly, but few of them take into account the gradient information of multiple steps.

Learning a sufficiently accurate model for planning has proven challenging. Based on the limited states visited from the exploration, it is difficult to fit a model that is uniformly applicable in the state space. Furthermore, the distribution from which the samples are drawn to learn the model is not stationary, but changes with the behavior of the agent [24]. The aleatoric uncertainty and epistemic uncertainty contained in the environment raise

* Corresponding authors.
E-mail addresses: doctorliyunxia@163.com (Y. Li), yaoru@tongji.edu.cn (Y. Sun).

https://doi.org/10.1016/j.neucom.2022.02.022
0925-2312/© 2022 Elsevier B.V. All rights reserved.

another dilemma [25]. Defective models pose the problem of cumulative errors during the rollout process [26,27].

RL requires extensive trial-and-error exploration in the environment, but much of this is repetitive and ineffective in the early training. In contrast, imitation learning can reduce redundant exploration guided by expert demonstrations. Behavioral cloning (BC) [28] methods learn a state-to-action mapping directly from demonstrations. Inverse reinforcement learning (IRL) [29] recovers a reward signal from the expert behavior and constructs RL methods to learn the policy. However, the learner can never outperform the expert with the cloned behavior in imitation learning.

In this paper, we propose a model-based deterministic policy gradient (MBDPG) algorithm with an off-policy actor-critic framework [30]. We use a linear combination of multiple Gaussian distributions to approximate the dynamics model of the environment. The LSTM [31] network is used as the memory module of the dynamics model to encode the history state information. The model is trained with multi-step prediction to reduce the cumulative error. Successful experience is provided in the early training to increase exploration efficiency. We expand the clipped value estimation in the model, which provides a stable critic model with a low overestimation bias. Finally, we derive the model-based deterministic policy gradient by linking the dynamics model, the value function, and the policy to form an end-to-end pathway for training. As a result, the policy gradient can be backpropagated through rollout trajectories.

The main contributions of our work are summarized as follows:

(1) Mixture Gaussian Network for multi-step prediction. Deep neural networks are utilized to parameterize linear combinations of an ensemble of Gaussian distributions, which effectively captures the stochasticity and uncertainty of the environment. In contrast to other probabilistic ensemble methods, our approach constructs a recurrent memory network to encode historical information, and trains the model on a multi-step prediction task to reduce the cumulative error.
(2) Learning from successful experience. In early training, our approach speeds up the convergence of policy by cloning the behavior of the expert and learning critic model to interpret the intention of the tasks.
(3) Long horizon planning with clipped value expansion and deterministic policy gradient. We exploit multi-step expansion based on the dynamics model for both critic estimation and policy optimization. In addition, the model-based deterministic policy gradient in our method avoids integration over the action space in contrast to stochastic policy. We backpropagate multi-step gradient along the imaginary trajectories to obtain faster convergence compared to one-step policy gradient methods in MBRL. The experiments demonstrate that an appropriate growth of the horizon can accelerate the convergence of the algorithm.

2. Related work

2.1. Dynamics model

The learning of dynamics model is essentially a supervised learning problem. PILCO [12] uses Bayesian non-parametric and probabilistic Gaussian Processes (GPs) [32] to estimate the dynamics model, but it is difficult to apply to high-dimensional tasks. Neural networks (NNs) are used to parameterize deterministic models of high-dimensional state space in [33] due to their powerful representation capabilities. However, deterministic models cannot represent stochasticity and uncertainty in complex environments. PETS [34] uses an ensemble of probabilistic networks to sample the trajectory and incorporates the uncertainty into the learned dynamics model. The recurrent neural network is used to integrate previous observation information as prior knowledge in [35]. AMRL [36] uses aggregators to increase the robustness of LSTM against noise, to maintain the information gradient in the long horizon. PlaNet [37] builds a recurrent state-space model with both deterministic and stochastic components. Our method is similar to World Models [38], in the sense that we both use a Mixture Density Network combined with an RNN network, but in our method, the dynamics model is trained for multi-step prediction.

2.2. Policy optimization

Heuristic algorithms can quickly obtain near-optimal solutions to an optimization problem from the search space. SLEPSO [39] proposes a novel path planning method for intelligent robots based on non-homogeneous Markov chain and differential evolution to balance local search and global search. RODDPSO [40] presents a distributed approach to introduce randomly occurring time-delays to expand the search space and improve the performance of getting rid of the local optima dilemma. To alleviate the premature convergence problem and escape from the local optima, DNSPSO [41] proposes a distance-based dynamic neighborhood to integrate the neighborhood information, and an adaptively adjusted switching learning strategy to close the global optimum.

Random Shooting (RS) methods [19,42] sample several random action sequences from a policy, and perform these sequences in the learned dynamics model. The agent selects the optimal action sequence with the highest episode return and executes the first action in the real environment. However, action sampling in RS lacks effective planning. In the CEM algorithm [20], the most rewarding actions are sorting out according to long-term rewards to obtain a better solution. In PETS [34], the policy is trained using the RS method and the CEM method respectively. Our algorithm differs from the PETS, in the meaning that we use end-to-end gradient information to update the policy instead.

The dynamics model in ME-TRPO [17] and SLBO [43] is used to generate imagination data, which is employed as a supplement to the environment data to optimize policy with TRPO [44]. MBPO [13] uses an ensemble of probabilistic neural networks to approximate the dynamics of the environment, which is similar to PETS, but uses a MFRL method, SAC [7], to update the policy. In our approach, multi-step planning can be performed in the learned model. Compared to MBPO, which utilizes one-step gradient information, our method can roll out state sequences and obtain more gradient information within the foresight horizon.

The policy gradient methods in MFRL have demonstrated their strong asymptotic performance. SAC [7,45] maximizes the entropy to encourage the agent to explore the environment. Under the actor-critic framework, deterministic policy gradient (DPG) [46] and Deep DPG (DDPG) [47] optimize the deterministic policy with the formulation of deterministic policy gradient. DPG proves that the deterministic policy gradient algorithm is significantly more effective than the stochastic policy gradient algorithm in the high-dimensional action space. TD3 [48] mentions that overestimation bias is a common problem in the Q learning process, and proposes a clipped double Q-learning algorithm to overcome the overestimation bias. We give the model-based derivation of the deterministic policy gradient. Compared with the model-free policy gradient methods, our method uses the model to carry out multi-step gradient backpropagation, which yields much higher sampling efficiency than model-free methods.

Prior works have explored either directly planning the actions or incorporating policy gradient methods into MBRL to speed up the convergence. In PILCO [12] algorithm, the policy is optimized by the gradient sampled from model rollouts. Since it learns the dynamics model through the Gaussian process, when the dimensionality of the task increases, the complexity of the algorithm will increase exponentially. SVG [23] uses reparameterization to introduce noise into the policy and model and makes the backpropagation of stochastic sampling possible. MAAC [22] and Dreamer [49] perform a latent space representation of the input image information. The policy gradient is estimated by backpropagating its gradient through the trajectory to learn a stochastic policy in MAAC. Our method proposes a model-based deterministic policy gradient method, which does not require complicated sampling in the action space in contrast to stochastic policy gradient. In addition, the learned model is used to correct the value estimations in our algorithm.

3. Preliminaries

In this section, we describe the reinforcement learning problem in detail and explain the notation used in this paper.

We regard a standard RL problem as a Markov decision process (MDP) [50], which is defined by a tuple $(S, A, p, R, \gamma, \rho_0)$. Here, S and A denote the state and action spaces, respectively. R represents the reward function, ρ_0 represents the initial state distribution and $\gamma \in (0, 1)$ is the discount factor. Inputting the current state s_t and action a_t, the state transition model $s_{t+1} \sim p(s_t, a_t)$ returns the next state s_{t+1}, and the reward function $r_{t+1} \sim R(s_t, a_t)$ returns the immediate reward r_{t+1}. The goal of reinforcement learning is to obtain an optimal policy that maximizes the expected reward:

$$R_t = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right] \tag{1}$$

MBRL algorithms learn the dynamics model from the empirical data $\tau = (s_0, a_o, s_1, a_1, \ldots)$ by interacting with the environment. We use a parameterized function $s_{t+1} \sim f_s(s_t, a_t)$ to approximate the state transition function $p(s_t, a_t)$ of the environment, Similarly, a parametric function $r_{t+1} \sim f_r(s_t, a_t)$ is used to approximate the ground truth reward function $R(s_t, a_t)$. H represents the length of the horizon in the model.

In actor-critic methods, the actor model and the critic model are updated alternately. We learn an action-value function $q_t = Q(s_t, a_t)$ that approximates the expected return conditioned on a state s_t and action a_t. Then, the learned critic model is used to optimize a policy $a_t = \mu(s_t)$.

Generally, the action-value function, Q, is evaluated by iteratively minimizing the Bellman residual in Q-learning method:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right] \tag{2}$$

Where $r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)$ is called TD error, γ here is the discount factor which controls the contribution of rewards further in the future, and α is the step size. In Q-learning, a greedy strategy is employed to update the Q function, which may cause large estimates of the value function, called overestimation bias.

For a large state space, DQN constructs function approximators with neural networks parameterized by ψ. Given state and action as the input, the Q-Network outputs Q-value estimation. The optimization objective is to minimize the loss:

$$J_Q(\psi) = E\left[\left(Q(s_t, a_t|\psi) - (r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}|\psi'))\right)^2\right] \tag{3}$$

The target value is given by the target network $Q(s_t, a_t|\psi')$ to maintain a fixed objective over multiple iterations.

In actor-critic methods, the actor-network $\mu(s_t|\theta)$ is trained to maximize the outputs of the learned Q-networks through the DPG algorithm:

$$\nabla_\theta J(\theta) = E_{s\sim\rho^\mu}\left[\nabla_\theta \mu(s|\theta) \nabla_a Q(s, a)|_{a=\mu(s|\theta)}\right] \tag{4}$$

The off-policy approach is applied in the actor-critic framework, sampling random mini-batches from an experience replay buffer to reduce the correlation among the samples.

4. Methods

Our approach focuses on modeling dynamics for long-term prediction and exploiting multi-step policy gradient to improve convergence performance. In this section, we first present the Mixture Gaussian Network to approximate the dynamics for multi-step prediction. Second, we use an imitation learning approach to accelerate the initial period of exploration. Finally, we present a model-based deterministic policy gradient approach to update the policy in an end-to-end manner.

4.1. Dynamics model learning

Before planning, we need to construct a simulation of the environment from the agent trajectories $\tau = (s_0, a_o, s_1, a_1, \cdots)$, which approximates the ground truth sufficiently well over a long horizon. The computation graph is shown in Fig. 2.

4.1.1. Mixture Gaussian network

We consider sample sequences $\{s_t, a_t, r_t\}_{t=1}^T$ with time step t, state s_t, continuous action a_t, and scalar reward r_t. A Mixture Gaussian Network is used to predict the next state $\hat{s}_{t+1}$ and reward $\hat{r}_{t+1}$. Stochastic dynamics models can capture part of the aleatoric uncertainty of the environment, but a single probability density function still cannot accurately fit all situations, since the stochasticity of the environment is not unimodal. Therefore, we define an ensemble of Gaussian distributions $\{(\mathcal{N}(\mu_1, \sigma_1), \alpha_1), \ldots, (\mathcal{N}(\mu_M, \sigma_M), \alpha_M)\}$. Each Gaussian distribution of the ensemble is parameterized by a probabilistic feed-forward neural network, which outputs the mean μ_i, variance σ_i, and corresponding weights α_i of the distribution. Each Gaussian distribution represents a single dynamics model, which generates the predicted state $\hat{s}^i_{t+1}$ and reward $\hat{r}^i_{t+1}$ by sampling in the probability density:

$$p(\hat{s}^i_{t+1}, \hat{r}^i_{t+1}|s_t, a_t) = \mathcal{N}(\mu_i, \sigma_i) \tag{5}$$

By summing the predicted values of Gaussian distributions with adaptive weights, the Mixture Gaussian Network can output a more accurate state $\hat{s}_{t+1}$ and reward $\hat{r}_{t+1}$ in the long term:

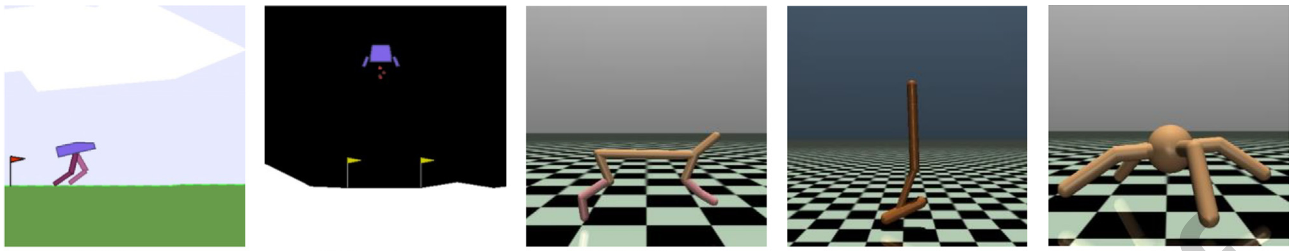
$$p(\hat{s}_{t+1}, \hat{r}_{t+1}|s_t, a_t) = \sum_{i=1}^{M} \alpha_i \left(\hat{s}^i_{t+1}, \hat{r}^i_{t+1} \sim \mathcal{N}(\mu_i, \sigma_i)\right) \tag{6}$$

To mitigate the partial observability, LSTM is used as the memory component of the Mixture Gaussian Network, which enables the agent to utilize historical data as prior knowledge. We express the memory model as $h_{t+1} = f(h_t, s_t, a_t)$, which encodes the sequential inputs into the hidden state h_t. The dynamics model contains a state model $\hat{s}_{t+1} \sim f_s(s_t, a_t, h_t)$, to predict the next state, and a reward model $\hat{r}_{t+1} \sim f_r(s_t, a_t, h_t)$, to predict the reward. In more detail, the LSTM network outputs the next hidden state h_{t+1} conditioned on s_t, a_t and h_t, and the multilayer fully connected network maps the next hidden state h_{t+1} into the parameters $\{\mu_1, \sigma_1, \alpha_1, \cdots, \mu_M, \sigma_M, \alpha_M\}$ of the ensemble of Gaussian distributions. According to Eq. (6), we obtain the dynamics model in our work:

Hidden state model:

$$h_{t+1} = f_h(h_t, s_t, a_t) \tag{7}$$

Gaussian distributions:



(a) Walker (b) LunarLander (c) HalfCheetah (d) Hopper (e) Ant

Fig. 1. Continuous control tasks in OpenAI Gym and MuJoCo used in our expertiments. Several challenges in RL are included in these environments, such as sparse rewards, high-dimensional spaces, many degrees of freedom, and complex dynamics. Our approach shows great generalization in different environments.

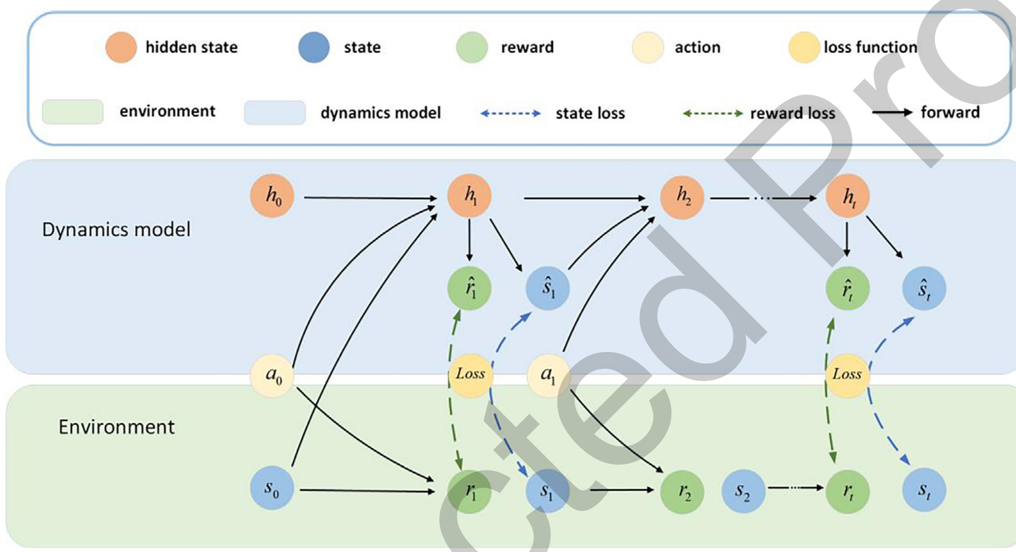


Fig. 2. Computation graph of learning dynamics model. Given the initial state, the same sequence of actions is executed respectively in the model and in the environment. Solid lines represent forward propagation, dashed lines with double arrows denote the loss calculation between the predictions and the ground truth data.

$$(\mu_i, \sigma_i, \alpha_i)_{i=1}^{M} = f_g(h_t, s_t, a_t) = g(h_{t+1}) = g(f_h(h_t, s_t, a_t)) \tag{8}$$

State model:

$$\hat{s}_{t+1} = f_s(s_t, a_t, h_t) = p(\hat{s}_{t+1}|s_t, a_t, h_t) = \sum_{i=1}^{M} \alpha_i \left(\hat{s}^i_{t+1} \sim \mathcal{N}(\mu_i, \sigma_i)\right) \tag{9}$$

Reward model:

$$\begin{aligned}\hat{r}_{t+1} &= f_r(s_t, a_t, h_t) = p(\hat{r}_{t+1}|s_t, a_t, h_t) \\ &= \sum_{i=1}^{M} \alpha_i \left(\hat{r}^i_{t+1} \sim \mathcal{N}(\mu_i, \sigma_i)\right)\end{aligned} \tag{10}$$

4.1.2. Multi-step prediction training

We include the multi-step prediction loss in the overall training goal, in which the error information at each step is utilized to modify the parameters. During dynamics training, we execute the same action sequences $\{a_0, a_1, \cdots, a_{H-1}\}$ in the simulation model as in the ground-truth trajectory $\tau = \{s_0, a_0, r_1, s_1, a_1, r_2, \cdots, s_{H-1}, a_{H-1}, r_H, s_H\}$ from an initial state s_0. The objective function contains the Maximum Likelihood Estimation of the Gaussian distribution and the error of the predicted reward. The training objective aims to maximize the probability of the ground-truth state s_t under the learned mixture Gaussian distribution $\mathcal{N}(\mu_i, \sigma_i)_{i=1}^{M}$, and minimizes the mean-squared error (MSE) between the predicted and true rewards:

$$J_{f_s f_r} = \sum_{t=1}^{H} (r_t - \hat{r}_t)^2 - \sum_{t=1}^{H} \sum_{i=1}^{M} \alpha_i \log p_i(s_t|\mu_i, \sigma_i) \tag{11}$$

Where $p_i(s_t|\mu_i, \sigma_i)$ represents the probability of the ground truth s_t under the i^{th} Gaussian distribution. The greater the likelihood is, the closer the learned mixture Gaussian distribution is to the state distribution of the real environment. Compared with one-step training, multi-step training significantly reduces the cumulative error for prediction.

Note that in our work, a deterministic dynamics model is employed during planning. Sampling from Gaussian distributions in stochastic dynamics makes the model non-differentiable, which prevents our algorithm from training the policy end-to-end. The reparameterization trick can maintain the differentiability of the dynamics model but introduces additional computation. In stochastic methods, the learned Gaussian distributions are constructed by linear transformation with the noise drawn from a standard normal distribution:

$$\hat{s}^i_{t+1} = \mu_i + \sigma_i * \varepsilon, \varepsilon \sim \mathcal{N}(0, 1) \tag{12}$$

In contrast, the deterministic model avoids sampling operations when predicting the next state and reward and allows gradient backpropagation along the model. As the training of the stochastic dynamics model proceeds, the variance of the Gaussian distribution gradually converges. While using the model, we directly use the weighted sum of the means of the Gaussian distributions as the predicted value:

$$\hat{s}_{t+1}, \hat{r}_{t+1} = \sum_{i=1}^{M} \alpha_i \mu_i \tag{13}$$

4.2. Learning from success with imitation learning

The following shows the initialization of the critic model and actor model with successful demonstrations.

Expert data is collected in advance using SOTA MFRL methods. We adopt TD3 [48] to train expert behavior and gather the trajectories of completed tasks in the environment.

Deep neural networks are employed to represent the actor and critic models with parameters θ and ψ, respectively. To learn the intent of the expert, we update the critic model with successful experience. Here, the critic network $q_t \sim Q(s_t, a_t|\psi)$ is updated with a temporal difference method. The optimization objective of the critic model in a time step t is to minimize the error between the target value $r_{t+1} + \gamma Q(s_{t+1}, a_{t+1})$ and the estimated value $Q(s_t, a_t)$:

$$J_Q(\psi)_t = E\left[(Q(s_t, a_t) - (r_{t+1} + \gamma Q(s_{t+1}, a_{t+1})))^2\right] \tag{14}$$

The updated critic model outputs estimations with a small variance, which facilitates robustness for later learning and prevents abrupt changes.

The actor-network $a_t \sim \mu(s_t|\theta)$ is updated with behavioral cloning. The problem is converted into a supervised learning problem by solving a regression task to obtain an initial policy.

$$J_\mu(\theta) = E\left[(\mu(s_t) - a_t)^2\right] \tag{15}$$

In most cases, we cannot guarantee that the expert policy is optimal. In our approach, only successful trajectories for completing tasks are collected as expert demonstrations. Although expert behavior is defective, it still provides enough learning signals for the initialization of the actor and critic model. Guided by expert data, a roughly accurate policy and value function that points to the endpoint of the task can be quickly obtained, along with adequate empirical data that is effectively explored. However, the learned critic and actor model have a large bias and poor generalization due to data limitation. In the following, with the guidance of the initialized actor and critic model, we further optimize the policy in the dynamics model to obtain the optimum that outperforms the expert.

4.3. Long horizon planning

In this section, we show a model-based deterministic policy gradient method, MBDPG, that backpropagates multi-step gradient along the imaginary trajectories to update the deterministic policy under the actor-critic framework. The architecture of MBDPG is shown in Fig. 3.

4.3.1. Critic model with clipped value expansion

In our approach, a critic model is constructed to estimate the action-value function Q. Compared to MFRL, the bias of the value function can be reduced by value expansion in the learned dynamics. We roll out H steps in simulation dynamics to obtain an imaginary trajectory $\{a_0, \hat{s}_1, a_1, \hat{s}_2, a_2 \ldots, \hat{s}_H\}$ and imagined rewards $\{\hat{r}_1, \hat{r}_2, \ldots, \hat{r}_H\}$ from an initial state s_0. Here, a neural network is used to approximate the critic with parameters ψ. The optimization goal of the critic model is to minimize the error between the estimated and the target Q through the Bellman equation. We give the objective function of the critic model from time step t:

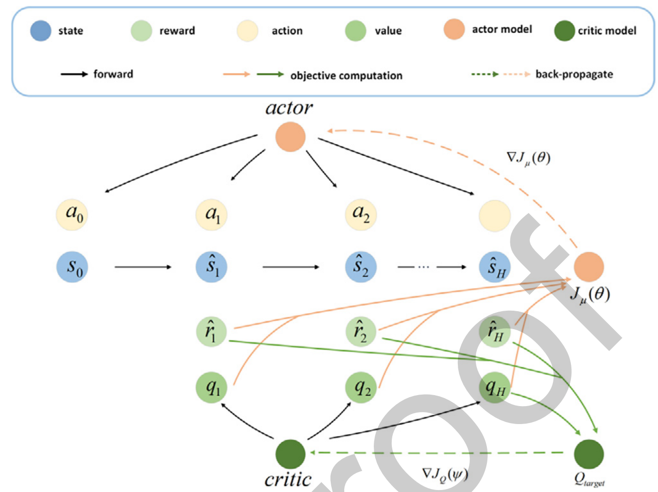


Fig. 3. Computation graph of Model-based deterministic policy gradient and clipped value expansion with horizons of H in the model.

$$\begin{aligned} J_Q(\psi)_t &= E\left[\left(Q(\hat{s}_t, a_t) - \left(\sum_{h=t}^{H-1} \gamma^h \hat{r}_{h+1} + \gamma^H Q(\hat{s}_H, a_H)\right)\right)^2\right] \\ &= E\left[\left(Q(\hat{s}_t, a_t) - \left(\sum_{h=t}^{H-1} \gamma^h f_r(\hat{s}_h, a_h) + \gamma^H Q(\hat{s}_H, a_H)\right)\right)^2\right] \end{aligned} \tag{16}$$

The target state-action value approximates the ground-truth return by accumulating the simulation rewards of multi-steps $\sum_{h=t}^{H-1} \gamma^h \hat{r}_{h+1}$. The $Q(\hat{s}_H, a_H)$ contains the expected return over the horizon.

In the actor-critic framework, maximizing the expectation of a critic network containing noise induces a consistent overestimation of the Q estimates. The overestimation bias causes a negative state to be overestimated, and it is easily propagated and accumulated through multiple updates, which leads to a suboptimal policy. Here, we construct a clipped double Q-network, which has two Q networks with the same structure to estimate the Q values independently. Then, we choose the minimum of them as the final estimation result to reduce the overestimation error:

$$Q = \min_{i=1,2} Q_i(s_t, a_t) \tag{17}$$

Although taking the minimum may induce an underestimation bias, it is far preferable to overestimation bias [48]. Our approach expands the clipped estimates in the learned model to further reduce the variance of the estimator error, which results in a more stable learning target for the update.

Algorithm 1. Model-based Deterministic Policy Gradient

1: Initialize environment buffer D_{env} and model buffer D_{model}. Initialize networks of dynamics f_ϕ, critic Q_ψ, actor μ_θ, and corresponding learning rates λ_f, λ_Q, and λ_μ.
2: Train expert via TD3 and generate demonstrations.
3: Update policy μ_θ and critic Q_ψ via demonstrations.
4: **for** N epochs do
5: Sample trajectories from the environment via stochastic policy π. Add them to D_{env}.
6: **for** M epochs **do**
7: Sample data from D_{env} and update dynamics model $\phi \leftarrow \phi - \lambda_f \nabla_\phi J_f(\phi)$.
8: **end**
9: **for** E epochs do

(continued on next page)

a (*continued*)

Algorithm 1. Model-based Deterministic Policy Gradient

```
        for H steps rollout in the model do
            Perform H steps rollout in model using policy μ_θ.
            Add trajectories data to D_model.
            Update value function ψ ← ψ − λ_Q ∇_ψ J_Q(ψ).
            Update policy θ ← θ + λ_μ ∇_θ J_μ(θ).
        end
    end
end
Return an optimal policy μ_θ.
```

4.3.2. Actor model with deterministic policy gradient

We use parameterized neural networks to represent a deterministic policy as $a_t = \mu(s_t|\theta)$, where θ is the parameter of the policy network. Consider an imagined trajectory with a finite horizon H under policy μ. The objective function of the actor model aims to maximize the total immediate rewards and the value estimations for the trajectory:

$$\begin{aligned} J_\mu(\theta) &= E\left[\beta \sum_{t=0}^{H-1} \gamma^t \hat{r}_{t+1} + \sum_{t=1}^{H} \gamma^t Q(\hat{s}_t, a_t)|_{\hat{s}_t = f_{s(\hat{s}_{t-1}, a_{t-1})}, a_t = \mu(\hat{s}_t)}\right] \\ &= E\left[\beta \sum_{t=0}^{H-1} \gamma^t f_r(\hat{s}_t, a_t) + \sum_{t=1}^{H} \gamma^t Q(\hat{s}_t, a_t)|_{\hat{s}_t = f_{s(\hat{s}_{t-1}, a_{t-1})}, a_t = \mu(\hat{s}_t)}\right] \end{aligned} \tag{18}$$

Where the parameter β is adaptive and indicates the weight of the immediate rewards $\sum_{t=0}^{H-1} \gamma^t \hat{r}_{t+1}$ in the total return. The agent can avoid short-sightedness and neglecting the future return by updating the parameter β. The term of the value estimations $\sum_{t=1}^{H} \gamma^t Q(\hat{s}_t, a_t)$ can prevent the exploding and vanishing gradient effect caused by backpropagation over time.

To optimize the objective function, we use the analytic gradient of the neural network. Utilizing the recurrence relation of the dynamics model, we can calculate the gradient of multiple steps in the trajectory. Here the model, policy, and value function are all deterministic and differentiable. Based on the chain rule and the deterministic policy gradient theorem [23], we give the method for calculating the gradient of the objective function with respect to the policy parameters in the model-based case:

$$\begin{aligned} \nabla_\theta J_\mu(\theta) = E[\beta \sum_{t=0}^{H-1} \gamma^t \left(\nabla_s f_r(\hat{s}_t, a_t) \nabla_\theta \hat{s}_t + \nabla_a f_r(\hat{s}_t, a_t) \nabla_\theta \mu_\theta(\hat{s}_t)\right) \\ + \sum_{t=1}^{H} \gamma^t \left(\nabla_a Q(\hat{s}_t, a_t) \nabla_\theta \mu_\theta(\hat{s}_t)\right)|_{\hat{s}_t = f_{s(\hat{s}_{t-1}, a_{t-1})}, a_t = \mu_\theta(\hat{s}_t)}] \end{aligned} \tag{19}$$

According to the Markov property and the learned state transfer model $\hat{s}_t \sim f_s(\hat{s}_{t-1}, a_{t-1})$, the gradient of the state with respect to the policy parameters can be expressed as the recursive formula:

$$\begin{aligned} \nabla_\theta \hat{s}_t &= \nabla_\theta f_s(\hat{s}_{t-1}, a_{t-1}) \\ &= \nabla_s f_s(\hat{s}_{t-1}, a_{t-1}) \nabla_\theta \hat{s}_{t-1} + \nabla_a f_s(\hat{s}_{t-1}, a_{t-1}) \nabla_\theta \mu_\theta(\hat{s}_{t-1}) \end{aligned} \tag{20}$$

Multi-step imagination is expanded in the model, which means that the agent can reach a larger state space, and richer gradient information can be exploited to optimize the policy compared to MFRL.

In general, a deterministic policy cannot guarantee sufficient exploration in the environment, unless there is adequate noise in the environment. We now consider an off-policy method that uses a stochastic actor $\pi(s|s_\mu, s_\sigma)$ as a behavioral policy to augment the exploration in the environment. The stochastic policy is a Gaussian distribution, the mean s_μ is the output of the deterministic policy $\mu(s_t|\theta)$, and the variance s_σ is a parameter that can be learned. The variance decreases with the increase in the expected reward of the deterministic policy. As the deterministic policy gradually converges, the stochasticity of exploration can be reduced. Two independent experience reply buffers are constructed to store environmental exploration samples and imagination data respectively. During the training process, the ratio of sampling from the model and environment can be tuned to control the sampling efficiency.

To prevent instability in the training process, we use target networks for the actor and critic, and the update frequency of the target network is lower than that of the actor and critic network.

5. Experiments

In this section, we describe the details of the experimental implementation. Our experiment aims to explore the following questions: (1) How well does our algorithm perform compared to the state-of-the-art model-based and model-free algorithms? (2) Whether our algorithm has an advantage in terms of time cost compared to other model-based methods? (3) What are the factors that affect the overall performance of the algorithm?

We compare our algorithm against two model-free and two model-based baselines. For MFRL baselines, we compare ours to soft actor-critic (SAC) [7], which has proven excellent convergence performance by maximizing entropy, and Twin Delayed Deep Deterministic policy gradient (TD3) [48], which considers the interplay between function approximation error in both value and policy updates. For MBRL baselines, we choose probabilistic ensembles with trajectory sampling (PETS) [34], which demonstrates excellent performance in approximating dynamics, and model-based policy optimization (MBPO) [13], which uses model-free SAC to accelerate the convergence of the policy.

We evaluate MBDPG and the baselines on a set of OpenAI Gym [51] and MuJoCo [52] continuous control tasks, illustrated in Fig. 1. These tasks have a series of challenges, including sparse rewards, complex dynamics, high-dimensional action space and state space. The actions are continuous and range from 2 to 8 dimensions. In BipedalWalker-v3 and LunarLanderContinuous-v2, there are abrupt changes in rewards that present a challenge to policy optimization.

5.1. Experiment implementation

All experiments are performed on a single Nvidia RTX TITAN XP GPU. To ensure the comparability of the experimental results, we modify all the algorithms to adapt to the experimental tasks and test them under the same experimental environment and conditions. The same hyperparameters are used across all tasks. In all methods, the random seed is set as 12345, and the initial parameters of all networks are generated from a Gaussian distribution with mean 0 and variance 1. To ensure a uniform evaluation standard, the maximum time steps for a test episode are set as 1,000. To prevent exploration from falling into an endless loop, an episode is terminated when the time steps exceed 1,000. If the agent reaches the termination state, we terminate the current episode in advance and start a new exploration episode.

In our method, the memory network is composed of LSTM Cells, which output hidden states and cell states. Three dense layers of size 256 are used to encode the hidden states to generate an ensemble Gaussian distribution and scaled rewards. The policy network, the critic network, and all other functions are implemented by three fully connected layers with hidden size of 256. In particular, the critic network adopts a double-Q network with the same structure. We use ELU activation to avoid the vanishing gradient problem in long-horizon prediction and planning.

A total of 4,000 time steps of random data are collected to initialize the parameters of the dynamics model. We sample batches of 128 containing sequences of length 200 to train the dynamics model, critic model, and actor model with learning rate 2×10^{-4}. The discount factor γ is set as 0.99. The entire training process is the alternate update of the critic model and the policy. The critic model is trained with a maximum step of 1000 per epoch in the model to achieve convergence, and then the actor model is updated.

5.2. Comparative experiments

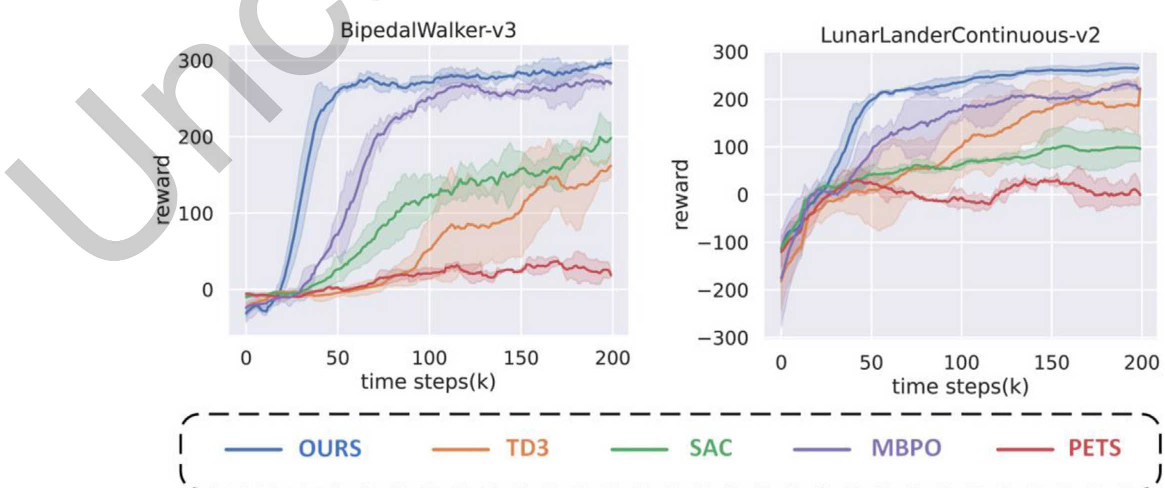
We compare our method with the SOTA MBRL and MFRL methods in terms of sampling efficiency, time efficiency, and progressive performance.

Performance in Box2D. We evaluate MBDPG and the baselines on the continuous control tasks in Box2D. Fig. 4 shows the learning curves for all methods over 200 k time steps. The results show that our method converges faster than baseline methods at the beginning of training. In addition, the results demonstrate that our method requires fewer samples than model-free methods. The learned dynamics model can simulate trajectories with a long horizon, so the sampling efficiency of our algorithm is higher than that of MBRL with one-step prediction. Moreover, our MBDPG method can converge in fewer time steps than the model-free method, and finally achieve comparable performance. For example, MBDPG approaches the best performance at 50 k time steps in the BipedalWalker-v3 task, but TD3 is still in the process of convergence at 200 k steps. The long horizon planning in our method drives the agent to stress the future rewards to approach the global optimal policy, while more exploration is needed in TD3 to obtain the equivalent learning signals.

Performance in MuJoCo. To verify the performance of MBDPG in high-dimensional tasks, we conducted comparative experiments in several MuJoCo environments. The results in Fig. 5 prove the good performance of MBDPG in high-dimensional tasks. Compared to MFRL, the learned dynamics model turns the RL problem into a planning problem so that the error of value estimation can be reduced and the gradient information can be backpropagated along the trajectory by expansion in the model. Compared to other MBRL algorithms, our method shows faster convergence of the policy, even several times faster on some tasks, proving the potential of multi-step deterministic policy gradient for fast optimization of policy. PETS requires multiple Monte Carlo experiments in the model to optimize the policy, which leads to its lower convergence performance. MBPO only uses the model to generate imaginary samples and does not fully utilize the gradient information of the model. In contrast, our approach exploits the model to both improve the accuracy of the critic model and enrich the learning signal of the policy. Compared to stochastic policy methods, the gradient estimation of deterministic policy in MBDPG avoids integration in action space, which ensures higher sampling efficiency than stochastic versions. For example, our policy performance is several times better than MBPO and SAC with the same number of training samples in the Hopper-v2 environment. Furthermore, MBDPG can accomplish tasks with high-dimensional actions, such as Ant-v2, which cannot be achieved for some model-based methods such as PETS, showing the importance of purposeful exploration.

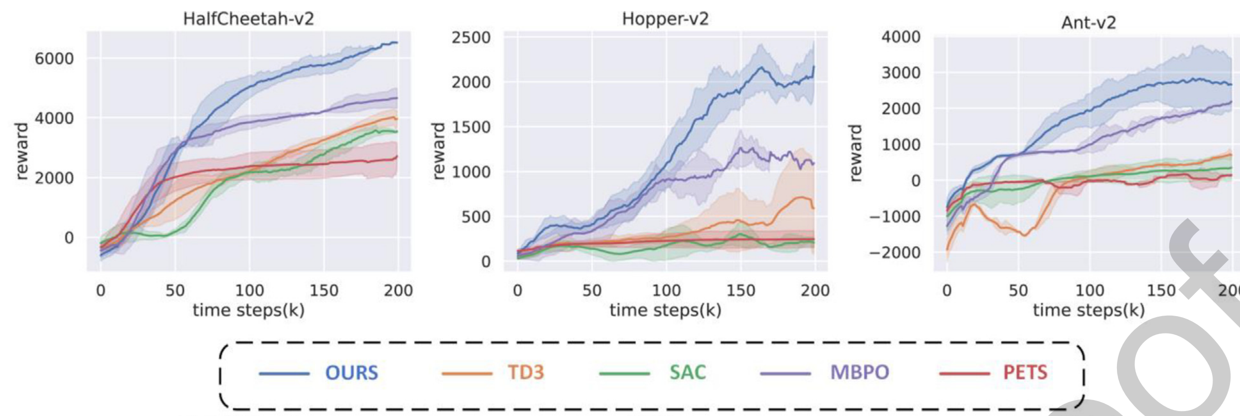
Time efficiency. We estimate the wall-clock time required for various methods to complete 200 k training time steps in different tasks, and the results are shown in Table 1. The time efficiency of our method exceeds that of some model-based methods, such as PETS, and is slightly lower than that of MFRL because of model learning. The time efficiency of some MB methods, such as PETS, is sensitive to the state-space and action-space dimensions of the task. PETS is inefficient in high-dimensional tasks, such as HalfCheetah-v2, because it requires a large number of complete trials. MBDPG is effective on complex tasks due to the use of the deterministic model and deterministic policy. The length of the horizon has a greater impact on the time efficiency of our algorithm. When $H = 10$, the time efficiency can be compared to SAC and TD3.

Aggregation methods of the dynamics model. We evaluate the prediction performance of the learned dynamics model with different aggregation methods, and Fig. 6 shows the results. In the bootstrap aggregating (bagging) method, we train each Gaussian network in the ensemble separately, and use the average of multiple independent Gaussian networks as the prediction result, which reduces the prediction variance of the learned dynamics model. In the XGBoost method, we choose a single Gaussian network from the ensemble as a regressor and keep using other Gaussian networks in the ensemble to fit the prediction residuals of the previous Gaussian network. The results show that this sequential aggregation method reduces the bias but not the variance. In contrast, the Mixture Gaussian Network in our approach demonstrates



(a) Walker (b) LunarLander

Fig. 4. Training curves of our algorithm and baselines in Box2D environment within 200 k total training time steps and 1000 steps in a test episode. The solid line indicates the mean of five experiments and the shaded regions indicate the standard deviation.



(a) HalfCheetah (b) Hopper (c) Ant

Fig. 5. Training curves of our algorithm and baselines in MuJoCo environment within 200 k total training time steps and 1000 steps in a test episode. The solid line indicates the mean of five experiments and the shaded regions indicate the standard deviation.

Table 1
The total wall-clock time in hours consumed for each algorithm to complete 200 k steps training and test.

Methods	BipedalWalker-v3	LunarLanderContinuous-v2	HalfCheetah-v2	Hopper-v2	Ant-v2
TD3	4.5 h	4.9 h	3.9 h	3.1 h	3.7 h
SAC	5.9 h	6.1 h	4.9 h	4.2 h	4.9 h
OURS	6.5 h	6.2 h	7.6 h	6.7 h	10.4 h
PETS	14.2 h	13.8 h	20.2 h	10.2 h	25.2 h
MBPO	10.4 h	9.8 h	17.2 h	10.1 h	21.6 h

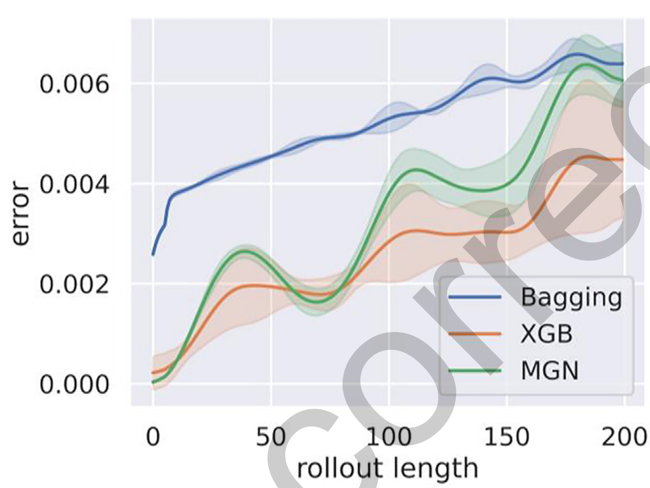


Fig. 6. Aggregation methods of the dynamics model.

a trade-off between bias and variance because of the adaptive weights parameterized by the neural networks.

5.3. Design evaluation

We next make comparison and ablation experiments on our method to investigate the factors affecting the performance of the algorithm.

5.3.1. Hyperparameters

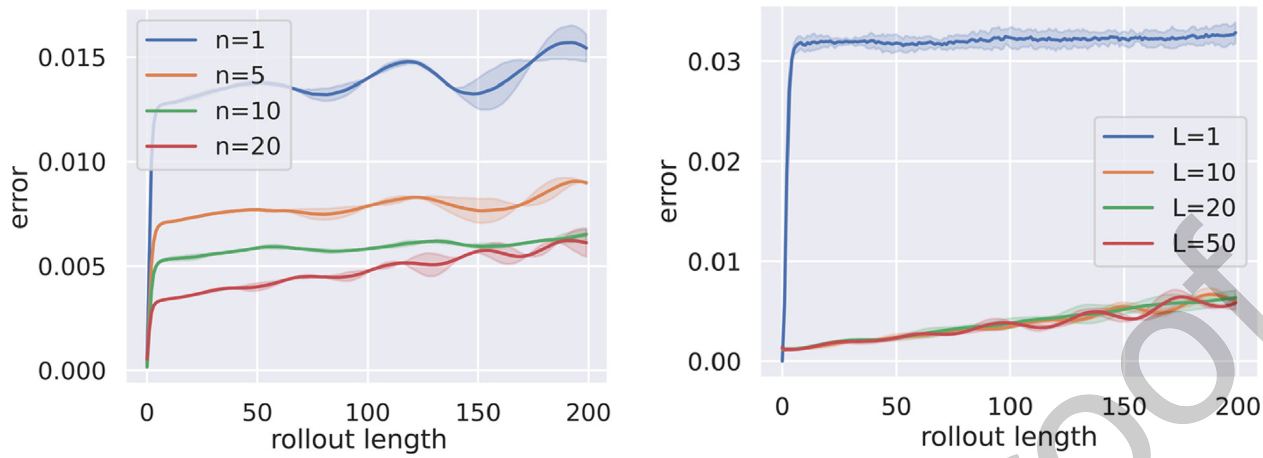
Experiments on the hyperparameters of the dynamics model are implemented as follows.

Size of the ensemble. To investigate its ability to approximate dynamics, we compare the prediction errors of multiple Mixture Gaussian Networks with different numbers n of Gaussian distributions. The networks are trained in the same environment with sequence samples of length 50, and Fig. 7(a) shows their prediction errors within 200 steps. The results show that increasing the size of the ensemble can effectively improve the prediction performance of the dynamics model, because more Gaussian distributions capture more stochasticity in the environment. When $n > 5$, the improvement in prediction accuracy from increasing the ensemble size is not significant. In contrast, more Gaussian models will bring more computational pressure. In our method, the number of Gaussian distributions is set to 5.

Model training length. The dynamics model is trained with different lengths L of rollouts, and the accuracy of the predictions is compared. Fig. 7(b) shows the error between the prediction and ground truth dynamics by implementing 200 steps in the model. We found that increasing L improves the accuracy of the predictions, as more training targets can be obtained, and the hidden states of LSTM can learn more historical information from longer trajectories. During one-step training, the prediction error for longer steps increases sharply. When $L > 10$, the error of each step is controlled in a small range, effectively suppressing the impact of accumulated error. However, an excessive L does not help much to improve the prediction accuracy, and it brings a computational burden. We set $L = 10$ in the experiments.

Experiments on the hyperparameters of policy optimization are implemented as follows.

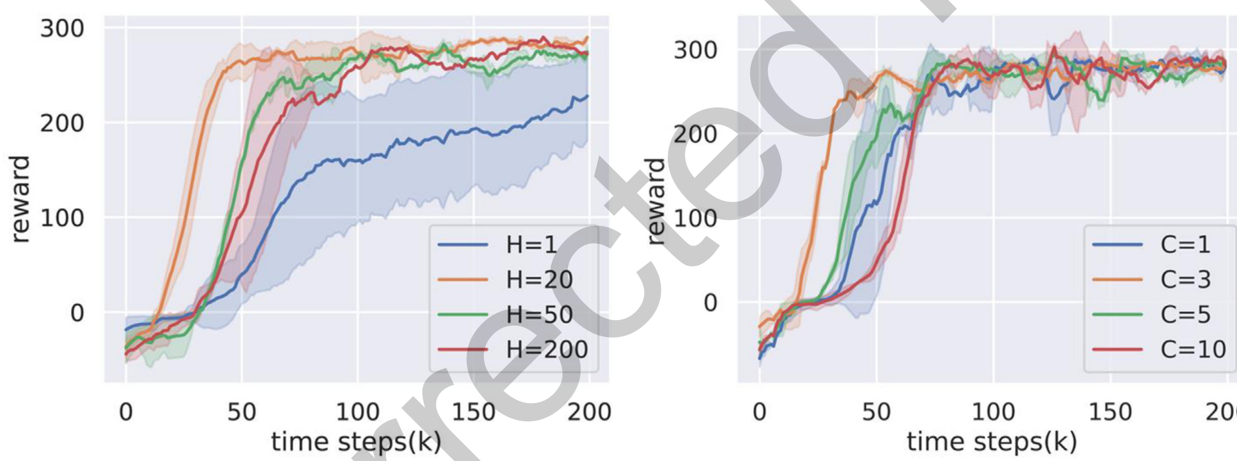
Horizons. One of the important parameters in MBDPG is the planning horizon H. Fig. 8(a) shows the performance of MBDPG with different imagination horizons in the BipedalWalker-v3 environment. Increasing H speeds up the convergence while $H < 20$, as it allows more sufficient exploration in the state space, which gives more informative policy gradient. Meanwhile, more precise value estimations can be obtained with a longer planning horizon. However, when $H > 50$, the error of the learned dynamics model will offset or even negate the benefit of more exploration. Moreover, the gradient backpropagation through an excessive horizon would consume unnecessary computation. We find that our algorithm still works when $H = 200$, benefiting from the value function,



(a) Size of ensemble

(b) Model training lenght

Fig. 7. Learing curves with different hyperparameters of dynamics model. The solid line indicates the mean of five experiments and the shaded regions indicate the standard deviation.



(a) Horizons

(b) Update frequency

Fig. 8. Learing curves with different hyperparameters of policy optimization. The solid line indicates the mean of five experiments and the shaded regions indicate the standard deviation.

although the performance is worse than for shorter values. The value function contains long-term returns outside the horizon. The upcoming predicted rewards and the value function depend on the predicted state, so an accurate model over a long horizon is critical. On the other hand, increasing the horizon improves the sampling efficiency, as more simulation samples can be obtained from the model. In our experiments, $H = 20$ is an appropriate value.

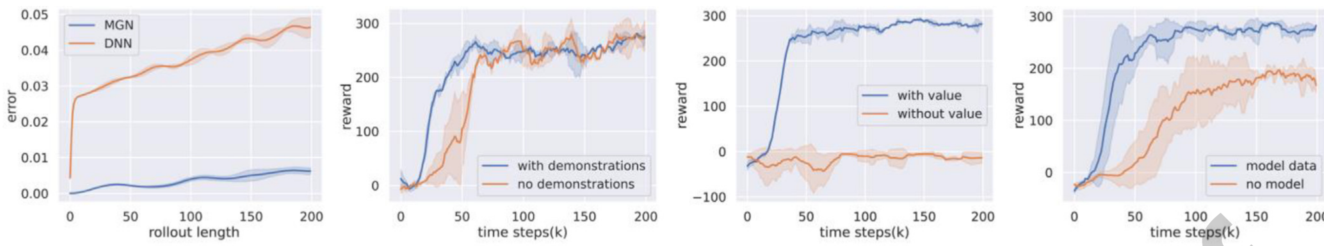
Update frequency of the target networks. The update frequency C is a crucial factor. In our experiments, the critic networks are updated in each iteration, and the target networks are only updated after C iterations. The result in Fig. 8(b) shows that the convergence of the policy is not robust when $C = 1$, because updating the target network and the critic model simultaneously leads to unstable learning objectives. When $C \geqslant 5$, increasing C hurts policy learning, as updating the target network too late makes it inaccurate in the current state space. In our experiments, we set $C = 3$ to obtain a stable and accurate target value.

5.3.2. Ablation experiments

The following will discuss a series of ablation experiments in the BipedalWalker-v3 environment to verify the role of each component in our algorithm.

Mixture Gaussian Network. To prove its effectiveness, we compare our Mixture Gaussian Network to a deep neural network with three fully connected layers. The hidden size of each layer is 256. The results in Fig. 9(a) show that removing the Mixture Gaussian Network module significantly reduces the multi-step prediction performance of learned dynamics. The prediction errors of the deep neural network increase dramatically with the length of rollout in the model, even after training the network for multi-step prediction. Fully connected layers have difficulty capturing the aleatoric uncertainty and the epistemic uncertainty in the system. In contrast, the prediction errors of the Mixture Gaussian Network can still be controlled in a reasonable range when $H = 200$, which guarantees the robustness of planning in the learned model.

Successful experience. We test the performance of the algorithm with and without demonstrations and show the result in



(a) Mixture Gaussian Network (b) Successful experience (c) Terminal value (d) Model

Fig. 9. Training curves of ablation experiments. The solid line indicates the mean of five experiments and the shaded regions indicate the standard deviation.

Fig. 9(b). The algorithm performance is severely declined in the early stage of training without the guidance of expert data, proving the importance of successful experience on policy learning.

Terminal value. The standard MBDPG and the version without the terminal value expansion are compared under the same settings. Without using the terminal Q value, the performance of the algorithm is severely degraded, and the agent ignores the temporal backup from the future value function and only takes advantage of the reward sequence. As a result, the agent struggles with the local optima dilemma. Fig. 9(c) shows that terminal value expansion indeed contributes to long-horizon planning and tasks with delayed rewards.

Model. Ablating the model, which yields the model-free version of the current algorithm, severely slows the convergence in Fig. 9 (d). Using the same number of environment samples, the learned dynamics model increases the richness of the training samples with imaginary data compared to the model-free version, even if there are errors in the model. In addition, the data generated by the model prevents overfitting. The time efficiency does improve when using only the environment data, but the algorithm would not be able to match the sample efficiency requirements.

6. Conclusions

In this work, we present the model-based deterministic policy gradient, MBDPG, a model-based policy optimization method that exploits the gradient information of multi-step simulation in the learned dynamics to optimize a deterministic policy. Our algorithm demonstrates significantly higher sampling efficiency than SOTA MFRL methods. To ensure the accuracy of long-term prediction, a Mixture Gaussian Network equipped with a recurrent module is built to approximate the dynamics model. Second, this work incorporates guidance from successful experience. Third, we roll out multi-step simulation in the learned dynamics model and back-propagate the trajectory gradient in an end-to-end manner through the differentiable dynamics model. A clipped value expansion is used to learn an accurate and stable critic model. Experimental results demonstrate that our MBDPG method achieves faster convergence than SOTA MBRL methods, especially for tasks with large action spaces. Future research will be directed to extend the state representation to high-dimensional visual information. It would be enticing to apply the algorithm to a real environment, such as robotics, unmanned vehicles, video games, and multi-agent games. In addition, we aim to develop our algorithm to improve the performance of Mixture Gaussian Network by integrating the latest aggregation methods, such as the attention mechanism.

CRediT authorship contribution statement

Shiqing Gao: Conceptualization, Methodology, Software, Investigation, Writing – original draft. **Haibo Shi:** Methodology, Investigation, Writing – review & editing. **Fang Wang:** Data curation, Validation. **Zijian Wang:** Validation, Writing – review & editing. **Siyu Zhang:** Visualization, Writing – review & editing. **Yunxia Li:** Project administration, Funding acquisition, Writing – review & editing. **Yaoru Sun:** Conceptualization, Supervision, Funding acquisition, Writing – review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was supported by the grants from the National Key R&D Program of China (2019YFC1906201), and the National Natural Science Foundation of China (91748122).

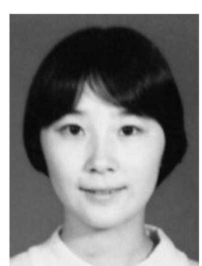
References

[1] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, Mastering the game of Go with deep neural networks and tree search, nature, 529 (2016) 484-489.
[2] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Mastering the game of go without human knowledge, Nature 550 (2017) 354–359.
[3] V. Mnih, A.P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, International conference on machine learning, (PMLR2016), pp. 1928-1937.
[4] H. Van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning, Proceedings of the AAAI conference on artificial intelligence2016).
[5] V. Varga, A. Lőrincz, Reducing human efforts in video segmentation annotation with reinforcement learning, Neurocomputing 405 (2020) 247–258.
[6] R. Ceren, K. He, P. Doshi, B. Banerjee, PALO bounds for reinforcement learning in partially observable stochastic games, Neurocomputing 420 (2021) 36–56.
[7] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, International conference on machine learning, (PMLR2018), pp. 1861-1870.
[8] N. Zeng, H. Li, Z. Wang, W. Liu, S. Liu, F.E. Alsaadi, X. Liu, Deep-reinforcement-learning-based images segmentation for quantitative analysis of gold immunochromatographic strip, Neurocomputing 425 (2021) 173–180.
[9] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation, CoRL2018).
[10] C. Finn, I. Goodfellow, S. Levine, Unsupervised learning for physical interaction through video prediction, Adv. Neural Inf. Process. Syst. 29 (2016) 64–72.
[11] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347, (2017).
[12] M. Deisenroth, C.E. Rasmussen, PILCO: A model-based and data-efficient approach to policy search, Proceedings of the 28th International Conference on machine learning (ICML-11), (Citeseer2011), pp. 465-472.
[13] M. Janner, J. Fu, M. Zhang, S. Levine, When to trust your model: model-based policy optimization, Adv. Neural Inf. Process. Syst. 32 (2019) 12519–12530.
[14] R. Lieck, M. Toussaint, Temporally extended features in model-based reinforcement learning with partial observability, Neurocomputing 192 (2016) 49–60.
[15] S. Racanière, T. Weber, D.P. Reichert, L. Buesing, A. Guez, D. Rezende, A.P. Badia, O. Vinyals, N. Heess, Y. Li, Imagination-augmented agents for deep reinforcement learning, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 5694–5705.

[16] J. Buckman, D. Hafner, G. Tucker, E. Brevdo, H. Lee, Sample-Efficient Reinforcement Learning with Stochastic Ensemble Value Expansion, NeurIPS2018).
[17] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, P. Abbeel, Model-Ensemble Trust-Region Policy Optimization, International Conference on Learning Representations2018).
[18] A.V. Rao, A survey of numerical methods for optimal control, Adv. Astronaut. Sci. 135 (2009) 497–528.
[19] A.G. Richards, Robust constrained model predictive control, Massachusetts Institute of Technology (2005).
[20] Z.I. Botev, D.P. Kroese, R.Y. Rubinstein, P. L'Ecuyer, The cross-entropy method for optimization, Handbook of statistics, (Elsevier, 2013), pp. 35-59.
[21] S. Kamthe, M. Deisenroth, Data-efficient reinforcement learning with probabilistic model predictive control, International conference on artificial intelligence and statistics, (PMLR2018), pp. 1701-1710.
[22] I. Clavera, Y. Fu, P. Abbeel, Model-Augmented Actor-Critic: Backpropagating through Paths, International Conference on Learning Representations2019).
[23] N. Heess, G. Wayne, D. Silver, T. Lillicrap, T. Erez, Y. Tassa, Learning Continuous Control Policies by Stochastic Value Gradients, Adv. Neural Inf. Process. Syst. 28 (2015) 2944–2952.
[24] T.M. Moerland J. Broekens C.M. Jonker Model-based reinforcement learning: A survey, arXiv preprint arXiv:2006.16712 2020
[25] A. Der Kiureghian, O. Ditlevsen, Aleatory or epistemic? Does it matter?, Struct Saf. 31 (2009) 105–112.
[26] M.C. Machado, M.G. Bellemare, E. Talvitie, J. Veness, M. Hausknecht, M. Bowling, Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents, Journal of Artificial Intelligence Research 61 (2018) 523–562.
[27] E. Talvitie, Self-correcting models for model-based reinforcement learning, Thirty-First AAAI Conference on Artificial Intelligence2017).
[28] F. Torabi, G. Warnell, P. Stone, Behavioral cloning from observation, in: Proceedings of the 27th International Joint Conference on Artificial Intelligence, 2018, pp. 4950–4957.
[29] A.Y. Ng, S.J. Russell, Algorithms for inverse reinforcement learning, Icml (2000) 2.
[30] T. Degris, M. White, R.S. Sutton, Off-policy actor-critic, in: Proceedings of the 29th International Coference on International Conference on Machine Learning, 2012, pp. 179–186.
[31] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[32] M.P. Deisenroth, D. Fox, C.E. Rasmussen, Gaussian processes for data-efficient learning in robotics and control, IEEE Trans. Pattern Anal. Mach. Intell. 37 (2) (2015) 408–423.
[33] J. Oh, X. Guo, H. Lee, R.L. Lewis, S. Singh, Action-Conditional Video Prediction using Deep Networks in Atari Games, Adv. Neural Inf. Process. Syst. 28 (2015) 2863–2871.
[34] K. Chua, R. Calandra, R. McAllister, S. Levine, Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models, NeurIPS2018).
[35] D. Ha, J. Schmidhuber, Recurrent world models facilitate policy evolution, in: Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018, pp. 2455–2467.
[36] J. Beck, K. Ciosek, S. Devlin, S. Tschiatschek, C. Zhang, K. Hofmann, Amrl: aggregated memory for reinforcement learning, International Conference on Learning Representations2019).
[37] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, J. Davidson, Learning latent dynamics for planning from pixels, International Conference on Machine Learning, (PMLR2019), pp. 2555-2565.
[38] D. Ha, J. Schmidhuber, World models, arXiv preprint arXiv:1803.10122, (2018).
[39] N. Zeng, H. Zhang, Y. Chen, B. Chen, Y. Liu, Path planning for intelligent robot based on switching local evolutionary PSO algorithm, Assembly Automation 36 (2) (2016) 120–126.
[40] W. Liu, Z. Wang, X. Liu, N. Zeng, D. Bell, A novel particle swarm optimization approach for patient clustering from emergency departments, IEEE Trans. Evol. Comput. 23 (2018) 632–644.
[41] N. Zeng, Z. Wang, W. Liu, H. Zhang, K. Hone, X. Liu, A dynamic neighborhood-based switching particle swarm optimization algorithm, IEEE Transactions on, Cybernetics (2020).
[42] A. Nagabandi, G. Kahn, R.S. Fearing, S. Levine, Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning, 2018 IEEE International Conference on Robotics and Automation (ICRA), (IEEE2018), pp. 7559-7566.
[43] Y. Luo, H. Xu, Y. Li, Y. Tian, T. Darrell, T. Ma, Algorithmic Framework for Model-based Deep Reinforcement Learning with Theoretical Guarantees, International Conference on Learning Representations2018).
[44] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, Trust region policy optimization, International conference on machine learning, (PMLR2015), pp. 1889-1897.
[45] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, Soft actor-critic algorithms and applications, arXiv preprint arXiv:1812.05905, (2018).
[46] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, Deterministic policy gradient algorithms, International conference on machine learning, (PMLR2014), pp. 387-395.
[47] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, arXiv preprint arXiv:1509.02971, (2015).
[48] S. Fujimoto, H. Hoof, D. Meger, Addressing function approximation error in actor-critic methods, International Conference on Machine Learning, (PMLR2018), pp. 1587-1596.
[49] D. Hafner, T. Lillicrap, J. Ba, M. Norouzi, Dream to Control: Learning Behaviors by Latent Imagination, International Conference on Learning Representations2019).
[50] M.L. Puterman, Markov decision processes: discrete stochastic dynamic programming, John Wiley & Sons, 2014.
[51] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, Openai gym, arXiv preprint arXiv:1606.01540, (2016).
[52] E. Todorov, T. Erez, Y. Tassa, Mujoco: A physics engine for model-based control, 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, (IEEE2012), pp. 5026-5033.

Shiqing Gao received the B.S. degree from the School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China, in 2019. He is currently pursuing the M.S. degree in Computer Science and Technology at Tongji University, Shanghai, China. His research interests include deep reinforcement learning and computer vision.

Haibo Shi received the Ph.D. degree in pattern recognition and intelligent system from Tongji University, Shanghai, China. He is currently an Assistant Professor with the Department of Applied Statistics, Shanghai University of Finance and Economics, Shanghai, China. His research interests cover cognitive computation and deep reinforcement learning.



Fang Wang is a senior lecturer in the Department of Computer Science at Brunel University, UK and her research interests cover software agents, cognitive neuroscience and distributed computing. She has published many papers, filed a number of patents and received several technical awards.

Zijian Wang received the Ph.D. degree from Tongji University, P.R. China. Now, he works as a lecturer in School of Computer Science and Technology, Donghua University. His research interest includes artificial intelligence and medical imaging analysis.

Siyu Zhang received the M.S. degree from the School of Computer and Information Technology, Liaoning Normal University, China, in 2016. She is currently pursuing the Ph.D degree in Computer science and technology from Tongji University, Shanghai, China. Her research interests include signal processing and computer vision.

Yunxia Li received the M.D. degree in 2007 and now works as a Chief Physician in Department of Neurology of Tongji Hospital and professor at School of Medicine, Tongji University. Her current research interests include neural learning in neurocognitive disorders diagnose and neural injury mechanisms.

Yaoru Sun received the Ph.D. degree in artificial intelligence from the University of Edinburgh. He is currently a full Professor with the Department of Computer Science and Technology, Tongji University, China. His research interests include brian-like computation, machine intelligence and cognitive neuroscience.