*ECAI 2023*
*K. Gal et al. (Eds.)*


*doi:10.3233/FAIA230427*

# Model-Based Reinforcement Learning with Multi-Step Plan Value Estimation

**Haoxin Lin**[a,b;*], **Yihao Sun**[a;*], **Jiaji Zhang**[a] **and Yang Yu**[a,b,c;**]

[a]National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu, China
[b]Polixir Technologies, Nanjing, Jiangsu, China
[c]Peng Cheng Laboratory, Shenzhen, Guangdong, China

**Abstract.** A promising way to improve the sample efficiency of reinforcement learning is model-based methods, in which many explorations and evaluations can happen in the learned models to save real-world samples. However, when the learned model has a non-negligible model error, sequential steps in the model are hard to be accurately evaluated, limiting the model's utilization. This paper proposes to alleviate this issue by introducing multi-step plans into policy optimization for model-based RL. We employ the multi-step plan value estimation, which evaluates the expected discounted return after executing a sequence of action plans at a given state, and updates the policy by directly computing the multi-step policy gradient via plan value estimation. The new model-based reinforcement learning algorithm MPPVE (**M**odel-based Planning **P**olicy Learning with Multi-step **P**lan **V**alue **E**stimation) shows a better utilization of the learned model and achieves a better sample efficiency than state-of-the-art model-based RL approaches. The code is available at https://github.com/HxLyn3/MPPVE.

## 1 Introduction

Reinforcement Learning (RL) has attracted close attention in recent years. Despite its empirical success in simulated domains and games, insufficient sample efficiency is still one of the critical problems hindering the application of RL in reality [32]. One promising way to improve the sample efficiency is to train and utilize world models [32], which is known as model-based RL (MBRL). World model learning has recently received significant developments, including the elimination of the compounding error issue [31] and causal model learning studies [9, 34] for achieving high-fidelity models, and has also gained real-world applications [27, 26]. Nevertheless, this paper focuses on the dyna-style MBRL framework [29] that augments the replay buffer by the world model generated data for off-policy RL.

In dyna-style MBRL, the model is often learned by supervised learning to fit the observed transition data, which is simple to train but exhibits non-negligible model error [31]. Specifically, consider a generated $k$-step model rollout $s_t, a_t, \hat{s}_{t+1}, a_{t+1}, ..., \hat{s}_{t+k}$, where $\hat{s}$ stands for fake states. The deviation error of fake states increases with $k$ since the error accumulates gradually as the state transitions in imagination. If updated on the fake states with a large deviation, the policy will be misled by policy gradients given by biased state-action value estimation. The influence of model error on the directions of policy gradients is demonstrated in Figure 1.

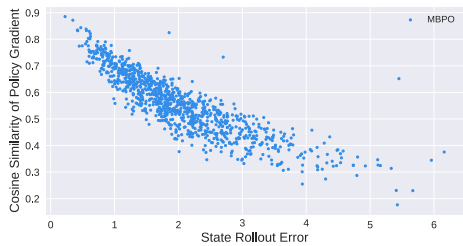



**Figure 1.** We make real branched rollouts in the real world and utilize the learned model to generate the fake, starting from some real states. For each fake rollout and its corresponding real rollout, we show their deviation along with the normalized cosine similarity between their multi-step policy gradients.

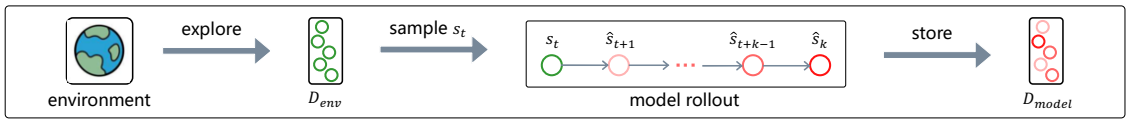
Therefore, the dyna-style MBRL often introduces techniques to reduce the impact of the model error. For example, MBPO [14] proposes the branched rollouts scheme with a gradually growing branch length to truncate imaginary model rollouts, avoiding the participation of unreliable fake samples in policy optimization. BMPO [17] further truncates model rollouts into a shorter branch length than MBPO after the same number of steps of environmental sampling since it can make imaginary rollouts from both forward and backward directions with the bidirectional dynamics model.

We argue that, while the above previous methods tried to reduce the impact of the accumulated model error by reducing the rollout steps, avoiding the explicit reliance on the rolled-out fake states can be helpful. This paper proposes employing $k$-step plan value estimation $Q(s_t, a_t, ..., a_{t+k-1})$ to evaluate sequential action plans. When using the $k$-step rollout to update the policy, we can compute the policy gradient only on the starting real state $s_t$, but not on any fake states. Therefore, the $k$-step policy gradients directly given by the plan value are less influenced by the model error. Based on this key idea, we formulate a new model-based algorithm called **M**odel-based Planning **P**olicy Learning with Multi-step **P**lan **V**alue **E**stimation (MPPVE). The difference between MPPVE and previous model-based methods in updating the actor is demonstrated in Figure 2.

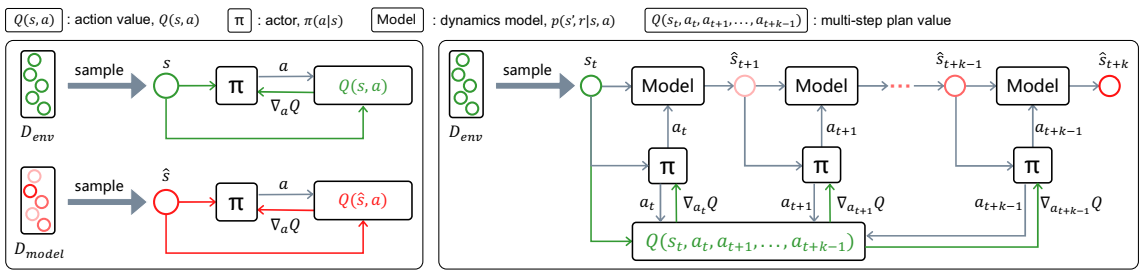
In general, our contributions are summarized as follows:

- We present a tabular planning policy iteration method alternating between planning policy evaluation and planning policy improvement, and theoretically prove the convergence of the optimal policy.

* Equal Contribution.
** Corresponding Author. Email: yuy@nju.edu.cn




(a) Data generation. Environmental data is stored in $\mathcal{D}_{\text{env}}$, while model data is stored in $\mathcal{D}_{\text{model}}$.




(b) Update actor with action-value.

(c) Update actor with multi-step plan value.

**Figure 2.** A comparison between MPPVE and previous model-based methods in updating actor. (a) Data generation, where green $s_t$ stands for the real environmental state, red $\hat{s}_{t+1}, \ldots, \hat{s}_{t+k}$ stand for fake states, and the darker the red becomes, the greater the deviation error is in expectation. (b) Illustration of how previous model-based methods update actor with action-value estimation. (c) Illustration of how MPPVE updates actor with $k$-step plan value estimation. The $k$-step policy gradients are given by the plan value estimation directly when the actor plans starting from the real environmental state.

- We propose a new model-based algorithm called MPPVE for general continuous settings based on the theoretical tabular planning policy iteration, which updates the policy by directly computing multi-step policy gradients via plan value estimation for short plans starting from real environmental states, mitigating the misleading impacts of the compounding error.
- We empirically verify that multi-step policy gradients computed by MPPVE are less influenced by model error and more accurate than those computed by previous model-based RL methods.
- We show that MPPVE outperforms recent state-of-the-art model-based algorithms in terms of sample efficiency while retaining competitive performance close to the convergence of model-free algorithms on MuJoCo [30] benchmarks.

## 2 Related Work

This work is related to dyna-style MBRL [29] and multi-step planning.

Dyna-style MBRL methods generate some fake transitions with a dynamics model for accelerating value approximation or policy learning. MVE [12] uses a dynamics model to simulate the short-term horizon and Q-Learning to estimate the long-term value, improving the quality of target values for training. STEVE [5] builds on MVE to better estimate Q value by stochastic ensemble model-based value expansion. SLBO [20] regards the dynamics model as a simulator and directly uses TRPO [25] to optimize the policy with whole trajectories sampled in it. Moreover, MBPO [14] builds on SAC [13], which is an off-policy RL algorithm, and updates the policy with a mixture of the data from the real environment and imaginary branched rollouts. Some recent dyna-style MBRL methods pay attention to reducing the influences of model error. For instance, M2AC [24] masks the high-uncertainty model-generated data with a masking mechanism. BMPO [17] utilizes both the forward and backward model to separate the compounding error in different directions, which can acquire model data with less error than only using the forward model. This work also adopts the dyna-style MBRL framework and focuses on mitigating the impact of the accumulated model error.

Multi-step planning methods usually utilize the learned dynamics model to make plans. Model Predictive Control (MPC) [6] obtains an optimal action sequence by sampling multiple sequences and applying the first action of the sequence to the environment. MB-MF [22] adopts the random-shooting method as an instantiation of MPC, which samples several action sequences randomly and uniformly in a learned neural model. PETS [10] uses CEM [4] instead, which samples actions from a distribution close to previous samples that yielded high rewards to improve the optimization efficiency.

The planning can be incorporated into the differentiable neural network to learn the planning policy end-to-end directly [23, 28, 15]. MAAC [11] proposes to estimate the policy gradients by back-propagating through the learned dynamics model using the path-wise derivative estimator during model-based planning. Furthermore, DDPPO [18] additionally learns a gradient model by minimizing the gradient error of the dynamics model to provide more accurate policy gradients through the model. This work also learns the planning policy end-to-end, but back-propagates the policy gradients through multi-step plan value instead, avoiding the reliance on the fake states.

The concept of multi-step sequential actions is introduced in other model-based approaches. [1, 2, 16, 7] propose a multi-step dynamics model to directly output the outcome of executing a sequence of actions. Unlike these, we involve the concept of multi-step sequential actions in the value function instead of the dynamics model.

Multi-step plan value used in this paper is also presented in GPM [33]. Our approach differs from GPM in the following two aspects: 1) GPM adopts the model-free paradigm while ours adopts the model-based paradigm; 2) GPM aims to enhance exploration and therefore present a plan generator to output multi-step actions based on the plan value, while we propose model-based planning policy improvement based on the plan value estimation to mitigate the influence of compounding error and increase the sample efficiency.

## 3 Preliminaries

A tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$ is considered to describe Markov Decision Process (MDP), where $\mathcal{S}$ and $\mathcal{A}$ are state and action spaces respectively, probability density function $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, \infty)$ represents the transition distribution over $s_{t+1} \in \mathcal{S}$ conditioned on $s_t \in \mathcal{S}$ and $a_t \in \mathcal{A}$, $r : \mathcal{S} \times \mathcal{A} \times \mathbb{R} \to [0, \infty)$ is the reward distribution conditioned on $s_t \in \mathcal{S}$ and $a_t \in \mathcal{A}$, and $\gamma$ is the discount factor. We will use $\rho^\pi : \mathcal{S} \to [0, \infty)$ to denote on-policy distribution over states induced

by dynamics function $p(s_{t+1}|s_t, a_t)$ and policy $\pi(a_t|s_t)$. The goal of RL is to find the optimal policy that maximizes the expectation of cumulative discounted reward: $\mathbb{E}_{\rho^\pi}\left[\sum_{t=0}^{\infty}\gamma^t r_t\right]$.

Recent MBRL methods aim to build a model of the dynamics function $p$ using supervised learning with data $\mathcal{D}_{\text{env}}$ collected via interaction in the real environment. Then fake data $\mathcal{D}_{\text{model}}$ generated via model rollouts will be used for an RL algorithm additionally with real data to improve the sample efficiency.

## 4 Method

In this section, we will first derive tabular planning policy iteration, and verify that the optimal policy can be attained with a convergence guarantee. Then we will present a practical neuron-based algorithm for general continuous environments based on this theory.

### 4.1 Derivation of Planning Policy Iteration

Planning policy iteration is a multi-step extension of policy iteration for optimizing planning policy that alternates between planning policy evaluation and planning policy improvement. With the consideration of theoretical analysis, our derivation will focus on the tabular setting.

#### 4.1.1 Planning Policy

With the dynamics function $p$, at any state, the agent can generate a plan consisting of a sequence of actions to perform in the next few steps in turn. We denote the $k$-step planning policy as $\boldsymbol{\pi}^k$, then given state $s_t$, the plan $\boldsymbol{\tau}_t^k = (a_t, a_{t+1}, \ldots, a_{t+k-1})$ can be predicted with $\boldsymbol{\tau}_t^k \sim \boldsymbol{\pi}^k(\cdot|s_t)$. Concretely, for $m \in \{0, 1, \ldots, k-1\}$, $a_{t+m} \sim \pi(\cdot|s_{t+m})$, $s_{t+m+1} \sim p(\cdot|s_{t+m}, a_{t+m})$, we have

$$\boldsymbol{\pi}^k(\boldsymbol{\tau}_t^k|s_t) = \pi(a_t|s_t)\sum_{\boldsymbol{s}_{t+1}^{k-1}}\left[\prod_{i=t+1}^{t+k-1} p(s_i|s_{i-1}, a_{i-1})\pi(a_i|s_i)\right], \tag{1}$$

where $\boldsymbol{s}_{t+1}^{k-1} = (s_{t+1}, \ldots, s_{t+k-1})$ is the state sequence at future $k-1$ steps. The planning policy just gives a temporally consecutive plan of fixed length $k$, not a full plan that plans actions until the termination can be reached.

#### 4.1.2 Planning Policy Evaluation

Given a stochastic policy $\pi \in \Pi$, the $k$-step plan value [33] function is defined as

$$\begin{aligned} &Q^\pi(s_t, \boldsymbol{\tau}_t^k) = Q^\pi(s_t, a_t, a_{t+1}, ..., a_{t+k-1}) \\ =&\mathbb{E}_{p,r,\pi}\left[\sum_{m=0}^{k-1}\gamma^m r_{t+m} + \gamma^k\sum_{m=0}^{\infty}\gamma^m r_{t+k+m}\right] \qquad (2) \\ =&\mathbb{E}_{p,r}\left[\sum_{m=0}^{k-1}\gamma^m r_{t+m} + \gamma^k\mathbb{E}_{\hat{\boldsymbol{\tau}}^k\sim\boldsymbol{\pi}^k}\left[Q^\pi(s_{t+k}, \hat{\boldsymbol{\tau}}^k)\right]\right]. \qquad (3) \end{aligned}$$

The former $k$ items $(r_t, r_{t+1}, ..., r_{t+k-1})$ are instant rewards respectively for $k$-step actions in the plan $\boldsymbol{\tau}_t^k$ taken at state $s_t$ step by step, while the following items are future rewards for starting making decisions according to $\pi$ from state $s_{t+k}$.

According to the recursive Bellman equation (3), the extended Bellman backup operator $\mathcal{T}^\pi$ can be written as

$$\mathcal{T}^\pi Q(s_t, \boldsymbol{\tau}_t^k) = \mathbb{E}_{p,r}\left[\sum_{m=0}^{k-1}\gamma^m r_{t+m}\right] + \gamma^k\mathbb{E}_p\left[V(s_{t+k})\right], \tag{4}$$

where

$$V(s_t) = \mathbb{E}_{\boldsymbol{\tau}_t^k\sim\boldsymbol{\pi}^k}\left[Q(s_t, \boldsymbol{\tau}_t^k)\right] \tag{5}$$

is the state value function. This update equation for $k$-step plan value function is different from multi-step Temporal Difference (TD) Learning [3], though there is naturally multi-step bootstrapping in plan value learning. Specifically, the plan value is updated without bias since the multi-step rewards $(r_t, r_{t+1}, ..., r_{t+k-1})$ correspond to the input plan, while the multi-step TD Learning for single-step action-value needs importance sampling or Tree-backup [3] to correct the bias from the difference between target policy and behavior policy.

Starting from any function $Q : \mathcal{S} \times \mathcal{A}^k \to \mathbb{R}$ and applying $\mathcal{T}^\pi$ repeatedly, we can obtain the plan value function of the policy $\pi$.

**Lemma 1** (Planning Policy Evaluation)**.** *Given any initial mapping $Q_0 : \mathcal{S} \times \mathcal{A}^k \to \mathbb{R}$ with $|\mathcal{A}| < \infty$, update $Q_i$ to $Q_{i+1}$ with $Q_{i+1} = \mathcal{T}^\pi Q_i$ for all $i \in N$, $\{Q_i\}$ will converge to the plan value of policy $\pi$ as $i \to \infty$.*

*Proof.* For any $i \in N$, after updating $Q_i$ to $Q_{i+1}$ with $\mathcal{T}^\pi$, we have

$$\begin{aligned} &Q_{i+1}(s_t, \boldsymbol{\tau}_t^k) = \mathcal{T}^\pi Q_i(s_t, \boldsymbol{\tau}_t^k) \\ =&\mathbb{E}_{p,r}\left[\sum_{m=0}^{k-1}\gamma^m r_{t+m} + \gamma^k\mathbb{E}_{\hat{\boldsymbol{\tau}}^k\sim\boldsymbol{\pi}^k}\left[Q_i(s_{t+k}, \hat{\boldsymbol{\tau}}^k)\right]\right], \end{aligned} \tag{6}$$

for any $s_t \in \mathcal{S}$ and $\boldsymbol{\tau}_t^k \in \mathcal{A}^k$. Then

$$\begin{aligned} &\|Q_{i+1} - Q^\pi\|_\infty = \max_{s_t\in\mathcal{S}, \boldsymbol{\tau}_t^k\in\mathcal{A}^k}\left|Q_{i+1}(s_t, \boldsymbol{\tau}_t^k) - Q^\pi(s_t, \boldsymbol{\tau}_t^k)\right| \\ \le&\max_{s_t\in\mathcal{S}, \boldsymbol{\tau}_t^k\in\mathcal{A}^k}\gamma^k\|Q_i - Q^\pi\|_\infty = \gamma^k\|Q_i - Q^\pi\|_\infty. \end{aligned} \tag{7}$$

So $Q_i = Q_\pi$ is a fixed point of this update rule, and the sequence $\{Q_i\}$ will converge to the plan value of policy $\pi$ as $i \to \infty$. □

#### 4.1.3 Planning Policy Improvement

After attaining the corresponding plan value, the policy $\pi$ can be updated with

$$\pi_{\text{new}} = \arg\max_{\pi\in\Pi}\sum_{\boldsymbol{\tau}_t^k\in\mathcal{A}^k}\boldsymbol{\pi}^k(\boldsymbol{\tau}_t^k|s_t)Q^{\pi_{\text{old}}}(s_t, \boldsymbol{\tau}_t^k) \tag{8}$$

for each state. We will show that $\pi_{\text{new}}$ achieves greater plan value than $\pi_{\text{old}}$ after applying Eq.(8).

**Lemma 2** (Planning Policy Improvement)**.** *Given any mapping $\pi_{\text{old}} \in \Pi : \mathcal{S} \to \Delta(\mathcal{A})$ with $|\mathcal{A}| < \infty$, update $\pi_{\text{old}}$ to $\pi_{\text{new}}$ with Eq.(8), then $Q^{\pi_{\text{new}}}(s_t, \boldsymbol{\tau}_t^k) \ge Q^{\pi_{\text{old}}}(s_t, \boldsymbol{\tau}_t^k)$, $\forall s_t \in \mathcal{S}$, $\boldsymbol{\tau}_t^k \in \mathcal{A}^k$.*

*Proof.* By definition of the state value function, we have

$$\begin{aligned} V^{\pi_{\text{old}}}(s_t) =& \sum_{\boldsymbol{\tau}_t^k\in\mathcal{A}^k}\boldsymbol{\pi}_{\text{old}}^k(\boldsymbol{\tau}_t^k|s_t)Q^{\pi_{\text{old}}}(s_t, \boldsymbol{\tau}_t^k) \\ \le& \sum_{\boldsymbol{\tau}_t^k\in\mathcal{A}^k}\boldsymbol{\pi}_{\text{new}}^k(\boldsymbol{\tau}_t^k|s_t)Q^{\pi_{\text{old}}}(s_t, \boldsymbol{\tau}_t^k), \end{aligned} \tag{9}$$

for any $s_t \in \mathcal{S}$. Reapplying the inequality (9), the result is given by

$$
\begin{aligned}
&Q^{\pi_{\text{old}}}(s_t, \boldsymbol{\tau}_t^k) = \mathbb{E}_{p,r}\left[\sum_{m=0}^{k-1} \gamma^m r_{t+m} + \gamma^k V^{\pi_{\text{old}}}(s_{t+k})\right] \\
\leq &\mathbb{E}_{p,r}\left[\sum_{m=0}^{k-1} \gamma^m r_{t+m} + \gamma^k \sum_{\boldsymbol{\tau}^k \in \mathcal{A}^k} \boldsymbol{\pi}_{\text{new}}^k(\boldsymbol{\tau}^k | s_{t+k}) Q^{\pi_{\text{old}}}(s_{t+k}, \boldsymbol{\tau}^k)\right] \\
= &\mathbb{E}_{p,r,\pi_{\text{new}}}\left[\sum_{m=0}^{2k-1} \gamma^m r_{t+m} + \gamma^{2k} V^{\pi_{\text{old}}}(s_{t+2k})\right] \leq \cdots \\
\leq &\mathbb{E}_{p,r,\pi_{\text{new}}}\left[\sum_{m=0}^{\infty} \gamma^m r_{t+m}\right] = Q^{\pi_{\text{new}}}(s_t, \boldsymbol{\tau}_t^k),
\end{aligned}
\tag{10}
$$

$\forall s_t \in \mathcal{S}, \boldsymbol{\tau}_t^k \in \mathcal{A}^k$. □

#### 4.1.4 Planning Policy Iteration

The whole planning policy iteration process alternates between planning policy evaluation and planning policy improvement until the sequence of policy $\{\pi_k\}$ converges to the optimal policy $\pi_*$ whose plan value of any state-plan pair is the greatest among all $\pi \in \Pi$.

**Theorem 1** (Planning Policy Iteration)**.** *Given any initial mapping $\pi_0 \in \Pi : \mathcal{S} \to \Delta(\mathcal{A})$ with $|\mathcal{A}| < \infty$, compute corresponding $Q^{\pi_i}$ in planning policy evaluation step and update $\pi_i$ to $\pi_{i+1}$ in planning policy improvement step for all $i \in N$, $\{\pi_i\}$ will converge to the optimal policy $\pi_*$ that $Q^{\pi_*}(s_t, \boldsymbol{\tau}_t^k) \geq Q^{\pi}(s_t, \boldsymbol{\tau}_t^k)$, $\forall s_t \in \mathcal{S}$, $\boldsymbol{\tau}_t^k \in \mathcal{A}^k$, and $\pi \in \Pi$.*

*Proof.* The sequence $\{\pi_i\}$ will converge to some $\pi_*$ since $\{Q^{\pi_i}\}$ is monotonically increasing with $i$ and bounded above $\pi \in \Pi$. By Lemma 2, at convergence, we can't find any $\pi \in \Pi$ satisfying

$$
V^{\pi_*}(s_t) < \sum_{\boldsymbol{\tau}_t^k \in \mathcal{A}^k} \boldsymbol{\pi}^k(\boldsymbol{\tau}_t^k | s_t) Q^{\pi_*}(s_t, \boldsymbol{\tau}_t^k), \tag{11}
$$

for any $s_t \in \mathcal{S}$. Then we obtain

$$
\begin{aligned}
&Q^{\pi_*}(s_t, \boldsymbol{\tau}_t^k) = \mathbb{E}_{p,r}\left[\sum_{m=0}^{k-1} \gamma^m r_{t+m} + \gamma^k V^{\pi_*}(s_{t+k})\right] \\
\geq &\mathbb{E}_{p,r}\left[\sum_{m=0}^{k-1} \gamma^m r_{t+m} + \gamma^k \sum_{\boldsymbol{\tau}^k \in \mathcal{A}^k} \boldsymbol{\pi}^k(\boldsymbol{\tau}^k | s_{t+k}) Q^{\pi_*}(s_{t+k}, \boldsymbol{\tau}^k)\right] \\
= &\mathbb{E}_{p,r,\pi}\left[\sum_{m=0}^{2k-1} \gamma^m r_{t+m} + \gamma^{2k} V^{\pi_*}(s_{t+2k})\right] \geq \cdots \\
\geq &\mathbb{E}_{p,r,\pi}\left[\sum_{m=0}^{\infty} \gamma^m r_{t+m}\right] = Q^{\pi}(s_t, \boldsymbol{\tau}_t^k),
\end{aligned}
\tag{12}
$$

$\forall s_t \in \mathcal{S}, \boldsymbol{\tau}_t^k \in \mathcal{A}^k$, and $\pi \in \Pi$. Hence $\pi_*$ is the optimal policy. □

### 4.2 Model-based Planning Policy Learning with Multi-step Plan Value Estimation

The tabular planning policy iteration cannot be directly applied to scenarios with inaccessible dynamics functions and continuous environmental state-action space. Therefore, we propose a neuron-based algorithm based on planning policy iteration for general application, called **M**odel-based Planning **P**olicy Learning with Multi-step **P**lan **V**alue **E**stimation (**MPPVE**).

**Algorithm 1** MPPVE

**Input**: Initial neural parameters $\theta$, $\phi$, $\psi$, $\psi^-$, plan length $k$, environment buffer $\mathcal{D}_{\text{env}}$, model buffer $\mathcal{D}_{\text{model}}$, start size $U$, batch size $B$, and learning rate $\lambda_Q$, $\lambda_\pi$.

```
Explore for U environmental steps and add data to D_env
for N epochs do
    Train model p_θ on D_env by maximizing Eq.(13)
    for E steps do
        Perform action in the environment according to π_φ; add the
        environmental transition to D_env
        for M model rollouts do
            Sample s_t from D_env to make model rollout using policy
            π_φ; add generated samples to D_model
        end for
        for G critic updates do
            Sample B k-step trajectories from D_env ∪ D_model to up-
            date critic Q_ψ via ψ ← ψ − λ_Q ∇̂_ψ J_Q(ψ) by Eq.(18)
            Update target critic via ψ^- ← τψ + (1 − τ)ψ^-
        end for
        Sample B states from D_env to update policy π_φ via φ ←
        φ − λ_π ∇̂_φ J_{π^k}(φ) by Eq.(21)
    end for
end for
```

As shown in Figure 2(c), our MPPVE adopts the framework of model-based policy optimization [14], which is divided into dynamics model $p_\theta$, actor $\pi_\phi$ and critic $Q_\psi$, where $\theta$, $\phi$ and $\psi$ are neural parameters. The algorithm will be described in three parts: 1) model learning; 2) multi-step plan value estimation; 3) model-based planning policy improvement.

#### 4.2.1 Model Learning

Like MBPO [14], we use an ensemble neural network that takes the state-action pair as input and outputs Gaussian distribution of the next state and reward to jointly represent the dynamics function $p$ and the reward function $r$. That is, $p_\theta(s_{t+1}, r_t | s_t, a_t)$ comes from $\mathcal{N}(\mu_\theta(s_t, a_t), \Sigma_\theta(s_t, a_t))$, where $\mu_\theta(s_t, a_t)$ and $\Sigma_\theta(s_t, a_t))$ are the outputs of the network. The environmental model is trained to maximize the expected likelihood:

$$
J_p(\theta) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim \mathcal{D}_{\text{env}}}\left[\log p_\theta(s_{t+1}, r_t | s_t, a_t)\right]. \tag{13}
$$

#### 4.2.2 Multi-step Plan Value Estimation

In order to approximate $k$-step plan value function with continuous inputs, we use a deep Q-network [21, 19]: $Q_\psi(s_t, \boldsymbol{\tau}_t^k)$ with parameters $\psi$. The plan value is estimated by minimizing the expected multi-step TD error:

$$
J_Q(\psi) = \mathop{\mathbb{E}}_{(s_t, \boldsymbol{\tau}_t^k, \boldsymbol{r}_t^k, s_{t+k}) \sim \mathcal{D}_{\text{env}} \cup \mathcal{D}_{\text{model}}}\left[l_{\text{TD}}(s_t, \boldsymbol{\tau}_t^k, \boldsymbol{r}_t^k, s_{t+k})\right], \tag{14}
$$

with

$$
\boldsymbol{r}_t^k = (r_t, r_{t+1}, ..., r_{t+k-1}), \tag{15}
$$

$$
l_{\text{TD}}(s_t, \boldsymbol{\tau}_t^k, \boldsymbol{r}_t^k, s_{t+k}) = \frac{1}{2}\left(Q_\psi(s_t, \boldsymbol{\tau}_t^k) - y_{\psi^-}(\boldsymbol{r}_t^k, s_{t+k})\right)^2, \tag{16}
$$

$$
y_{\psi^-}(\boldsymbol{r}_t^k, s_{t+k}) = \sum_{m=0}^{k-1} \gamma^m r_{t+m} + \gamma^k \mathbb{E}_{\hat{\boldsymbol{\tau}}^k}\left[Q_{\psi^-}(s_{t+k}, \hat{\boldsymbol{\tau}}^k)\right], \tag{17}
$$

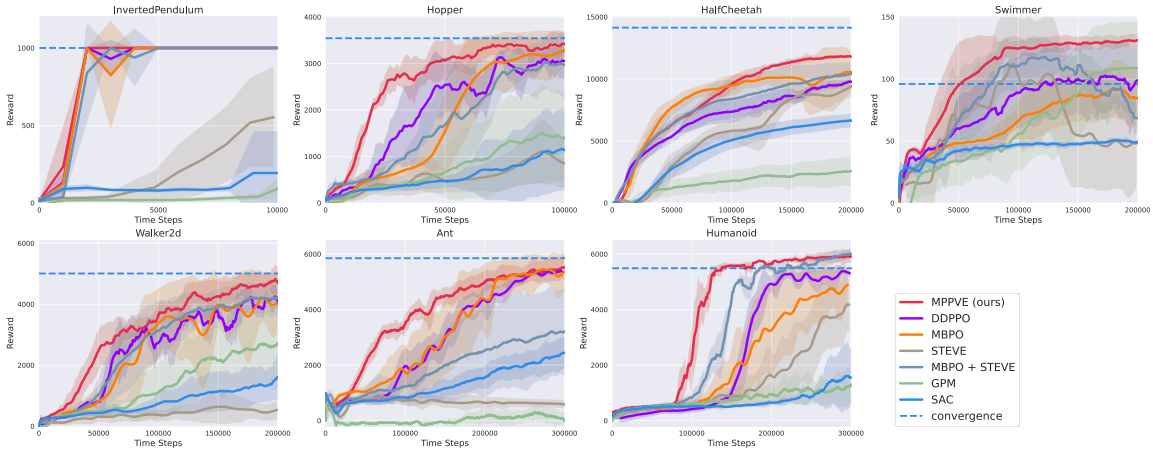



**Figure 3.** Learning curves of our MPPVE (red) and other six baselines on MuJoCo (v3 version) continuous control tasks. The blue dashed lines indicate the asymptotic performance of SAC in these tasks for reference. The solid lines indicate the mean, and shaded areas indicate the standard error over eight different random seeds. Each evaluation, taken every 1,000 environmental steps, calculates the average return over ten episodes.

where $\psi^-$ is the parameters of the target network, which is used for stabilizing training [21]. The gradient of Eq.(14) can be estimated without bias by

$$\hat{\nabla}_\psi J_Q(\psi) = \left(Q_\psi(s_t, \boldsymbol{\tau}_t^k) - \hat{y}_{\psi^-}(\boldsymbol{r}_t^k, s_{t+k})\right) \nabla_\psi Q_\psi(s_t, \boldsymbol{\tau}_t^k), \tag{18}$$

with

$$\hat{y}_{\psi^-}(\boldsymbol{r}_t^k, s_{t+k}) = \sum_{m=0}^{k-1} \gamma^m r_{t+m} + \gamma^k Q_{\psi^-}(s_{t+k}, \boldsymbol{\tau}_{t+k}^k), \tag{19}$$

where $(s_t, \boldsymbol{\tau}_t^k, \boldsymbol{r}_t^k, s_{t+k})$ is sampled from the replay buffer and $\boldsymbol{\tau}_{t+k}^k$ is sampled according to current planning policy.

### 4.2.3 *Model-based Planning Policy Improvement*

The probabilistic policy $\pi_\phi$ is represented by the conditioned Gaussian distribution of the action. That is, $\pi_\phi(a_t|s_t)$ comes from $\mathcal{N}(u_\phi(s_t), \sigma_\phi(s_t))$, where $u_\phi(s_t)$ and $\sigma_\phi(s_t)$ are the outputs of the policy network. As shown in Eq.(1), a $k$-step model-based planning policy $\boldsymbol{\pi}_{\phi,\theta}^k$ is composed of dynamics model $p_\theta$ and policy $\pi_\phi$. Since the plan value function is represented by a differentiable neural network, we can train model-based planning policy by minimizing

$$J_{\boldsymbol{\pi}^k}(\phi) = \mathbb{E}_{s_t \sim \mathcal{D}_{\text{env}}} \left[ \mathbb{E}_{\boldsymbol{\tau}_t^k \sim \boldsymbol{\pi}_{\phi,\theta}^k(\cdot|s_t)} \left[ -Q_\psi(s_t, \boldsymbol{\tau}_t^k) \right] \right], \tag{20}$$

whose gradient can be approximated by

$$\hat{\nabla}_\phi J_{\boldsymbol{\pi}^k}(\phi) = \left(-\nabla_{\boldsymbol{\tau}_t^k} Q_\psi(s_t, \boldsymbol{\tau}_t^k)\right) \nabla_\phi \boldsymbol{\tau}_t^k, \tag{21}$$

where $\boldsymbol{\tau}_t^k$ is sampled according to current planning policy $\boldsymbol{\pi}_{\phi,\theta}^k$.

During the model-based planning policy improvement step, the actor makes $k$-step plans starting from $s_t$ using policy $\pi_\phi$ and model $p_\theta$. Then, the corresponding state-plan pairs are fed into the neural plan value function to directly compute the gradient (21). Therefore, the policy gradients within these $k$ steps are mainly affected by the bias of plan value estimation at $s_t$, avoiding the influence of action-value estimation with compounding error at the following $k-1$ states in previous model-based methods.

The complete algorithm is described in Algorithm 1. We also utilize model rollouts to generate fake transitions, as MBPO [14]. The primary difference from MBPO is that only the critic is additionally trained with $k$-step rollouts starting from fake states to support larger Update-To-Data (UTD) [8] ratio in our algorithm, while the actor is trained merely with $k$-step rollouts starting from real states at a lower update frequency than the critic. There are two reasons for the modification: 1) plan value is more difficult to estimate than action-value, then the critic can guide the actor only after learning with sufficient and diverse samples; 2) it ensures that the $k$-step policy gradient is computed only on the starting real state, but not on any fake states.

## 5 Experiments

In this section, we conduct three experiments to answer: 1) How well does our method perform on benchmark tasks of reinforcement learning, in contrast to a broader range of state-of-the-art model-free and model-based RL methods? 2) Does our proposed model-based planning policy improvement based on plan value estimation provide more accurate policy gradients than previous model-based RL methods? 3) How does the plan length $k$ affect our method?

### 5.1 *Comparison in RL Benchmarks*

We evaluate our method MPPVE on seven MuJoCo continuous control tasks [30], including InvertedPendulum, Hopper, Swimmer, HalfCheetah, Walker2d, Ant, and Humanoid. All the tasks adopt version v3 and follow the default settings. Four model-based methods and two model-free methods are selected as our baselines:

- **STEVE** [5] uses the $k$-step return computed from the ensemble dynamics model, ensemble reward model, and ensemble Q as Q targets to facilitate an efficient action-value estimation.
- **MBPO** [14] is a representative model-based algorithm. Updating the policy with a mixture of data from the real environment and imaginary branched rollouts improves the efficiency of policy optimization. A comparison to MBPO is necessary since we follow the model-based framework of MBPO.
- **MBPO + STEVE** is a combination of MBPO and STEVE. Replacing the value estimation in MBPO with the value expansion method in STEVE while keeping the original policy optimization process of MBPO, MBPO combined with STEVE enables both

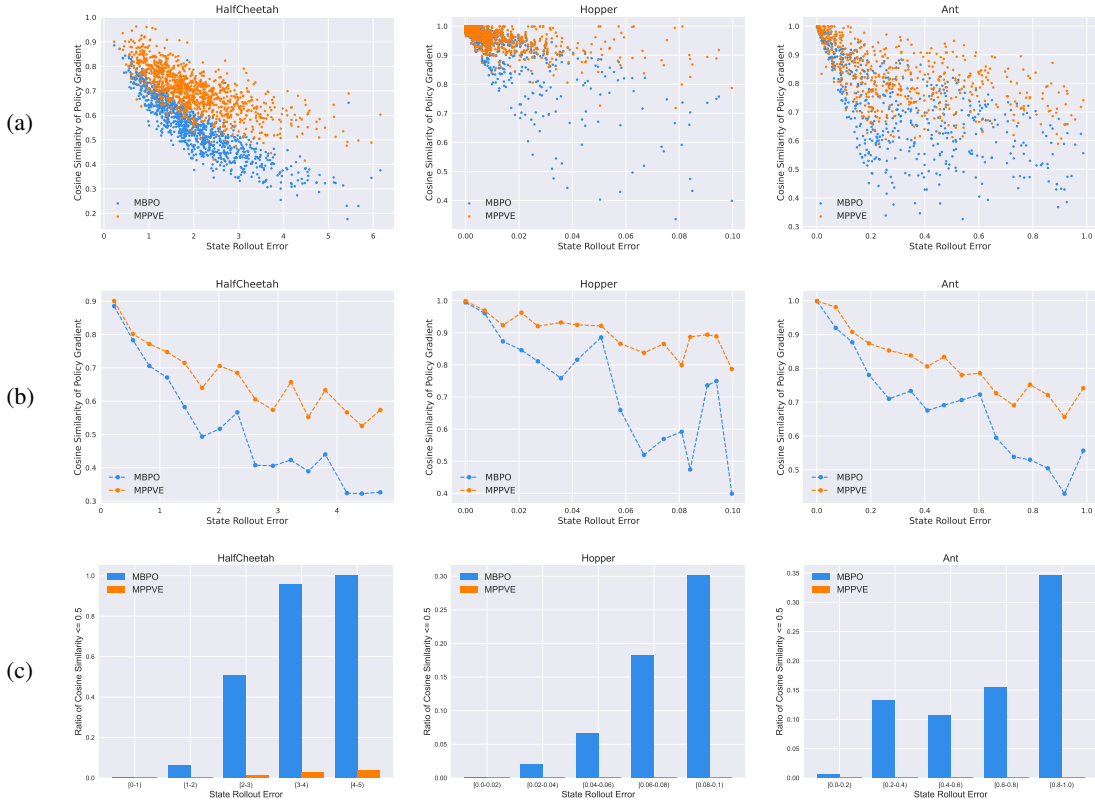



**Figure 4.** We compare the influence of model error on the directions of $k$-step policy gradients given by MPPVE and MBPO on the tasks of HalfCheetah ($k = 3$), Hopper ($k = 4$) and Ant ($k = 4$). We make real branched rollouts in the real world and utilize the learned model to generate the fake, starting from some real states. (a) For each fake rollout and its corresponding real rollout, we show their deviation along with the normalized cosine similarity between their $k$-step policy gradients. (b) We select some samples and plot further, where each group of orange and blue points on the same X-coordinate corresponds to the same starting real state. (c) We count the ratio of points with severely inaccurate k-step policy gradients for each interval of state rollout error.

efficient value estimation and policy optimization. We compare MPPVE with this algorithm to show that we improve the sample efficiency more than purely adding value expansion to MBPO.

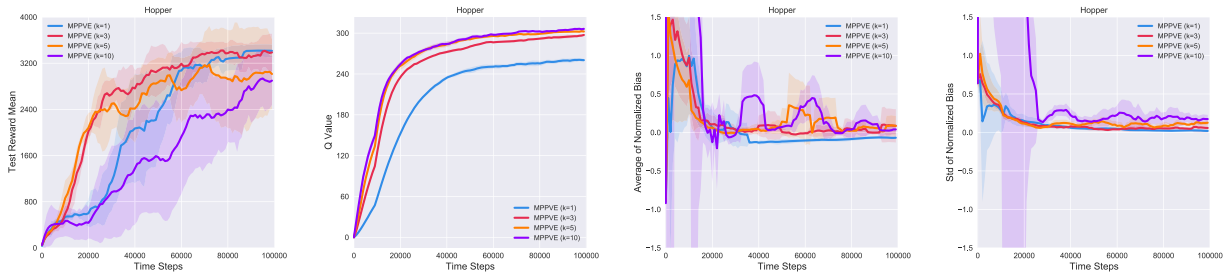
- **DDPPO** [18] adopts a two-model-based learning method to control the prediction error and the gradient error of the dynamics model. In the policy optimization phase, it uses the prediction model to roll out the data and the gradient model to calculate the policy gradient. We compare MPPVE with this model-based algorithm to show that even without a gradient model the policy gradient can be accurate enough to achieve a better sample efficiency.
- **GPM** [33] is a model-free algorithm that also proposes the concept of plan value and features an inherent mechanism for generating temporally coordinated plans for exploration. We compare MPPVE with GPM to demonstrate the power of multi-step plan value in the model-based paradigm.
- **SAC** [13] is the state-of-the-art model-free algorithm that can obtain competitive rewards after training. We use SAC's asymptotic performance for reference to better evaluate all the algorithms.

Figure 3 shows all approaches' learning curves, along with SAC's asymptotic performance. MPPVE achieves great performance after fewer environmental samples than baselines. Take Hopper as an example, MPPVE has achieved 90% performance (about 3000) after 50k steps, while DDPPO, MBPO and MBPO + STEVE, need about 75k steps, and other three methods, STEVE, SAC, GPM, can achieve only about 1000 after 100k steps. MPPVE performs about 1.5x faster than DDPPO, MBPO and MBPO + STEVE, and dominates STEVE, SAC and GPM, in terms of learning speed on the Hopper task. After training, MPPVE can achieve a final performance close to the asymptotic performance of SAC on all these seven MuJoCo benchmarks. These results show that MPPVE has both high sample efficiency and competitive performance.

## 5.2 *On Policy Gradients*

We conduct a study to verify that multi-step policy gradients computed by our MPPVE are less influenced by model error and more accurate than those computed by previous model-based RL methods, as we claimed. Without loss of rigor, we only choose MBPO for comparison since many other model-based methods follow the same way of computing the policy gradients as MBPO.

In order to exclude irrelevant factors as far as possible, we fix the policy and the learned dynamics model after enough environmental samples, then learn the multi-step plan value function and action-value function, respectively, until they are both converged. Next, we measure the influence of model error on the directions of policy gradients computed by MPPVE and MBPO, respectively. Specifically, we sample some real environmental states from the replay buffer and start from them to make $k$-step real rollouts with perfect oracle dynamics and generate $k$-step imaginary rollouts with the learned model. Both MPPVE and MBPO can compute $k$-step policy gradients on the generated fake rollouts and their corresponding real rollouts. MPPVE utilizes $k$-step plan value estimation on the first states of the rollouts to compute $k$-step policy gradients directly, while MBPO needs to compute the single-step policy gradients with action-value estimation




(a) Performance (b) Policy evaluation (c) Average of value bias (d) Standard error of value bias

**Figure 5.** Study of plan length on Hopper task, through comparison of MPPVE with $k = 1$, $k = 3$, $k = 5$ and $k = 10$ respectively. (a) Evaluation of episodic reward during the learning process. (b) Learning plan value estimation for the same fixed policy. (c) Average of the bias between neural value estimation and true value evaluation over state-action(s) space. (d) Standard error of the bias between neural value estimation and true value evaluation over state-action(s) space.

on all states of the rollouts and averages over the $k$ steps. For each fake rollout and its corresponding real rollout, we can measure their deviation along with the normalized cosine similarity between their $k$-step policy gradients given by MBPO or MPPVE.

Figure 4(a) shows the influence of model error on the directions of $k$-step policy gradients on the tasks of HalfCheetah ($k = 3$), Hopper ($k = 4$) and Ant ($k = 4$), where orange stands for MPPVE while blue stands for MBPO. Each point corresponds to one starting real state for making $k$-step rollouts, and its X-coordinate is the error of the fake rollout from the real rollout, while its Y-coordinate is the normalized cosine similarity between $k$-step policy gradients computed on the real rollout and the fake rollout. On the one hand, the point with more state rollout error tends to have a smaller normalized cosine similarity of policy gradients, both for MPPVE and MBPO. The tendency reveals the influence of model error on the policy gradients. On the other hand, the figure demonstrates that the orange points are almost overall above the blue points, which means that the $k$-step policy gradients computed by our MPPVE are less influenced by model error.

For the sake of clear comparison, we select only about a dozen real states and plot further. Figure 4(b) shows the result, where each group of orange and blue points on the same X-coordinate corresponds to the same starting real state for rollout. We connect the points of the same color to form a line. The orange line is always above the blue line, indicating the advantage of MPPVE in computing policy gradients.

Furthermore, we count the ratio of points with normalized cosine similarity between $k$-step policy gradients computed on real and fake rollouts smaller than 0.5 for each interval of state rollout error, as shown in Figure 4(c). A normalized cosine similarity smaller than 0.5 means that the direction of the $k$-step policy gradient deviates by more than 90 degrees. As for MBPO, the ratio of severely biased policy gradients increases as the state rollout error increases and reaches approximately 1 when the state rollout error is great enough. By contrast, our MPPVE only provides severely inaccurate policy gradients with a tiny ratio, even under a great state rollout error.

In summary, by directly computing multi-step policy gradients via plan value estimation, MPPVE provides policy gradients more accurately than MBPO, whose computation of policy gradients is also adopted by other previous state-of-the-art model-based RL methods.

### 5.3 *On Plan Length*

The previous section empirically shows the advantage of MPPVE. Furthermore, a natural question is what plan length $k$ is appropriate. Intuitively, a larger length allows the policy to be updated on longer model rollouts, which may improve the sample efficiency. Nevertheless, it also makes learning the plan value function more difficult. We next make ablations to better understand the effect of plan length $k$.

Figure 5 presents an empirical study of this hyper-parameter. The performance increases first and then decreases as plan length $k$ increases. Specifically, $k = 3$ and $k = 5$ learn fastest at the beginning of training followed by $k = 1$ and $k = 10$, while $k = 1$ and $k = 3$ are more stable than $k = 5$ and $k = 10$ in the subsequent training phase and also perform better. Figure 5(b), 5(c) and 5(d) explain why they perform differently. We plot their policy evaluation curves for the same fixed policy in Figure 5(b). It indicates that a larger $k$ can make learning the plan value function faster. We then quantitatively analyze their estimation quality in Figure 5(c) and 5(d). Specifically, we define the normalized bias of the $k$-step plan value estimation $Q_\psi(s_t, \boldsymbol{\tau}_t^k)$ to be $(Q_\psi(s_t, \boldsymbol{\tau}_t^k) - Q^\pi(s_t, \boldsymbol{\tau}_t^k))/|E_{s\sim\rho^\pi}[E_{\boldsymbol{\tau}^k\sim\boldsymbol{\pi}^k(\cdot|s)}[Q^\pi(s, \boldsymbol{\tau}^k)]]|$, where actual plan value $Q^\pi(s_t, \boldsymbol{\tau}_t^k)$ is obtained by Monte Carlo sampling in the real environment. Strikingly, compared to the other two settings, $k = 5$ and $k = 10$ have a high normalized average and standard error of bias during training, indicating the value function with too large plan length is hard to fit and causes fluctuations in policy learning. $k = 3$ achieves a trade-off between stability and learning speed of plan value estimation, so it performs best.

In conclusion, enlarging $k$ felicitously can lead to great sample efficiency. However, an extremely large $k$ lets the training of MPPVE become unstable. The problem of stabilizing MPPVE when the plan length is large to improve the sample efficiency further is thrown out.

## 6 Conclusion

In this work, we propose a novel model-based RL method, namely **M**odel-based Planning **P**olicy Learning with Multi-step **P**lan **V**alue **E**stimation (MPPVE). The algorithm is from the tabular planning policy iteration, whose theoretical derivation shows that any initial policy can converge to the optimal policy when applied in tabular settings. For general continuous settings, we empirically show that directly computing multi-step policy gradients via plan value estimation, the key idea of MPPVE, is less influenced by model error and provides more accurate policy gradients than previous model-based RL methods. Experimental results demonstrate that MPPVE achieves better sample efficiency than previous state-of-the-art model-free and model-based methods while retaining competitive performance on several continuous control tasks. In the future, we will explore the scalability of MPPVE and study how to estimate plan value stably when the plan length is large to improve the sample efficiency further.

## 7 Acknowledgments

This work is supported by National Key Research and Development Program of China (2020AAA0107200), the National Science Foundation of China (61921006), and The Major Key Project of PCL (PCL2021A12).

## References

[1] Kavosh Asadi, Evan Cater, Dipendra Misra, and Michael L. Littman, 'Towards a simple approach to multi-step model-based reinforcement learning', *CoRR*, **abs/1811.00128**, (2018).

[2] Kavosh Asadi, Dipendra Misra, Seungchan Kim, and Michael L. Littman, 'Combating the compounding-error problem with a multi-step model', *CoRR*, **abs/1905.13320**, (2019).

[3] Kristopher De Asis, J. Fernando Hernandez-Garcia, G. Zacharias Holland, and Richard S. Sutton, 'Multi-step reinforcement learning: A unifying algorithm', in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI'18)*, New Orleans, Louisiana, (2018).

[4] Zdravko I. Botev, Dirk P. Kroese, Reuven Y. Rubinstein, and Pierre L'Ecuyer, 'Chapter 3 - the cross-entropy method for optimization', in *Handbook of Statistics*, volume 31 of *Handbook of Statistics*, 35–59, Elsevier, (2013).

[5] Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee, 'Sample-efficient reinforcement learning with stochastic ensemble value expansion', (2018).

[6] E.F. Camacho and C.B. Alba, *Model Predictive Control*, Advanced Textbooks in Control and Signal Processing, Springer London, 2013.

[7] Tong Che, Yuchen Lu, George Tucker, Surya Bhupatiraju, Shane Gu, Sergey Levine, and Yoshua Bengio. Combining model-based and model-free RL via multi-step control variates. https://openreview.net, 2018.

[8] Xinyue Chen, Che Wang, Zijian Zhou, and Keith W. Ross, 'Randomized ensembled double q-learning: Learning fast without a model', in *9th International Conference on Learning Representations (ICLR'21)*, Virtual Conference, (2021).

[9] Xiong-Hui Chen, Yang Yu, Zheng-Mao Zhu, Zhihua Yu, Zhenjun Chen, Chenghe Wang, Yinan Wu, Hongqiu Wu, Rong-Jun Qin, Ruijin Ding, and Fangsheng Huang, 'Adversarial counterfactual environment model learning', *CoRR*, **abs/2206.04890**, (2022).

[10] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine, 'Deep reinforcement learning in a handful of trials using probabilistic dynamics models', in *Advances in Neural Information Processing Systems 31 (NeurIPS'18)*, Montréal, Canada, (2018).

[11] Ignasi Clavera, Yao Fu, and Pieter Abbeel, 'Model-augmented actor-critic: Backpropagating through paths', in *8th International Conference on Learning Representations (ICLR'20)*, Addis Ababa, Ethiopia, (2020).

[12] Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I. Jordan, Joseph E. Gonzalez, and Sergey Levine, 'Model-based value estimation for efficient model-free reinforcement learning', *CoRR*, **abs/1803.00101**, (2018).

[13] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine, 'Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor', in *Proceedings of the 35th International Conference on Machine Learning (ICML'18)*, Stockholm, Sweden, (2018).

[14] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine, 'When to trust your model: Model-based policy optimization', in *Advances in Neural Information Processing Systems 32 (NeurIPS'19)*, Vancouver, Canada, (2019).

[15] Péter Karkus, Xiao Ma, David Hsu, Leslie Pack Kaelbling, Wee Sun Lee, and Tomás Lozano-Pérez, 'Differentiable algorithm networks for composable robot learning', in *Robotics: Science and Systems XV (RSS'19)*, Freiburg im Breisgau, Germany, (2019).

[16] Nan Rosemary Ke, Amanpreet Singh, Ahmed Touati, Anirudh Goyal, Yoshua Bengio, Devi Parikh, and Dhruv Batra, 'Modeling the long term future in model-based reinforcement learning', in *7th International Conference on Learning Representations (ICLR'19)*, New Orleans, LA, (2019).

[17] Hang Lai, Jian Shen, Weinan Zhang, and Yong Yu, 'Bidirectional model-based policy optimization', in *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*, Virtual Conference, (2020).

[18] Chongchong Li, Yue Wang, Wei Chen, Yuting Liu, Zhi-Ming Ma, and Tie-Yan Liu, 'Gradient information matters in policy optimization by back-propagating through model', in *10th International Conference on Learning Representations (ICLR'22)*, Virtual Conference, (2022).

[19] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra, 'Continuous control with deep reinforcement learning', in *4th International Conference on Learning Representations (ICLR'16)*, San Juan, Puerto Rico, (2016).

[20] Yuping Luo, Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma, 'Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees', in *7th International Conference on Learning Representations (ICLR'19)*, New Orleans, LA, (2019).

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis, 'Human-level control through deep reinforcement learning', *Nature*, **518**(7540), 529–533, (2015).

[22] Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey Levine, 'Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning', in *2018 IEEE International Conference on Robotics and Automation (ICRA'18)*, Brisbane, Australia, (2018).

[23] Masashi Okada, Luca Rigazio, and Takenobu Aoshima, 'Path integral networks: End-to-end differentiable optimal control', *CoRR*, **abs/1706.09597**, (2017).

[24] Feiyang Pan, Jia He, Dandan Tu, and Qing He, 'Trust the model when it is confident: Masked model-based actor-critic', in *Advances in Neural Information Processing Systems 33 (NeurIPS'20)*, Virtual Conference, (2020).

[25] John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz, 'Trust region policy optimization', in *Proceedings of the 32nd International Conference on Machine Learning (ICML'15)*, Lille, France, (2015).

[26] Wenjie Shang, Qingyang Li, Zhiwei Qin, Yang Yu, Yiping Meng, and Jieping Ye, 'Partially observable environment estimation with uplift inference for reinforcement learning based recommendation', *Machine Learning*, **110**(9), 2603–2640, (2021).

[27] Jing-Cheng Shi, Yang Yu, Qing Da, Shi-Yong Chen, and Anxiang Zeng, 'Virtual-taobao: Virtualizing real-world online retail environment for reinforcement learning', in *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI'19)*, Honolulu, Hawaii, (2019).

[28] Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn, 'Universal planning networks: Learning generalizable representations for visuomotor control', in *Proceedings of the 35th International Conference on Machine Learning (ICML'18)*, Stockholm, Sweden, (2018).

[29] Richard S. Sutton, 'Integrated architectures for learning, planning, and reacting based on approximating dynamic programming', in *Proceedings of the 7th International Conference on Machine Learning (ICML'90)*, Austin, Texas, (1990).

[30] Emanuel Todorov, Tom Erez, and Yuval Tassa, 'Mujoco: A physics engine for model-based control', in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'20)*, Vilamoura, Portugal, (2012).

[31] Tian Xu, Ziniu Li, and Yang Yu, 'Error bounds of imitating policies and environments for reinforcement learning', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2021).

[32] Yang Yu, 'Towards sample efficient reinforcement learning', in *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*, Stockholm, Sweden, (2018).

[33] Haichao Zhang, Wei Xu, and Haonan Yu, 'Generative planning for temporally coordinated exploration in reinforcement learning', in *10th International Conference on Learning Representations (ICLR'22)*, Virtual Conference, (2022).

[34] Zheng-Mao Zhu, Xiong-Hui Chen, Hong-Long Tian, Kun Zhang, and Yang Yu, 'Offline reinforcement learning with causal structured world models', *CoRR*, **abs/2206.01474**, (2022).