FARSIGHTER: EFFICIENT MULTI-STEP EXPLORATION FOR DEEP REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Exploration in deep reinforcement learning (RL), especially uncertainty-based exploration, plays a key role in improving sample efficiency and boosting total reward. Uncertainty-based exploration methods often measure the uncertainty (variance) of the value function; However, existing exploration strategies either only consider the uncertain impact of next "one-step" or propagate the uncertainty for all the remaining steps in an episode. Neither approach can explicitly control the bias-variance trade-off of the value function. In this paper, we propose Farsighter, an explicit multi-step uncertainty exploration framework in DRL. Specifically, Farsighter considers the uncertainty of exact k future steps and it can adaptively adjust k. In practice, we learn Bayesian posterior over Q-function to approximate uncertainty in each step. In model-free cases, we recursively deploy Thompson sampling on the learned posterior distribution for k steps and in model-based cases, we solve a joint optimization problem of higher dimension for a tree-based model. Our method can work on general tasks with high/low-dimensional states, discrete/continuous actions, and sparse/dense rewards. Empirical evaluations show that Farsighter outperforms SOTA explorations on a wide range of Atari games, robotic manipulation tasks, and general RL tasks.

1 INTRODUCTION

While reinforcement learning (RL) has shown great performance in tackling tasks such as robots Schulman et al. (2015), Atari games Mnih et al. (2015), and AlphaGo Silver et al. (2016), significant barriers remain to applying RL in applications with sparse rewards. Sparse rewards can rarely provide informative feedback on actions. This problem is particularly challenging when the reward of an action is only obtained after a long sequence of actions. Recently, exploration strategies are proposed to tackle the above problems Yang et al. (2021): 1) Uncertainty-based methods Janz et al. (2019) estimate the uncertainty (variance) of Q values via Bayesian posterior and take actions based on its uncertainty; and 2) Intrinsic-motivation methods Chentanez et al. (2005); Houthooft et al. (2016) take the uncertainties of states as intrinsic rewards and use them as exploration bonuses. However, these one-step uncertainty estimation methods still cannot solve the problem efficiently. First, none of them works very well on the tasks with sparse rewards, e.g. Skiing. Worse still, these methods introduce a new uncertainty vanishing issue Ecoffet et al. (2019): as an agent explores the environment and becomes familiar with a local area after a number of steps, the uncertainty of the area diminishes, thus the agent loses its exploration ability and may get stuck in a local area. Because of those problems, the Q-value estimation is usually biased since the agent cannot explore the environment enough. To handle the problems, some other uncertainty-based algorithms (e.g. OB2I Bai et al. (2021), WQL Metelli et al. (2019)) propagate the uncertainty in a long-term manner: they accumulate uncertainties for all the remaining steps in an episode. In such cases, because the environments usually contain thousands of steps (e.g. Atari), the uncertainty (variance) estimation is usually very large. Both the one-step uncertainty and uncertainty propagation methods lack the ability to explicitly adjust the number of future uncertainty steps and thus it is difficult to use them to explicitly control the bias-variance(uncertainty) trade-off of the value function.

To address this challenge, in this paper, we propose Farsighter, an explicit multi-step uncertainty exploration framework in DRL. Existing exploration methods only consider the uncertain impact of next "one-step" or propagate the uncertainty for all the remaining steps in an episode. In contrast, Farsighter considers the uncertainty for k future steps and it can explicitly adjust k to consider the

multi-step impact. First, is beneficial in cases with long-term sparse rewards. The agent learns the impact of the current action on future k step rewards even if no immediate reward is given. Second, considering future uncertainties helps escape the local familiar areas, thus alleviating the uncertainty vanishing issue. Third, the number of k allows the agent to explicitly adjust the number of future steps and thus balance the bias-variance of Q estimation.

Specifically, Farsighter first learns Bayesian posterior over Q-function/action to approximate uncertainty. In the model-free case, Farsighter estimate multi-step uncertainty using a recursive Gaussian process while addressing its computational complexity. Farsighter applies in both discrete and continuous action tasks. For discrete action tasks, we deploy the value-based DDQN Van Hasselt et al. (2016) and use Bayesian linear regression for the last layer of the Q-network to approximate the Bayesian posterior over Q-function. For continuous action tasks, we build on NAF Gu et al. (2016), and use the Bayesian Neural network to approximate the Bayesian posterior over actions of the Q-function. This allows us to directly incorporate the uncertainty over the Q-function in each step. To estimate the "k-step" uncertainty in practice without exponential computational complexity, we formulate the problem as a recursive Gaussian process, in which we recursively deploy Thompson sampling on the learned posterior distributions for k steps. In model-based case, we consider the multi-step uncertainty with a tree-based model. In each step, we sample action instances based on the posterior and then get next states from the dynamic model. We perform the process for k step and then solve a joint optimization problem of higher dimension to get each action.

In summary, we make the following contributions:

- We propose Farsighter that allows explicit multi-step exploration that considers the k-step uncertainty. The Farsighter helps address the issues of long-term sparse rewards and uncertainty vanishing.
- Farsighter works in model-free and model-based cases. In model-free cases, we formulate the multi-step uncertainty estimation problem as a recursive Gaussian process and approximate it using recursive Thompson sampling. In model-based cases, we formulate the problem as a tree-based model and solve a joint optimization problem of higher dimensions.
- We develop Farsighter implementations that apply in both discrete and continuous action tasks. Moreover, we also develop an adaptive Farsighter to further improve the exploration performance.
- Empirical results show that the proposed method is scalable and outperforms SOTA on a wide range of RL tasks with high/low-dimensional states, discrete/continuous actions, and sparse/dense rewards, including hard-to-explore problems such as high-dimensional Atari games and continuous control robotic manipulation tasks.

2 Related Work

Uncertainty-based methods usually model the uncertainty via the Bayesian posterior of the value function. The agent is encouraged to explore the unknown environment with high uncertainty. We can categorize existing methods to two types: parametric posterior and the non-parametric posterior.

Parametric Posterior based Exploration: Parametric posterior is usually learned by Bayesian regression in linear MDPs, where the transition and reward functions are assumed to be linear to state-action features. RLSVI Osband et al. (2016b) performs Bayesian regression in linear MDPs so that it can sample the value function through Thompson Sampling. BDQN Azizzadenesheli et al. (2018) performs Bayesian Linear Regression (BLR) in the last layer of the Q-network. It approximately considers the last-layer Q-network as a linear MDP problem. Successor Uncertainty Janz et al. (2019) approximates the posterior through successor features which are linear to the Q value of the corresponding state-action pairs.

Non-Parametric Posterior based Exploration: Bootstrap-based exploration constructs a non-parametric posterior based on the bootstrapped value functions. Bootstrapped DQN Osband et al. (2016a) maintains several independent Q-estimators and randomly samples one of them at the beginning of each episode, which enables the agent to conduct exploration.

The above methods only consider the uncertainty in next one-step. Moreover, some methods propagate the uncertainty in a long-term manner. They accumulate uncertainties for all the remaining

steps in an episode. In model-free cases, WQL Metelli et al. (2019) approximates the parametric posterior distribution based on Wasserstein barycenters. OB2I Bai et al. (2021) performs backward induction of bootstrapped-based uncertainty to capture the long-term uncertainty in an whole episode. In model-based cases, Plan2explore Sekar et al. (2020) leverages planning to explore by imagining the consequences of the actions using the current dynamic model. The planning objective is to maximize expected novelty over all future steps.

3 PRELIMINARIES

3.1 MARKOV DECISION PROCESS (MDP)

A MDP is represented by the tuple (S, A, R, P, γ) Sutton & Barto (2018), where S is the set of states; A is the set of actions; R is the reward function; P is the transition probability function and γ is the reward discount factor. The objective of an MDP is to learn a policy π to maximize the discounted cumulative reward. Given a state s and action a, the Q function is $Q(s,a) = \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty}\gamma^t R(s_t, a_t, s_{t+1})|s_0 = s, a_0 = a]$.

4 FARSIGHTER: MULTI-STEP EXPLORATION

In this section, we introduce Farsighter that performs exploration by considering the uncertainty of the next "k-step". In Sec. 4.1, we first motivate why we need the multi-step uncertainty estimation. In Sec. 4.2 and 4.3, we present how to estimate uncertainty with discrete actions and continuous actions. In Sec. 4.4 and 4.5, we introduce how to perform multi-step exploration in model-free and model-based cases. In Sec. 4.6, we show how to adaptive choose the number of k.

4.1 MOTIVATION OF MULTI-STEP EXPLORATION

Assume the ground truth of a Q-value is Q_g, we define the Bayesian posterior of a Q-estimation as $\mathcal{N}(Q_e, \epsilon)$, where Q_e is the mean value and ϵ is the variance (uncertainty) of the Q-estimation. We call the distance of $|Q_g - Q_e|$ as the bias of the Q-estimation.

Empirically, in one-step uncertainty estimation methods (e.g. BDQN), the variance ϵ is usually small, which leads the agent cannot explore the environment enough. Thus the Q-estimation usually has high bias. On the other hand, in uncertainty propagation methods (e.g. OB2I), which propagate all the remaining uncertainty in an episode, the Q-estimation is usually less biased, because the variance ϵ is usually very large and the agent can explore more in the environment. However, large variance is at the risk of too much unnecessary exploration and thus slow down the learning convergence speed. Thus we need a method that explicitly adjust the uncertainty exploration steps that balance the bias-variance trade-off. In Sec. 5.1, we empirically show the significance of Farsighter.

4.2 ESTIMATING BAYESIAN UNCERTAINTY WITH DISCRETE ACTIONS

For discrete action cases, we build our algorithm on the DDQN Van Hasselt et al. (2016) and estimate the uncertainty on Q-function. DDQN architecture consists of a deep neural network where the last layer is usually a linear MLP function of the state representation and action. Thus, given any state s and action a, $Q(s,a) = \phi_\theta(s)^T\omega_a$, where $\phi_\theta(s) \in \mathbb{R}^d$ parameterized by θ represents state s and $\omega_a \in \mathbb{R}^d$ is the parameter of the last linear MLP layer on action a.

To estimate the exploration uncertainty, we build Farsighter over DDQN with the Bayesian framework. In the last layer of Q-network $Q(s,a)$, instead of using the linear MLP regression, Farsighter deploys the Gaussian Bayesian linear regression (BLR) (Rasmussen (2003)), which results in an approximated Bayesian posterior on the ω_a and consequently on the Q-function. The Bayesian posterior ω_a is modeled as a Gaussian with $\{\bar{\omega}_a, Cov_a\}$, where $\bar{\omega}_a$ is the posterior mean and Cov_a is the posterior covariance. Moreover, we leverage the re-parameterization trick to write

$$Q(s,a) = \phi_\theta(s)^T\omega_a = \phi_\theta(s)^T(\bar{\omega}_a + \sqrt{Cov_a}z), \tag{1}$$

where z is a random variable $z \sim \mathcal{N}(0, I)$. Through BLR, the agent efficiently approximates the distribution over the Q-values and captures the uncertainty over the Q estimates. In the parameters

updating process, the BLR-based Q-function updates parameters θ and $\bar{\omega}_a, Cov_a$ separately. The process is shown in the Algorithm 1.

Update $\phi_\theta(s)$**:** we update $\phi_\theta(s)$ as the standard DDQN (Eq. 7). We keep the ω_a as the mean value of the posterior $\bar{\omega}_a$ and update θ using the following loss function:

$$(Q(s,a,\theta,\bar{\omega}_a) - r - \gamma Q(s', \arg\max_{a'} Q(s',a',\theta,\bar{\omega}_a), \theta^{target}, \bar{\omega}_a^{target}))^2. \tag{2}$$

Update $\bar{\omega}_a, Cov_a$**:** we update $\bar{\omega}_a$ and Cov_a with fixed $\phi_\theta(s)$. Given a dataset $\mathcal{D} = \{s_i, a_i, y_i\}_{i=1}^D$, where y_i are target values, we construct $|\mathcal{A}|$ disjoint datasets for each action, $\mathcal{D} = \cup_{a\in\mathcal{A}}\mathcal{D}_a$, where $\mathcal{D}_a$ is a set of tuples (s_i, a_i, y_i) with the action $a_i = a$. Let us construct a matrix $\Phi_a \in \Re^{d\times D_a}$, a concatenation of feature column vectors $\{\phi(s_i)\}_{i=1}^{D_a}$, and $\mathbf{y}_a \in \Re^{D_a}$, a concatenation of target values in set $\mathcal{D}_a$. We then approximate the posterior distribution of ω_a as follows

$$\bar{\omega}_a = \frac{1}{\sigma_\epsilon^2} Cov_a \Phi_a \mathbf{y}_a, \quad Cov_a = \left(\frac{1}{\sigma_\epsilon^2}\Phi_a\Phi_a^\top + \frac{1}{\sigma^2} I\right)^{-1}, \tag{3}$$

where $I \in \Re^d$ is an identity matrix. This is the derivation of the BLR, with zero mean prior and as σ and σ_ϵ^2 as the variance of prior and likelihood respectively.

4.3 Estimating Bayesian Uncertainty with Continuous actions

Value-Based methods, like DDQN, suit problems with discrete action spaces. For continuous action cases, we build our algorithm on the NAF Gu et al. (2016) and estimate the uncertainty on actions. NAF architecture consists three output streams $\mu(s|\theta^a), L(s|\theta^P)$, and $V(s|\theta^V)$, as shown in Eq. 8. Usually, the three sub-networks are functions of a shared state representation network $\phi_\theta(s)$. Thus, we have $\mu(s|\theta^a) = \mu(\phi(s)|\theta^a)$, where θ^a is the parameter of layers taking state representation $\phi(s)$ as input and output action a. The network architecture is shown in the Appendix.

Original NAF cannot estimate the uncertainty for actions, in our work, we propose to estimate the exploration uncertainty for continuous actions using a Bayesian neural network (BNN) Kononenko (1989) for the action sub-network $\mu(\phi(s)|\theta^a)$. BNN treats the model weights and output action as variables. Instead of finding a set of optimal estimates, BNN fits the Bayesian posterior distributions for them. Every weight in θ^a is modeled as a Gaussian distribution with a mean and variance. It directly learns the uncertainties of the actions given a state representation $\phi(s)$. To get action, we can sample one set of weights from the distribution. To update the parameters, we update parameters of $\theta^V, \theta^P, \theta$ and θ^a separately.

Update $\theta^V, \theta^P, \theta$**:** we update $\theta^V, \theta^P, \theta$ with a fixed θ^a, which is mean value from the Bayesian posterior. The update rule is same as Eq. 2, replacing $\arg\max_{a'} Q(s',a',\theta,\bar{\omega}_a)$ with $\mu(\phi(s')|\theta^a)$.

Update θ^a**:** to learn the posterior distribution $\mu(\theta^a|(\phi(s),a))$, we fix the parameters of $(\theta^V, \theta^P, \theta)$ and update the parameters of θ^a with the Evidence Lower Bound(ELBO) loss Kononenko (1989). Specifically, we approximate the posterior distribution $\mu(\theta^a|(\phi(s),a))$ with another distribution $\hat{\mu}(\theta^a)$, which is called a variational distribution. We further minimize the KL divergence between them $D_{KL}(\hat{\mu}(\theta^a)||\mu(\theta^a|(\phi(s),a)))$. Based on the variational inference theory Blei et al. (2017), we get the ELBO loss:

$$D_{KL}(\hat{\mu}(\theta^a)||\mu(\theta^a)) - \mathbb{E}_{\theta^a\sim\hat{\mu}}[\log \mu(a|s,\theta^a)] \tag{4}$$

Note that we use BNN for continuous action tasks and BLR for discrete ones. BNN has better performance but at the cost of high computation complexity. Because the dimension of state representation is typically low for continuous action tasks, e.g. robotic manipulation tasks, we consider it computationally acceptable. On the other hand, as discussed in the appendix, BLR does not increase the computation complexity compared to MLP. It is suitable when the dimension of state representation is high, and thus we choose it for discrete action tasks.

4.4 Exploration with Multi-step Uncertainty in model-free Cases

In Sec. 4.2 and 4.3, we show how to estimate the uncertainty. Each step is a Gaussian process with a posterior on Q-function/actions. For example, in discrete cases, the GP posterior applies on the ω_a(Eq. 1) and consequently on the Q-function. In each step, we can sample an instance from the

Algorithm 1 Farsigher: Multi-step Exploration

Initialize θ, θ^{target}, k, Q-variance target ϵ, and $\forall a, \bar{\omega}_a, Cov_a, \bar{\omega}_a^{target}$; Replay buffer $RB = \{\}$

for t=0, k, 2k, 3k... **do**
  $\{r^k, s_{t+k}\}$ = K-STEP($s_t, \theta, \bar{\omega}_a, \sqrt{Cov_a}, \gamma, r^k = 0, itr = 0$)
  Store $\{s_t, a_t, r^k, s_{t+k}\}$ into replay buffer RB
  Sample a mini-batch $\{s_i, a_i, r^k, s_{i+k}\}$ from the latest N steps to alleviate off-policyness bias
  Update the parameters of θ with Eq. 2, where $r = r^k, s' = s_{i+k}$ and keep $\bar{\omega}_a, Cov_a$ fixed
  Every M steps: Update the GP posterior $\{\bar{\omega}_a, Cov_a\}$ for all actions
    if Q-variance $< \epsilon$: k+=1; **else if** Q-variance $> \epsilon$: k-=1; Empty Replay buffer.
  Every N steps: reset $\theta^{target} = \theta, \bar{\omega}_a^{target} = \bar{\omega}_a$
end for

Algorithm 2 K-STEP($s_t, \theta, \bar{\omega}_a, \sqrt{Cov_a}, \gamma, r^k, itr$)

Input: s_t it the current state; $\theta, \bar{\omega}_a$ and Cov_a are parameters of Q-function, γ is the discounted factor; r^k is the discounted sum of k-step rewards; itr is the number of steps in the k loop.
Output: the discounted sum of k-step rewards r^k and the last state after k steps.

if itr=k: **return** r^k, s_{t+1}
Sample $z_t \sim N(0, I)$ and then get a deterministic $Q(s, a)$ with Eq. 1
Take action $a_t = \arg\max_a \phi_\theta(s_t)^T(\bar{\omega}_a + \sqrt{Cov_a} z_t)$
Get next state s_{t+1} and reward r_t by interacting with the environment.
$r^k += \gamma^{itr} * r_t$
return K-STEP($s_{t+1}, \theta, \bar{\omega}_a, \sqrt{Cov_a}, \gamma, r^k, itr + 1$)

posterior. To extend the learning process to k-step in model-free cases, we formulate the "k-step" process as a recursive Gaussian process, which means the Q-function becomes

$$Q(s_t, a_t) = \mathbb{E}_{\tau\sim\pi}[R(s_t, a_t, s_{t+1}) + \gamma R(s_{t+1}, a_{t+1}, s_{t+2}) + ... + \gamma^k \max_{a_{t+k}\in A} Q^*(s_{t+k}, a_{t+k})|s_t, a_t].$$

More specifically, we recursively deploy Thompson sampling on the learned posterior distributions for k steps to approximate the k-step uncertainty. For discrete action cases, we sample a random variable z for Eq. 1 in each step and obtain a deterministic Q-function. Given the deterministic Q-function, we can decide which action maximizes the Q values. For continuous action cases, we sample a set of weights from the BNN posterior θ^a in each step and then directly get maximal action from the sampled weights. After taking the action, we go to the next state from the environment. As shown in Algorithm 2, we recursively deploy the process for k steps and get the last state s_{t+k} and the discounted sum of k-step rewards r^k, where the k-step uncertainties information is stored.

The pseudocode of the whole learning process for discrete action cases is shown in Algorithm 1. Instead of store one-step state and action tuple, we get k-step state s_{t+k} and reward r^k from Algorithm 2. For continuous action cases, the workflow is similar to discrete action cases; we provide the pseudocode in the Appendix. For multi-step updates, we keep the update rule same as one-step updates as mentioned in Sec. 4.2 and 4.3. We only change the way to calculate the target value, $y = r + \gamma Q(s', \arg\max_{a'} Q(s', a', \theta, \bar{\omega}_a), \theta^{target}, \bar{\omega}_a^{target})$, where r is the discounted sum of k-step rewards $r^k$and s' is the last state after k steps s_{t+k}. Thus, our multi-step uncertainty estimation would not increase the computation and the memory complexity. Moreover, to alleviate the bias introduced by off-policyness in multi-step learning, the network is trained using the latest N-step samples, where N is the target network update period, as suggested in Mnih et al. (2016). In addition, since the k-step reward and state are obtained from recursive Thompson sampling and they contain the uncertainty information of the future k steps, the learned Q function also contains the uncertainty information, which is represented on the variance of the posterior. The variance helps us quantify the uncertain impact of the next k-step in turn.

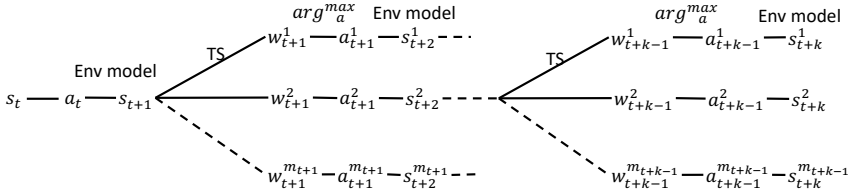


Figure 1: Illustration of the k-step decision tree with m_t actions in each step (discrete case).

4.5 Exploration with Multi-step Uncertainty in Model-based cases

In the last section, we introduce how to apply multi-step exploration in model-free cases. Multi-step exploration can also work if we have the dynamic model of the environment. In the model-based cases, we learn the dynamic model (transition $P(s_{t+1}|s_t, a_t)$ and reward $R(s_t, a_t, s_{t+1})$) using supervise models. In each step, we plan to get an action by considering k-step 'fantasy' steps with the model. For example, to get action a_t with state s_t, we need virtually consider how the next k 'fantasy' steps $(s_{t+1}, a_{t+1}, s_{t+2}, a_{t+2}, ..., s_{t+k}, a_{t+k})$ perform, where $(s_{t+1}, s_{t+2}, ..., s_{t+k})$ are getting from the dynamic model and $(a_{t+1}, a_{t+2}, ..., a_{t+k-1}, a_{t+k})$ are getting from maximizing the Q-function (discrete case) or action sub-network (continuous case).

We assume we can draw action instances from the GP posterior in each step, which is $a^j_t, j = 1, 2, ..\infty$. All the a_t instances $(a^j_t, j = 1, 2, ..\infty)$ in each step constitute the one-step GP posterior. Each a_t instance $(a^j_t, j = 1, 2, ..\infty)$ corresponds to one next state getting from the dynamic model. This essentially means building a discrete scenario tree (Figure 1), where each branch in a node corresponds to a particular fantasized outcome drawn from the posterior.

Solving the k-step problem requires recursive maximization of Q-function to get actions and integration over k steps. Since each step has different GP posteriors based on different states and actions, these nested expectations are analytically intractable; we cannot directly calculate the "k-step" uncertainty distribution. Moreover, the number of instances in the recursive Gaussian process grows exponentially in k. Considering all the possible roll-outs in k steps is computationally difficult. Therefore, we must resort to numerical integration. Letting $m_{t+1}, m_{t+2}, ..., m_{t+k-1}$ denote the number of fantasy samples from the posterior in each step, we have the following approximation.

$$\begin{aligned} Q^*(s_t, a_t) =& \mathbb{E}_{\tau\sim\pi}[R(s_t, a_t, s_{t+1}) + \frac{\gamma}{m_{t+1}} \sum_{j_{t+1}=1}^{m_{t+1}} [R(s_{t+1}, a_{t+1}^{m_{j_{t+1}}}, s_{t+2}^{m_{j_{t+1}}}) + ...+ \\ & \frac{\gamma^k}{m_{t+k-1}} \sum_{j_{t+k}=1}^{m_{t+k-1}} \max_{a_{t+k}\in A} Q^*(s_{t+k}^{m_{j_{t+k}}}, a_{t+k}, \theta, \bar{\omega}_a)]|s_t, a_t]. \end{aligned} \tag{5}$$

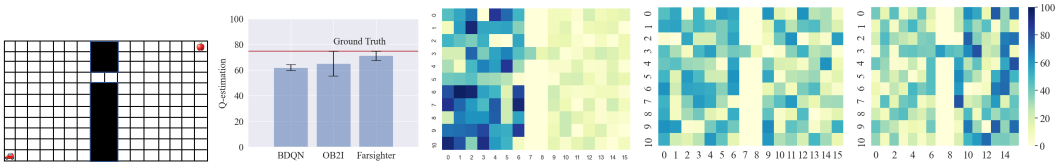
Instead of solving above nested optimization problem, we can solve a joint optimization problem of higher dimension and subsequently extract the optimizer. Thus, we get action a_t in each step with virtually considering next k steps.

$$\begin{aligned} & a_t, \hat{a}_{t+1}, ..., \hat{a_{a+k}} = \underset{a_t, \hat{a_{t+1}}, ..., \hat{a_{a+k}}}{\arg\max} \mathbb{E}_{\tau\sim\pi}[R(s_t, a_t, s_{t+1}) + \frac{\gamma}{m_{t+1}} \sum_{j_{t+1}=1}^{m_{t+1}} R(s_{t+1}, a_{t+1}^{m_{j_{t+1}}}, s_{t+2}^{m_{j_{t+1}}}) \\ & + ... + \frac{\gamma^k}{m_{t+k-1} * \prod_{l=t+1}^{t+k-1} m_l} \sum_{j_{t+1}=1}^{m_{t+1}} ... \sum_{j_{t+k}=1}^{m_{t+k-1}} \max_{a_{t+k}\in A} Q^*(s_{t+k}^{m_{j_{t+1}}...m_{j_{t+k}}}, a_{t+k}, \theta, \bar{\omega}_a)]|s_t, a_t], \end{aligned}$$

where $\hat{a_{t+1}} = a^{j_1}_{t+1}, j_1 = 1, 2, ..., m_{t+1}$; $\hat{a_{t+2}} = a^{j_1 j_2}_{t+2}, j_1 = 1, 2, ..., m_{t+1}, j_2 = 1, 2, ..., m_{t+2}$, and so on. We show the pseudocode of learning process for model-based cases in the Appendix.

4.6 Adaptive k

Empirically, to learn a good policy as soon as possible, it is desirable to have more exploration at the beginning stage and then gradually decrease exploration to increase exploitation. As shown above,



(a) Illustration of the environments (b) Performance of different estimators (c) Heatmap for BDQN (d) Heatmap for OB2I (e) Heatmap for Farsighter

Figure 2: Validation of the effectiveness of multi-step uncertainty.

the amount of uncertainty is represented by the variance of the Bayesian posterior. In principle, we can set a large initial k to enlarge the exploration at the beginning stage and then set posterior variance target to confine the exploration and enlarge the exploitation later. Usually, the variance target are smaller than the variance caused by the large initial k. Based on this intuition, we have developed an adaptive Farsighter. In the adaptive Farsighter, we initial k to be a large number and set a target to the variance. If the variance is smaller than the target, we increase k, otherwise, we decrease it. In this manner, the agent can keep exploring the environment without too much uncertainty. The pseudocode for discrete action cases is shown in the Algorithm 1. We show the affects of different k in Sec. 5.3.

5 Experiments

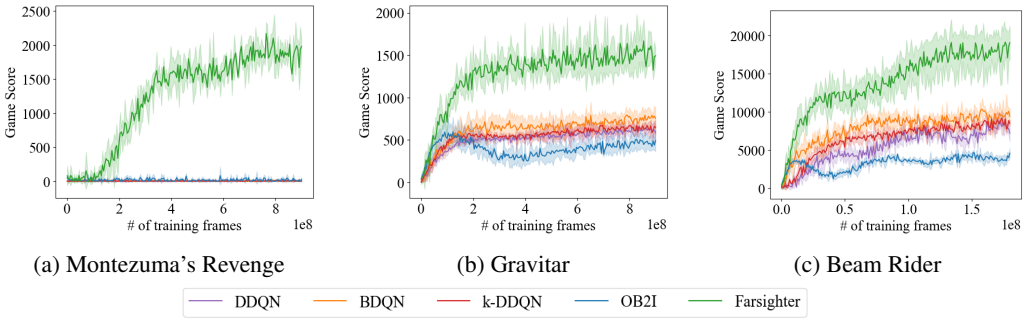
In this section, we investigate the following properties of Farsighter: 1) We illustrate the insight of multi-step uncertainty exploration using a toy example, 2) We compare the performance of Farsighter with SOTA, on a large range of RL tasks, including Atari games and continuous control tasks, and 3) We investigate the effect of a different number of future steps.

5.1 K-step Uncertainty Validation

To illustrate the intuition of multi-step uncertainty, we design a toy maze task as shown in Fig. 2a. The agent (car) starts from the bottom left corner. In each step, the car can go either up, down, left, or right. The car wants to get the apple (top right corner) and it cannot pass the black wall area. The bridge is the only way that connects the left and right sides. The reward is 100 if the car reaches the apple, and -1 otherwise each step.

We further compare the one-step uncertainty exploration (e.g., BDQN), uncertainty propagation (e.g., OB2I) and k-step uncertainty exploration(Farsighter) under same interaction steps (40k) in the game. The optimal Q-value Q_g for the car from the bottom left corner is 75. From Fig. 2b, we can see that the Q-estimation of BDQN is highly biased, as we discussed in Sec. 4.1, because the mean is around 62 which is far from the optimal 75 and the variance is low. On the other hand, the OB2I Q-estimation is less biased, while the variance of OB2I Q-estimation is very large. In comparison, the bias of Farsighter is the lowest and the variance is lower than OB2I.

In addition, we show the heatmap of the number of state visited times for BDQN (Fig. 2c), OB2I(Fig. 2d), and Farsighter (Fig. 2e) . For BDQN, fewer visits occur on the right side of the map and most of the interactions remain on the left side because the car does not cross the bridge often enough and repeatedly explore the left familiar side (uncertainty vanishing). On the other hand, it is easier for the car to cross the bridge with OB2I and Farsighter. More visits occur on the right, which enhances the car reaching to the apple more frequently. However, OB2I performs much over exploration. The visited times for both sides are similar. In comparison, Farsighter visits more on the right and frequently reaches the apple. Intuitively, multi-step uncertainty explorations (Farsighter and OB2I) consider more exploration for further locations. When the car is at the bridge, it is easier to find the new locations on the right, which encourages the car to explore more on the right side. In comparison, the one-step agent (BDQN) takes the left as the local optima area and sticks to it more often. Thus the Q-estimation is biased since the agent cannot explore the environment enough. However, the OB2I performs too much exploration, since the variance is high, which leads to slow converge speed. Farsighter balances the bias-variance trade-off by explicitly adjusting the k.



(a) Montezuma's Revenge (b) Gravitar (c) Beam Rider

Figure 3: The game score for Atari Games.

Moreover, we also study the changes of posterior variance among these exploration methods. the beginning variances are low because the networks are randomly initialized. When the learning starts, the variances increase rapidly to award exploration. After that, the posterior variance in BDQN gradually decreases because as the agent gathers more samples, the uncertainty is vanishing. In comparison, in the Farsighter, even the posterior variance decreases as well earlier, it becomes larger later on (because the agent accesses more states on the right side) and then decreases finally when the learning is converged. The results show that Farsighter alleviates the uncertainty vanishing problem because Farsighter learns high uncertainties on the right side by considering future steps. The OB2I can also help to alleviate the uncertainty vanishing, while it is hard to converge, since the variance is high.

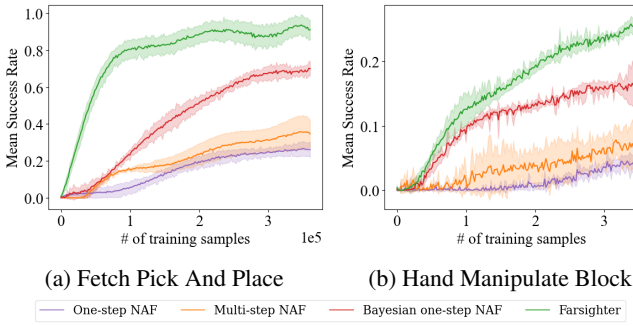
5.2 Exploration Performance

Environments: In model-free cases, we evaluate Farsighter on 49 Atari suite of games including hard-explored games with sparse rewards (e.g., Montezuma's Revenge, Gravitar, and Venture) and games with dense rewards (e.g., Beam Rider, Atlantis, and Freeway); two challenging robot control tasks (FetchPickAndPlace and HandManipulateBlock) with sparse rewards and a control task (Walker2D) with dense rewards. In model-based case, we consider two robotic control tasks (Acrobot and Walker2D).

Baselines: In model-free case, we compare Farsighter to four baselines in discrete action environments: DDQN with ϵ-greedy exploration and BDQN, a parametric posterior based exploration, which only considers one-step uncertainty. Moreover, to study the effects of multi-step learning, we also compare Farsighter with 'k-DDQN' which uses ϵ-greedy exploration in each step but considers k steps. We also compare with OB2I, which is the SOTA uncertainty propagation method that use non-parametric posterior based exploration. Similarly, for continuous control tasks, we select four baselines: standard one-step NAF with random exploration, multi-step NAF with random exploration, one-step NAF with Bayesian uncertainty exploration. In model-based case, we compare Farsigter with three baselines: one-step NAF with Bayesian uncertainty exploration, Farsighter-episode that considers whole episode uncertainties using dynamic model and Plan2explore Sekar et al. (2020) that explores by imaging all future steps in the dynamic model. To be fair, we keep the shared parts of the methods to be the same for different exploration methods, e.g. the state representation layers, and the hyper-parameters.

Performance: In model-free cases, Farsighter outperforms DDQN, BDQN and OB2I in 36 out of 49 Atari games. We show parts of the evaluation results in Fig. 3 and Fig. 4. More detailed results (e.g. game scores for 49 Atari games and model-based resutls) are available in the appendix. We run each experiment 10 times with different random seeds and show the average performance. The shaded area is the standard deviation in the Figures.

Figure 3 compares the game scores with the four baselines in Atari Games. We can see Farsighter achieves higher scores substantially. In the notoriously hard exploration game Montezuma's Revenge, Farsighter achieve positive results, while others achieve zero score. The reason is that we initial k=150, which accumulates the uncertainty over k timesteps before performing an update. A higher initial k leads to the agent to explore more in the game and encounter informative state faster. On the other hand, other methods cannot explore enough in the game and most of the reward feedback is zero, thus it is hard to get positive score. In Gravitar and Beam Rider, DDQN and BDQN show comparable performance. BDQN performs a little better since the agent can explore with



(a) Fetch Pick And Place (b) Hand Manipulate Block

Figure 4: The mean success rate for continue robotic tasks

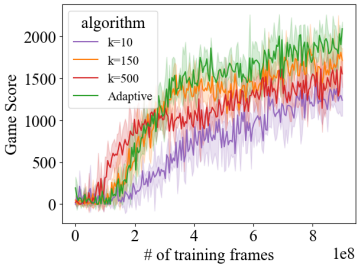


Figure 5: The effects of different number of uncertainty steps in Montezuma's Revenge

one-step uncertainty and k-DDQN cannot improve the performance compared with DDQN, which means k-step learning without uncertainty cannot improve the exploration either. Interestingly, the OB2I increases faster at early and then degenerates. This is because OB2I performs unnecessary exploration which may guide a direction that is unrelated to the environment reward. In comparison, Farsighter performs enough exploration and exploit it efficiently.

Figure 4 shows the performance comparison for continuous robotic tasks. The results show that the multi-step uncertainty exploration also outperforms one-step uncertainty exploration and random exploration in continuous action tasks. In the FetchPickAndPlace task, Farsighter achieves almost 100% success rate and it only takes around 100 million steps. The success rate in HandManipulate-Block is also the best and it takes the least samples for the sample success rate.

Overall, we can conclude that Farsighter is an effective exploration method by considering multi-step uncertainty and it works on general RL tasks. Moreover, as we discussed in Sec 4.4, multi-step uncertainty estimation would not increase the computation and memory complexity compared to one-step uncertainty estimation. More complexity analyses are provided in the appendix.

5.3 Step Size k

Figure 5 shows the impact of k on the performance in Montezuma's Revenge. We can see the performance increases with k initially and then drops, with $k = 150$ achieving the best score. This trend exists for other environments although the optimal step size varies. An interesting observation is that the increased velocity of the scores at the earlier stage is positively proportional to the number of uncertainty steps. This illustrates the importance, in particular in the early stages, of multi-step exploration. The number of uncertainty steps, k, is a trade-off between exploitation and exploration. When k is large, (e.g., k=500), the agent takes more cumulative uncertainty into account, and large uncertainty forces the agent to explore more about the environment, which could be desirable in the early stages, but at the risk of too much exploration and thus difficulties in convergence. This might explain why uncertainty propagation methods (e.g. OB2I, WQL) which accumulate uncertainties for all the remaining steps in an episode are outperformed by our method. On the contrary, when k is small (e.g. k=10), the agent only considers the uncertainty of the next few steps. The uncertainty is easy to vanish and the agent tends to exploit, which is more desirable in later stages. Thus, Farsighter can explicitly balance the bias-variance trade-off by adjusting the number of k. As discussed in Sec. 4.6, we can use an adaptive k. From Fig. 5, we can see the adaptive Farsighter achieves the best result, where the score increases quickly initially and also finishes with the highest value.

6 Conclusion

In this paper, we propose Farsighter, to consider the k-step uncertainty impact and we can explicitly adjust the number of future steps to balance the Q-estimation bias-variance trade-off. Farsighter helps to alleviate the sparse reward and uncertainty vanishing problem. It outperforms SOTA on a wide range of RL tasks with high/low-dimensional states, discrete/continuous actions, and sparse/dense rewards, including hard-to-explore problems such as high-dimensional Atari games and continuous control robotic manipulation tasks.

Reproducibility: We provide the code and appendix with more statement and experiment details (e.g. hyper-parameters, network architecture) at the anonymous link: `https://drive.google.com/file/d/1ejQvBgc11LkW3NcweW82PdwaDAYJEIZc/view?usp=sharing`.

REFERENCES

András Antos, Csaba Szepesvári, and Rémi Munos. Learning near-optimal policies with bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, 71(1):89–129, 2008.

Kamyar Azizzadenesheli, Emma Brunskill, and Animashree Anandkumar. Efficient exploration through bayesian deep q-networks. In *2018 Information Theory and Applications Workshop (ITA)*, pp. 1–9. IEEE, 2018.

Chenjia Bai, Lingxiao Wang, Lei Han, Jianye Hao, Animesh Garg, Peng Liu, and Zhaoran Wang. Principled exploration via optimistic bootstrapping and backward induction. In *International Conference on Machine Learning*, pp. 577–587. PMLR, 2021.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

Nuttapong Chentanez, Andrew G Barto, and Satinder P Singh. Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pp. 1281–1288, 2005.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *International conference on machine learning*, pp. 2829–2838. PMLR, 2016.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. *Advances in neural information processing systems*, 29, 2016.

David Janz, Jiri Hron, Przemysław Mazur, Katja Hofmann, José Miguel Hernández-Lobato, and Sebastian Tschiatschek. Successor uncertainties: exploration and uncertainty in temporal difference learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Igor Kononenko. Bayesian neural networks. *Biological Cybernetics*, 61(5):361–370, 1989.

Michail G Lagoudakis and Ronald Parr. Least-squares policy iteration. *The Journal of Machine Learning Research*, 4:1107–1149, 2003.

Alberto Maria Metelli, Amarildo Likmeta, and Marcello Restelli. Propagating uncertainty in reinforcement learning via wasserstein barycenters. *Advances in Neural Information Processing Systems*, 32, 2019.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PMLR, 2016.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Advances in neural information processing systems*, pp. 4026–4034, 2016a.

Ian Osband, Benjamin Van Roy, and Zheng Wen. Generalization and exploration via randomized value functions. In *International Conference on Machine Learning*, pp. 2377–2386. PMLR, 2016b.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pp. 63–71. Springer, 2003.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897, 2015.

Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*, pp. 8583–8592. PMLR, 2020.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Gerald Tesauro et al. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Tianpei Yang, Hongyao Tang, Chenjia Bai, Jinyi Liu, Jianye Hao, Zhaopeng Meng, and Peng Liu. Exploration in deep reinforcement learning: a comprehensive survey. *arXiv preprint arXiv:2109.06668*, 2021.