2023 IEEE International Conference on Robotics and Biomimetics (ROBIO) | 979-8-3503-2570-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/ROBIO58561.2023.10354785

Uncertainty in Bayesian Reinforcement Learning for Robot Manipulation Tasks with Sparse Rewards

Li Zheng[1], Yanghong Li[1], Yahao Wang[1], Guangrui Bai[1], Haiyang He[1] and Erbao Dong[1]

***Abstract*— This paper aims to explore the application of Bayesian deep reinforcement learning (BDRL) in robot manipulation tasks with sparse rewards, focusing on addressing the uncertainty in complex and sparsely rewarded environments. Conventional deep reinforcement learning (DRL) algorithms still face significant challenges in the context of robot manipulation tasks. To address this issue, this paper proposes a general algorithm framework called BDRL that combines reinforcement learning algorithms with Bayesian networks to quantify the model uncertainty, aleatoric uncertainty in neural networks, and uncertainty in the reward function. The effectiveness and generality of the proposed algorithm are validated through simulation experiments on multiple sets of different sparsely rewarded tasks, employing various advanced DRL algorithms. The research results demonstrate that the DRL algorithm based on the Bayesian network mechanism significantly improves the convergence speed of the algorithms in sparse reward tasks by accurately estimating the model uncertainty.**

I. INTRODUCTION

Reinforcement learning (RL) has emerged as a promising approach for training autonomous agents to perform complex tasks in various domains[1], [2], [3], [4], especially including robotics manipulation tasks such as grasping[5], tool assembly[6] and human-robot interaction[7]. However, RL algorithms often face challenges when dealing with sparse reward scenarios, where the agent receives limited feedback on its actions. In such cases, conventional RL methods struggle to explore the environment effectively and converge to optimal policies.

To address these challenges, recent researches have sought to enhance the accuracy and stability of algorithmic models by leveraging the predictive uncertainty of neural networks[8], [9], [10]. Bayesian Neural Networks (BNNs) offer a probabilistic interpretation of deep learning models by inferring the distribution of model weights[11], thereby providing a means to quantify predictive uncertainty. However, BNNs do not scale well to models with millions of parameters. To address this issue, the Dropout method has been developed as a Bayesian approximation technique[12], which allows obtaining estimates of uncertainty by performing practical evaluations on deep neural networks. This study focuses on incorporating uncertainty estimation into the RL framework by proposing a method called Bayesian deep reinforcement learning (BDRL). BDRL leverages Bayesian neural networks to model the predictions and actions of RL agents, aiming to quantify the associated uncertainties. In this framework, uncertainty is categorized into model uncertainty, aleatoric uncertainty, and reward function uncertainty. Model uncertainty captures uncertainties in model parameters and structure, aleatoric uncertainty captures inherent noise in the data, and reward function uncertainty quantifies the presence of noise or inaccuracy in the reward signal. This paper applies the proposed Bayesian mechanism to a variety of robot manipulation tasks characterized by sparse rewards. The aim is to explore the effectiveness and advantages of integrating Bayesian networks into various state-of-the-art DRL algorithms. In summary, we make the following contributions.

- We propose a deep reinforcement learning algorithm framework based on the Bayesian mechanism to address uncertainty perception.
- We define reward function uncertainty, which complements the aleatoric and model uncertainties, to evaluate the uncertainty related to data incompleteness and inaccurate definition of reward functions.
- To validate the universality of the proposed BDRL framework, we conducted experimental comparisons and validations between the Bayesian-based DRL algorithm and the conventional DRL algorithm in various reward-sparse environments.

II. PRELIMINARIES

This section introduces the background of the research content of this paper, including reinforcement learning, uncertainty and bayesian network.

A. Reinforcement learning

In RL, problems are usually modeled as a Markov decision processes (MDPs): $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state space and action space, respectively. The reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and the state transition probability $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, represents the probability of transitioning between two adjacent states. $\gamma \in [0, 1)$ is the discount factor.

The assembly policy of the agent can be represented as a mapping from states to actions $\pi_\theta : \mathcal{S} \to \mathcal{A}$. At each time step t, the agent selects and executes an assembly action $a_\mathrm{t} \in \mathcal{A}$ based on the current state $s_\mathrm{t} \in \mathcal{S}$. The environment transitions to next state $s_{\mathrm{t}+1} \in \mathcal{S}$ based on the state transition probability $\mathcal{P}_{\pi_\theta}(s_{\mathrm{t}+1} \mid s_\mathrm{t}, a_\mathrm{t})$, while also receiving a scalar reward r_t, which represents a quantified evaluation of the action-state value. The agent improves the learned policy by

[1]CAS Key Laboratory of Mechanical Behavior and Design of Materials, Department of Precision Machinery and Precision Instrumentation, University of Science and Technology of China, 96 Jinzhai Road, 230026, Hefei, Anhui Province, China Correspondence Email: ebdong@ustc.edu.cn

maximizing the sum $R(t)$ of the expected rewards r_{t} for future steps,

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\tau \sim P_{\pi_\theta}}[R(t)] \tag{1}$$

where $R(t) = \sum_{k=0}^{\infty} \gamma^k \cdot r_{\mathrm{t+k}}$. Therefore, the gradient of $J(\boldsymbol{\theta})$ can be written as:

$$\begin{aligned} \nabla_\theta J(\boldsymbol{\theta}) &= \nabla_\theta \mathbb{E}_{\tau \sim P_{\pi_\theta}}[R(t)] \\ &= \mathbb{E}_{\tau \sim P_{\pi_\theta}}\left[\nabla_\theta \log P_{\pi_\theta}(\tau) R(t)\right] \\ &\approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log P_{\pi_\theta}(\tau_i) R(t_i) \end{aligned} \tag{2}$$

where $\tau = \{s_0, a_0, s_1, a_1, \dots\}$ corresponds to a trajectory of each episode, and the subscript i represents the ith trajectory.

The Q-function or state-action value function is defined as $Q^{\pi}(s_{\mathrm{t}}, a_{\mathrm{t}}) = \mathbb{E}[G_{\mathrm{t}} \mid s_{\mathrm{t}}, a_{\mathrm{t}}]$. Let π^* denote an optimal policy i.e. any policy π^* s.t. $Q^{\pi^*}(s,a) > Q^{\pi}(s,a)$ for every $s \in \mathcal{S}$, $a \in \mathcal{A}$ and any policy π. All optimal policies have the same Q-function which is called the optimal Q-function and denoted Q^*. It is show as the following equation:

$$Q^*(s,a) = \mathbb{E}_{s' \sim p(\cdot|s,a)}\left[r(s,a) + \gamma \max_{a' \in \mathcal{A}} Q^*(s',a')\right] \tag{3}$$

Usually, the reward feedback is sparse, represented as follow:

$$r\left(s_{o_{t+1}}, s_{dg_t}, a_t\right) = -\mathbb{L}\left(s_{ag_{t+1}} = s_{dg_t}\right) \tag{4}$$

where $\mathbb{L}()$ is the indicator function, and $s_{ag_{t+1}} \in s_{o_{t+1}}$ is the achieved goal in the next state. This reward function indicates whether the current task was completed.

B. Uncertainty and bayesian neural network

Due to the deterministic nature of the model weights and biases in a deep neural network, it cannot provide probabilistic explanations for the model. When the model encounters data outside the training distribution, the reliability of the predictions decreases significantly. Therefore, using softmax as a probability interpretation for the model's output is highly inappropriate. It is crucial to employ more accurate methods to obtain uncertainty estimates for deep neural network models. BNNs models can be used to obtain uncertainty estimates for deep neural network models. Bayesian probability theory provides a solid mathematical framework for obtaining model uncertainty. It assumes that the model parameters follow a Gaussian prior distribution: $\omega \sim p(\omega)$, given an observed dataset: $X = \{X_1, X_2, \dots, X_{\mathrm{n}}\}, Y = \{Y_1, Y_2, \dots, Y_{\mathrm{n}}\}$, where X represents the observed data and Y represents the corresponding labels, we can use Bayesian inference to compute the posterior distribution of model parameters as follows:

$$p(\boldsymbol{\omega} \mid \boldsymbol{X}, \boldsymbol{Y}) = \frac{p(\boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{\omega}) \times p(\boldsymbol{\omega})}{p(\boldsymbol{Y} \mid \boldsymbol{X})} \tag{5}$$

where $p(\omega \mid X, Y)$ represents the posterior distribution of model weights given the observed data (X, Y), $p(Y \mid X, \omega)$ is the likelihood function, and $p(Y \mid X)$ is the marginal probability. BNNs are neural network models in which the weights and biases are assigned prior distributions. By solving the model and obtaining the posterior distribution, we can capture the uncertainty of the model, as illustrated in Figure1.

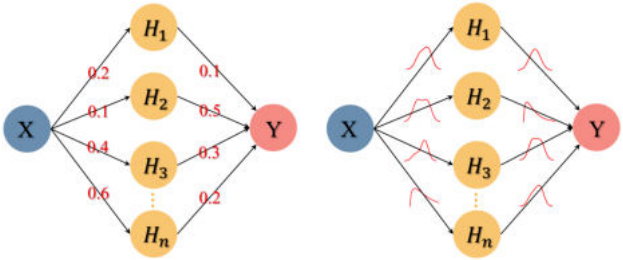


Fig. 1: (a)Ordinary neural network; (b)Bayesian neural network.

III. METHODS

The variables of the formulae used in this article are listed in Table I.

TABLE I: Symbols and notation.

Symbols	Meaning	Symbols	Meaning
S	State inputs	A	Action outputs
s_{i}	Inputs components	a_{i}	Output components
j	State dimension	k	Action dimension
ω	Random weight vector	σ	The noise in the data
θ	Variational parameter	$\mathcal{N}$	Gaussian distribution

A. Uncertainty definition

Quantifying the uncertainty of deep neural network models is crucial for assessing their reliability, robustness, generalization ability, interpretability, and visualizability. In DRL, uncertainty can mainly be attributed to three sources: model uncertainty stemming from model's inherent uncertainty, aleatoric uncertainty arising from the intrinsic noise in the data, and reward function uncertainty form inaccurate definition of reward functions. In a DRL task, the input to the policy network is the state vector S of the current agent, the output is $A = f^{\omega}(s_{\mathrm{i}})$, ω represents the distribution of weight values, and f^{ω} denotes the mapping formed by the entire network model. In the BDRL model, we initialize the model weights and biases to follow a normal distribution $\omega \sim \mathcal{N}(0, \sigma^2)$. Assuming the inputs and outputs of the model are denoted as $S = \{s_1, s_2, \dots, s_{\mathrm{j}}\}$ and $A = \{a_1, a_2, \dots, a_{\mathrm{k}}\}$ respectively. The posterior probability of the model can be computed using the standard Bayesian deep learning approach as shown in Equation 5. The model likelihood, denoted as $p(A \mid S, \omega)$, is defined in Equation 6, where the model's output is normalized using the $Tanh$ function.

$$p(A \mid S, \omega) = Tanh\left(\frac{1}{\delta} f^{\omega}(s_{\mathrm{i}})\right) \tag{6}$$

where δ represents the inherent noise in the data.

Since the true posterior distribution cannot be computed, we approximate the posterior distribution $q_\theta(\omega)$ using variational inference[13], by minimizing the Kullback-Leibler (KL) divergence between the assumed weight

distribution, which follows a parameterized θ, and the true posterior distribution. This can be formulated as follows:$-\int q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) \log p(\boldsymbol{A} \mid \boldsymbol{S}, \boldsymbol{\omega}) d\boldsymbol{\omega} + KL\left[q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) \| p(\boldsymbol{\omega})\right]$. By minimizing this objective, we can obtain the optimal approximation of the posterior distribution, denoted as $q_\theta^*(\omega)$.

We incorporate Dropout as a variational Bayesian approximation by adding Dropout layers before the weight layers and treating the dropout probabilities as random variables following a certain distribution. We sample from the approximate posterior using Dropout, and the predicted probabilities $\hat{\boldsymbol{a}}_i$ of the model are computed as follows:

$$\begin{aligned} \hat{\boldsymbol{a}}_i &= \boldsymbol{p}\left(a \mid \boldsymbol{s}_i, \boldsymbol{s}, \boldsymbol{a}\right) \\ &\approx \int p\left(a \mid \boldsymbol{s}_i, \boldsymbol{\omega}\right) \times q_\theta^*(\boldsymbol{\omega}) d\boldsymbol{\omega} \\ &\approx \frac{1}{T} \sum_{t=1}^{T} p\left(a \mid s_i, \hat{\boldsymbol{\omega}}_t\right) \\ &= \frac{1}{T} \sum_{t=1}^{T} \hat{a}_{i,t} \end{aligned} \tag{7}$$

where $\hat{\boldsymbol{a}}i$ represents the result of the t-th forward prediction, and $\hat{\boldsymbol{\omega}}t \sim q^*\theta(\omega)$.

*1) **Model Uncertainty**:* The model uncertainty primarily arises from the imperfections of the model. We quantify the model uncertainty by measuring the average Euclidean distance between the results of multiple Monte Carlo samples and the final output distribution of the model. The model uncertainty is defined as follows:

$$\begin{aligned} \mathbb{U}_{\text{model}}^{(i)} &= \int \left\{\boldsymbol{p}\left(a \mid s_i, \boldsymbol{\omega}\right) - \mathbb{E}\left[\boldsymbol{p}\left(a \mid s_i\right)\right]\right\}^{\otimes 2} q_\theta^*(\boldsymbol{\omega}) d\boldsymbol{\omega} \\ &\approx \frac{1}{T} \sum_{t=1}^{T} \left(\boldsymbol{p}\left(a \mid s_i, \hat{\omega}_t\right) - \left[\frac{1}{T} \sum_{t=1}^{T} \boldsymbol{p}\left(a \mid s_i, \hat{\omega}_t\right)\right] \right)^{\otimes 2} \\ &= \frac{1}{T} \sum_{t=1}^{T} D^2 \left(\hat{a}_{i,t}, \frac{1}{T} \sum_{t=1}^{T} \hat{a}_{i,t} \right) \end{aligned} \tag{8}$$

where $D(\boldsymbol{z}_1, \boldsymbol{z}_2) = \sqrt{\sum_{k=1}^{K} (z_1^k - z_2^k)^2}$, $\boldsymbol{z}_1$, and $\boldsymbol{z}_2$ represents the probability distribution vectors. Additionally, $\boldsymbol{z}^{\bigotimes 2} = \boldsymbol{z}^T \cdot \boldsymbol{z}$, $\mathbb{E}[\boldsymbol{p}(|\boldsymbol{s}_i)] \approx \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{p}(|\boldsymbol{s}_i, \hat{\boldsymbol{\omega}}_t)$, where $\hat{\boldsymbol{\omega}}_t$ is the weight vector obtained from the t-th sample drawn from the approximate posterior distribution.

*2) **Aleatoric uncertainty**:* The aleatoric uncertainty measures the interference on the predicted results caused by the inherent noise in the model. We obtain the model's aleatoric uncertainty output by placing a distribution on the model's output. The model's output is divided into two parts: the first part is the model's predicted result $f^{\omega}(\boldsymbol{s}i)$, and the second part is the aleatoric uncertainty σi^2 associated with the network output $\boldsymbol{s}_i$. By performing multiple Monte Carlo samples, we obtain multiple sets of outputs, and the average of these sets is taken as the final aleatoric uncertainty. The expression can be formulated as follows:

$$\mathbb{U}_{aleatoric}^{(i)} = \int (\sigma_i^2(\boldsymbol{\omega})) q_{\boldsymbol{\theta}}^*(\boldsymbol{\omega}) d\boldsymbol{\omega} \approx \frac{1}{T} \sum_{t=1}^{T} (\sigma_{i,t})^2 \tag{9}$$

where $\sigma_{i,t}$ refers to the observed noise obtained from the t-th sampling of the input $\boldsymbol{a}_i$.

*3) **Reward Functional uncertainty**:* Reward function uncertainty measures the uncertainty arising from the noise or inaccuracy in the reward signal. We quantify this uncertainty by calculating the KL divergence between the actual reward distribution $R(a \mid s)$ and the ideal expected distribution $E(a \mid s)$ under state s. The uncertainty is computed as follows:

$$\mathbb{U}_{reward}^{(i)} = D_{KL}(E \parallel R) = \sum_{i=1}^{n} E(i) log \frac{E(i)}{R(i)} \tag{10}$$

The expected reward distribution describes the sum of expected rewards obtained by the agent for all possible actions taken under state s. It can be represented as:

$$E(a \mid s) = \mathbb{E}_{a \sim \mathcal{N}(0, \sigma^2)}[r(a, s)] \tag{11}$$

B. Bayesian Deep Reinforcement Learning

The objective of our study is to accelerate the training process and improve the performance of conventional DRL algorithms in complex environments by leveraging uncertainty estimation through a Bayesian network. The proposed Bayesian deep reinforcement learning framework is outlined in Figure 2. Within our framework, a neural network is

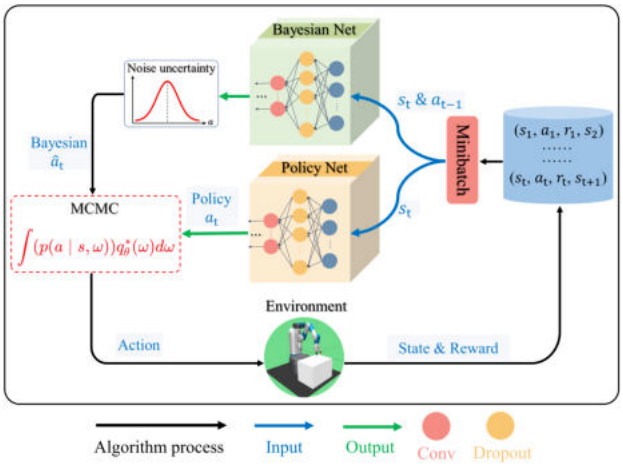


Fig. 2: Illustration of the proposed Bayesian mechanism for DRL algorithms.

modeled as a policy network, which takes the state as input and produces an action as output. The optimization objective of the policy network is to maximize the cumulative reward of the agent. Additionally, a Bayesian network is constructed to assist in the optimization of the policy network by evaluating its performance. The Bayesian network takes the state and the previous action as inputs and produces an action and the cumulative reward of the policy in a given state as outputs. The optimization objective of the Bayesian network is to minimize the KL divergence between the output accumulated reward and the expected accumulated reward. As an action selector, the Markov Chain Monte Carlo (MCMC) sampler is employed to choose an action

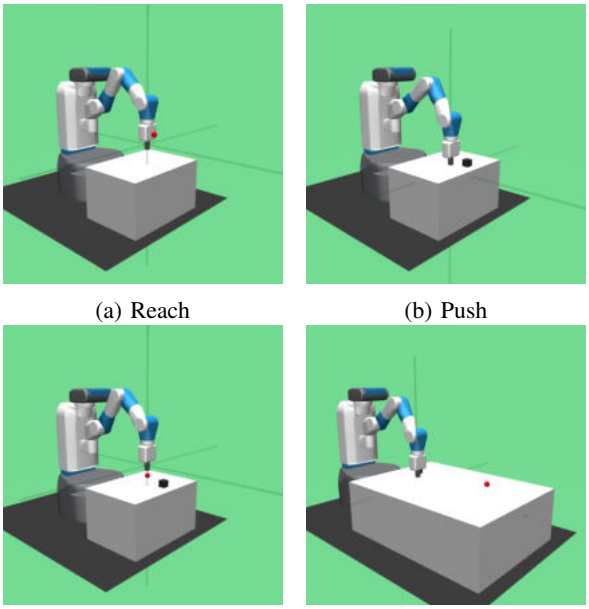
from either the Bayesian network or the policy network. Subsequently, the agent interacts with the environment by executing the selected action, and the data collected from these interactions are utilized to train both the policy network and the Bayesian network.

IV. EXPERIMENTS AND RESULTS

In this section, we evaluate the Bayesian deep reinforcement learning framework in four different robot manipulator tasks. Additionally, we employ various state-of-the-art deep reinforcement learning algorithms to assess whether the BDRL approach can effectively accelerate the training process of different DRL algorithms and address the issue of sparse rewards in complex environments. Our objective is to answer the following questions: a) Can BDRL effectively accelerate the training process of different DRL algorithms? b) Is BDRL applicable to various reinforcement learning-based robot manipulator tasks? c) Can BDRL effectively address the problems of sparse rewards and ineffective exploration? d) Does BDRL contribute to improving the stability and robustness of the algorithms?

A. Experimental Configuration

The four environments[14]:Reach, Push, Pick and Place, Slide, are based on the 7-DoF Fetch Mobile Manipulator arm, with a two-fingered parallel gripper attached to it, shown in Figure 3. All four environments share the characteristics of sparse rewards and complex motion control, with the Slide task being the most challenging among them. The details are



(a) Reach (b) Push

(c) PickAndPlace (d) Slide

Fig. 3: The four environments for manipulators operation tasks.

elaborated as follows:
Reach: Fetch has to move its end-effector to the desired goal position.
Push: Fetch has to move a box by pushing it until it reaches a desired goal position.
PickAndPlace: Fetch has to pick up a box from a table using its gripper and move it to a desired goal above the table.
Slide: Fetch has to hit a puck across a long table such that it slides and comes to rest on the desired goal.
The reward functions are still the most common binary rewards described in Equation 4.

B. Training the policy with Bayes

To address the aforementioned questions, we applied the proposed Bayesian network mechanism to four state-of-the-art DRL algorithms, namely DDPG [15], TD3 [16], SAC [17], PPO[18]. In order to demonstrate the generality of the Bayesian network mechanism, we selected four tasks with sparse rewards, and each task was trained 3000 episodes with differdent DRL algorithms. The experimental results were compared against the original DRL methods without the Bayesian network mechanism. The results of the comparative experiments are shown in Figure 4.

For question (a): according to the experimental results, it is observed that the DRL algorithms incorporating the Bayesian network mechanism exhibit faster convergence speed and achieve higher cumulative rewards in single-task scenarios. This suggests that the actions learned by the Bayesian network mechanism are more precise. More specifically, the Bayesian network mechanism enhances the robustness and generalization capability of the model by quantifying the model uncertainty and noise uncertainty in the neural network. Moreover, by capturing the uncertainty in rewards, a more accurate evaluation of the value of the network's policy in the current state can be obtained, thereby increasing the probability of the agent achieving high rewards in the environment and significantly accelerating the convergence process.

For question (b): from Figure 4, it can be observed that the DRL algorithms incorporating the Bayesian network mechanism also exhibit faster convergence speed when performing different tasks. Specifically, for DDPG and TD3, the convergence speed is generally slower in tasks with sparse rewards. The main reason for this is that DDPG and TD3 use single-step rewards to estimate Q-values, which leads to biased estimation. Additionally, in sparse reward tasks, when the agent has not reached the goal state, it receives a negative reward signal at each step. As a result, the majority of experiences in the replay buffer are negative, leading to inaccurate Q-value estimation. In contrast, SAC and PPO use multi-step rewards for evaluation, which provides unbiased estimation. Therefore, the performance of SAC and PPO gradually improves until convergence in sparse reward tasks. Furthermore, under the mechanism of the Bayesian network, the replay buffer contains a large number of high-reward state-action pairs. DDPG and TD3 with the Bayesian network mechanism demonstrate accelerated performance improvement during the training process, as shown in Figure a1-a4 and b1-b4. Additionally, the Bayesian network aids the policy network in escaping local optima and approaching

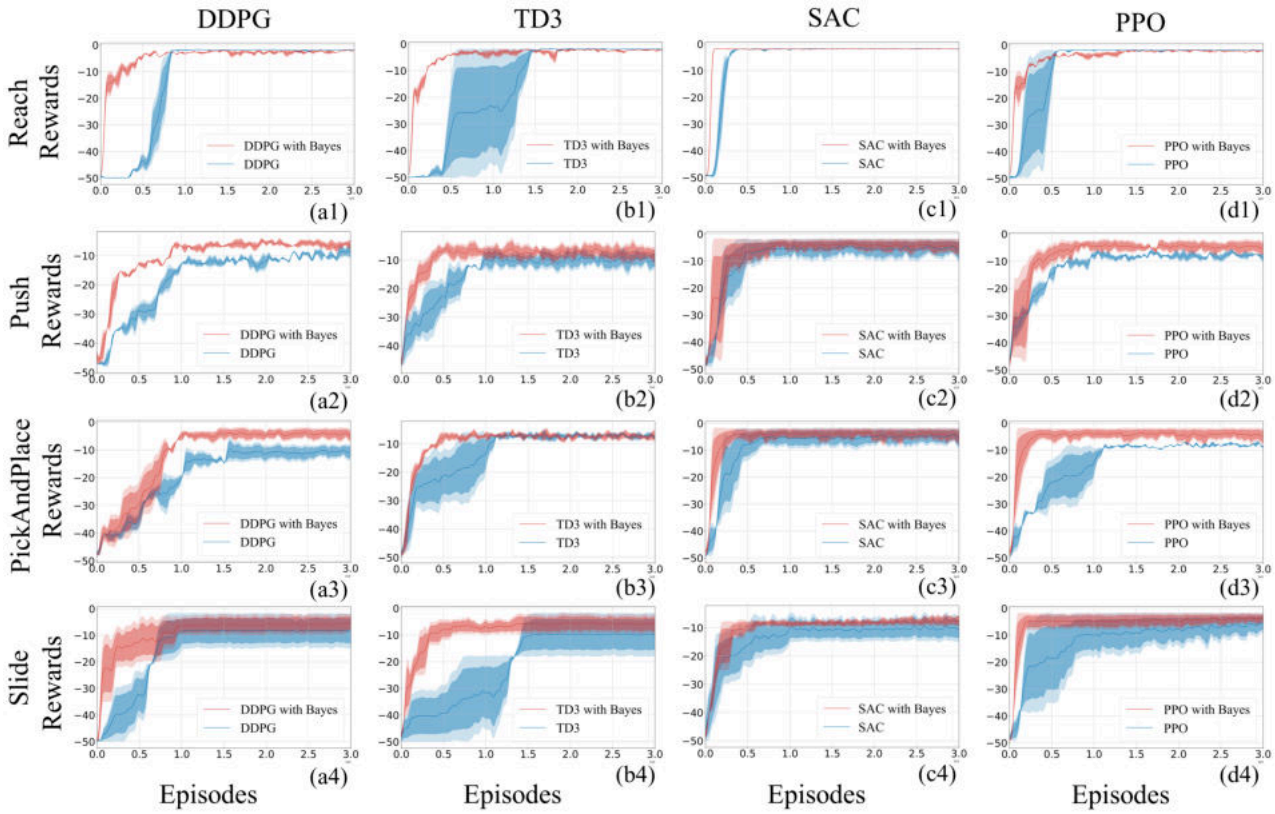


Fig. 4: Average rewards per episodes of the proposed Bayes mechanism with four DRL algorithms versus the DRL algorithms in four different sparse-reward environment.

TABLE II: Average episode rewards comparison of four DRL algorithms with Bayes versus the DRL algorithms in four sparse-reward tasks.

Task	DDPG	DDPG with Bayes	TD3	TD3 with Bayes	SAC	SAC with Bayes	PPO	PPO with Bayes
Reach	-24.16±5.21	-20.05±4.20	-23.93±9.08	-21.80±4.45	-7.49±3.83	-5.84±1.04	-15.49±4.83	-10.49±3.23
Push	-26.13±5.46	-24.05±4.87	-26.25±8.18	-23.80±5.32	-11.86±12.64	-9.30±9.14	-19.94±6.27	-12.34±4.12
PickAndPlace	-27.32±6.79	-23.64±5.01	-25.20±8.12	-23.61±5.98	-13.86±9.97	-10.45±8.63	-23.55±8.73	-17.54±5.93
Slide	-29.16±9.78	-26.01±7.32	-30.13.93±14.59	-23.17±8.66	-17.62±6.15	-14.67±5.24	-21.09±8.91	-9.82±5.57

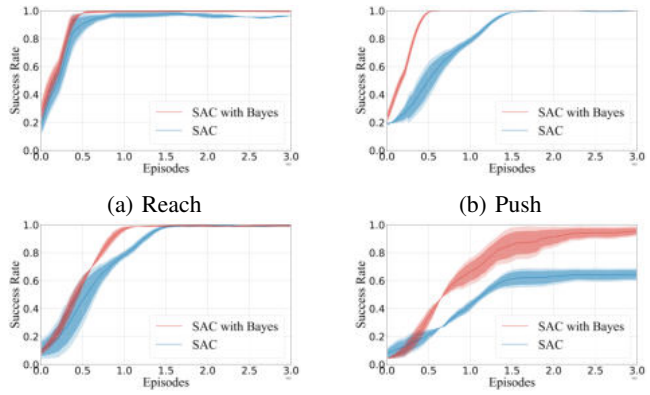
the optimal solution of the task through better exploration strategies.

For questions (c) and (d), based on the results presented in Table II, we can conclude that the Bayesian network mechanism performs well in addressing the four reward-sparse tasks across different DRL algorithms. Furthermore, the algorithms incorporating the Bayesian network mechanism generally exhibit smaller standard deviations in most cases. This indicates that by leveraging the neural network uncertainty and reward uncertainty through the Bayesian network mechanism, the performance of DRL algorithms in sparse-reward tasks can be significantly improved, accelerating the training process through enhanced exploration. Further analysis of the experimental results reveals that the Bayesian network-based algorithms are capable of accurately estimating model uncertainty and adjusting policies accordingly to adapt to different environments. This mechanism enables the agents to adopt a more conservative strategy when facing environmental uncertainties, mitigating risks and avoiding failures. In contrast, conventional DRL algorithms often rely too heavily on model predictions with excessive confidence, resulting in degraded performance and increased instability.

C. Testing the policy

To validate the stability of the proposed Bayesian network mechanism, we conducted comparative evaluation experiments between the DRL algorithm based on the Bayesian network mechanism and the original DRL algorithm, using the SAC algorithm with the best training performance. We executed approximately 300 episodes for each of the four tasks according to the learned policy, and the success rates for the four tasks are shown in Figure 5. The results demonstrate that the Bayesian network-based algorithm outperforms the, conventional DRL algorithm in terms of performance and stability. Specifically, for relatively simple

tasks such as Reach, Push, and PickAndPlace, the algorithm based on the Bayesian network mechanism not only ensures convergence but also improves algorithm stability. However, for the relatively complex task of Slide, the conventional DRL algorithm struggles to achieve satisfactory performance, resulting in a significant decrease in the agent's success rate. In contrast, the DRL algorithm based on the Bayesian network mechanism, by capturing the algorithm's uncertainty, adapts better to the complexity and uncertainty of the task, leading to a significant improvement in the agent's success rate.



(a) Reach (b) Push

(c) PickAndPlace (d) Slide

Fig. 5: Comparison of success rates between SAC with Bayesian and SAC in four different sparse reward environments.

V. CONCLUSIONS

The proposed DRL algorithm based on the Bayesian network mechanism demonstrates superior performance compared to conventional DRL algorithms when facing different reward-sparse tasks. Through a combination of comparative experiments on four reward-sparse environments and evaluation validation experiments, we have verified the algorithm's advantages in handling dynamic environmental changes and unknown factors. In these reward-sparse tasks, conventional DRL algorithms often struggle to effectively utilize limited reward signals to learn and infer the environment's state. However, the BDRL algorithm accurately estimates model uncertainty and incorporates uncertainty factors into the decision-making process, allowing for a better understanding of the complexity and uncertainty of the environment and adjusting the agent's decision-making strategy accordingly. The results of this study provide an effective approach to address reward sparsity. The BDRL algorithm not only exhibits excellent performance in experiments but also holds promising potential for wide-ranging applications. Future work will explore the application of this approach to various domains involving reward-sparse tasks, such as robot control, autonomous driving, and gaming, aiming to provide more reliable and robust solutions for decision-making in complex and uncertain environments.

ACKNOWLEDGMENT

This work was supported by the National Key R&D Program of China (Grant number:2018YFB1307400).

REFERENCES

[1] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[2] W. Shang, Y. Yu, Q. Li, Z. Qin, Y. Meng, and J. Ye, "Environment reconstruction with hidden confounders for reinforcement learning based recommendation," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 566–576.

[3] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2021.

[4] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 3, pp. 653–664, 2016.

[5] D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman, "Mt-opt: Continuous multi-task robotic reinforcement learning at scale," *arXiv preprint arXiv:2104.08212*, 2021.

[6] S. Kozlovsky, E. Newman, and M. Zacksenhouse, "Reinforcement learning of impedance policies for peg-in-hole tasks: Role of asymmetric matrices," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 898–10 905, 2022.

[7] L. Zheng, Y. Wang, R. Yang, S. Wu, R. Guo, and E. Dong, "An efficiently convergent deep reinforcement learning-based trajectory planning method for manipulators in dynamic environments," *Journal of Intelligent & Robotic Systems*, vol. 107, no. 4, p. 50, 2023.

[8] F. Ramos, R. C. Possas, and D. Fox, "Bayessim: adaptive domain randomization via probabilistic inference for robotics simulators," *arXiv preprint arXiv:1906.01728*, 2019.

[9] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" *Advances in neural information processing systems*, vol. 30, 2017.

[10] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *International conference on machine learning*. PMLR, 2015, pp. 1613–1622.

[11] N. Tishby, E. Levin, and S. A. Solla, "Consistent inference of probabilities in layered networks: Predictions and generalization," in *International Joint Conference on Neural Networks*, vol. 2. IEEE New York, 1989, pp. 403–409.

[12] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[13] G. E. Hinton and D. Van Camp, "Keeping the neural networks simple by minimizing the description length of the weights," in *Proceedings of the sixth annual conference on Computational learning theory*, 1993, pp. 5–13.

[14] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder *et al.*, "Multi-goal reinforcement learning: Challenging robotics environments and request for research," *arXiv preprint arXiv:1802.09464*, 2018.

[15] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[16] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.

[17] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.