Hindawi
Computational Intelligence and Neuroscience
Volume 2021, Article ID 7588221, 16 pages
https://doi.org/10.1155/2021/7588221

Research Article

Efficient Reinforcement Learning from Demonstration via Bayesian Network-Based Knowledge Extraction

Yichuan Zhang, Yixing Lan, Qiang Fang, Xin Xu, Junxiang Li, and Yujun Zeng

College of Intelligence Science and Technology, National University of Defense Technology, Changsha, China

Correspondence should be addressed to Yujun Zeng; yujunzeng@sina.cn

Received 29 July 2021; Accepted 19 August 2021; Published 24 September 2021

Academic Editor: Ahmed Mostafa Khalil

Reinforcement learning from demonstration (RLfD) is considered to be a promising approach to improve reinforcement learning (RL) by leveraging expert demonstrations as the additional decision-making guidance. However, most existing RLfD methods only regard demonstrations as low-level knowledge instances under a certain task. Demonstrations are generally used to either provide additional rewards or pretrain the neural network-based RL policy in a supervised manner, usually resulting in poor generalization capability and weak robustness performance. Considering that human knowledge is not only interpretable but also suitable for generalization, we propose to exploit the potential of demonstrations by extracting knowledge from them via Bayesian networks and develop a novel RLfD method called Reinforcement Learning from demonstration via Bayesian Network-based Knowledge (RLBNK). The proposed RLBNK method takes advantage of node influence with the Wasserstein distance metric (NIW) algorithm to obtain abstract concepts from demonstrations and then a Bayesian network conducts knowledge learning and inference based on the abstract data set, which will yield the coarse policy with corresponding confidence. Once the coarse policy's confidence is low, another RL-based refine module will further optimize and fine-tune the policy to form a (near) optimal hybrid policy. Experimental results show that the proposed RLBNK method improves the learning efficiency of corresponding baseline RL algorithms under both normal and sparse reward settings. Furthermore, we demonstrate that our RLBNK method delivers better generalization capability and robustness than baseline methods.

1. Introduction

Recent research on reinforcement learning (RL) has made impressive achievements in various domains, including video gaming [1], stock trading [2], and recommendation systems [3]. However, the resource-exhausting training seriously hinders the deployment of RL in real-world scenarios. One of the most important reasons for this issue is that RL agents have no background knowledge and have to learn from scratch, which is neither efficient nor realistic. In contrast, during the human learning process, we expect to learn new tasks by watching demonstrations first, and this inspires the research on reinforcement learning from demonstration (RLfD) [4], which has been proved to be promising in robot grasping [5] and unmanned vehicle driving [6], etc.

However, most previous RLfD methods do not take full advantage of expert demonstrations, limited by treating them as the accurate behavioral templates without providing insight into the reasons of performing such actions. Demonstrations in these RLfD methods are regarded as the low-level representation of human knowledge, which restrains their generalization capability [7]. Moreover, neural network-based RLfD methods aforementioned have limited interpretability, and they also lack the ability of acting robustly to the observation disturbance.

Considering that obtaining expert demonstrations is costly, it is essential to explore how these demonstrations can be used effectively. Therefore, a superior RLfD method should be able to extract knowledge from demonstrations that not only improves the algorithm performance for the same task but also provides explanations of the demonstrator's actions, which facilitates the generalization of the final learned behavioral policy. Here, following the definition proposed in [8], we treat "knowledge" as validated information about the relationships between entities in a certain context and the theoretical definition will be introduced in Section 4.2. Although such knowledge is generally efficient and concise, it is usually uncertain, coarse, and difficult to be expressed or quantified, which indicates that it needs to be further fine-tuned and adjusted to fully accomplish the target task.

As a probabilistic graphical model, Bayesian networks [9] can be used as an appropriate pattern to exploit task-agnostic knowledge from demonstrations since they have multiple advantages. Firstly, as a kind of probabilistic model, Bayesian networks can learn and represent uncertain and coarse knowledge to accomplish probabilistic reasoning. Besides, Bayesian networks have directed graph structures in which the nodes represent real-world observations and actions and the weights between nodes are conditional probability values used to quantify causal relationships between nodes. Thus, Bayesian networks are easy to be interpreted, which provides transparent insight into the extracted knowledge. Moreover, Bayesian networks can provide confidence in decision-making process compared to commonly used methods.

Inspired by the aforementioned ideas, we propose a novel RLfD method called Reinforcement Learning from demonstration via Bayesian Network-based Knowledge (RLBNK) that extracts probabilistic knowledge from expert demonstrations via Bayesian networks and combines the knowledge with RL. The RLBNK method aims to learn a hybrid policy that consists of a fixed knowledge module represented by a Bayesian network and a trainable refine module represented by a neural network, where the refine module undertakes the role of refining the probabilistic coarse knowledge represented by the Bayesian network. By leveraging Bayesian networks as the knowledge representation pattern, the agent can quantify the uncertainty of the prior knowledge extracted from demonstrations, which guides the employment of the probabilistic knowledge. More specifically, we propose two variant RLBNK methods called RLBNK-concat and RLBNK-switch. For RLBNK-concat, the agent concatenates the decision confidence vector provided by the Bayesian network to the current state vector as input and optimizes the whole policy by RL. In this method, the decision confidence vector implicitly provides instruction to the agent. As for RLBNK-switch, it divides the state space according to the decision confidence vector provided by the pretrained Bayesian network knowledge module: if the decision confidence is high, the decision will be made by the Bayesian network; otherwise, the decision will be made by the neural network-based refinement module. Note that for both variants, the knowledge module represented by the Bayesian network is fixed during the RL process. Simulation results illustrate that our RLBNK outperforms the well-established baselines in terms of data efficiency, generalization capability, and robustness.

In summary, the main contributions of this paper are threefold:

(1) An influence-based state abstraction algorithm NIW is proposed to obtain conceptional abstract states from original expert demonstrations. And Bayesian networks then extract probabilistic coarse knowledge from these abstract demonstrations.

(2) A novel RLfD method called RLBNK is proposed, which composes of a Bayesian network that represents probabilistic coarse knowledge and a neural network-based refine module that refines the prior knowledge. And the advantages of RLBNK are also analysed and discussed.

(3) Extensive experiments are conducted to verify the effectiveness of the RLBNK method. The results show that the RLBNK method can achieve better performance in data efficiency, generalization capability, and robustness than the baseline methods.

The remainder of this paper is structured as follows. Section 2 and Section 3 introduce the related works and preliminaries of this paper. The methodology of the RLBNK and the corresponding analysis and discussion are presented in Section 4. Finally, the experimental results are illustrated and analysed in Section 5. Section 6 concludes this paper and envisions the future work.

2. Related Work

2.1. Reinforcement Learning from Demonstration. Reinforcement learning from demonstration (RLfD) is considered as an important branch of learning from demonstration (LfD) method that combines demonstrations with conventional RL to improve the sample efficiency [10] in the training process. Existing RLfD methods are basically rooted in the following three ideas: (1) policy pretraining; (2) reward shaping; (3) providing auxiliary loss.

Policy pretraining [6] is the most commonly used RLfD method in practice. It pretrains the RL policy with demonstrations in a supervised manner via behavior cloning [11], then proceeding with regular RL. The typical work following this idea is the AlphaGo algorithm [12]. However, this approach cannot guarantee the exploration quality during the proceeding policy optimization process, which usually results in “catastrophic forgetting” [13]. Moreover, neural networks are often apt to overfit the demonstrations, which impedes the generalization of the pretrained policy.

Reward shaping aims to instruct the agent’s learning by constructing additional reward signals from expert demonstrations [14, 15]. With additional rewards, RL agents can learn more effectively by obtaining heuristic feedback from both the environment and the introduced rewards. For example, the soft Q imitation learning (SQIL) algorithm [15] stores demonstrations in the replay buffer and assigns a positive reward to them. The study [14] trains a supervised neural network from demonstration to act as a shaping function. However, this idea remains in the tendency of implicitly replicating the expert’s action by encouraging the agent to explore the state space that is covered by the demonstrations.

Providing extra loss terms derived from demonstrations for RL policy function or value function optimization is the third mainstream idea of RLfD. For instance, deep Q-learning from demonstration (DQfD) [16] introduces demonstrations into deep Q-network (DQN) [1] by storing demonstration data into the experience replay buffer to pretrain the Q-network with different loss terms. Then, in the RL process, a prioritized sampling mechanism is employed to select experience data from the replay buffer for Q-network optimization. Likewise, the deep deterministic policy gradient from demonstration (DDPGfD) algorithm [5] inherits this idea and takes deep deterministic policy gradient (DDPG) [17] as the basic algorithm to extend DQfD to robot control tasks with continuous actions. Similar to the reward shaping idea, this approach also aims to encourage the agent to copy the expert's actions by constraining the objective of the optimization.

2.2. Imitation Learning. Imitation learning (IL) also utilizes demonstrations to acquire expert-like policies, and it can be broadly classified into behavioral cloning (BC) and inverse reinforcement learning (IRL). BC [11, 12, 18] is the most common imitation learning paradigm as the expert policy is extracted through supervised learning. However, the policies learned via BC suffer from the compounding error caused by covariate shift [19] in sequential decision-making tasks. Thus, the agent may easily drift away from the demonstrated states. The other IL paradigm is IRL, which tries to recover the reward function of the task by regarding the expert demonstrations are optimal and then learns policies within the RL framework. Thus, this IRL method can avoid compounding error occurs in BC. Combining the idea of generative adversarial networks (GANs) [20] and IRL, the generative adversarial imitation learning (GAIL) [21] method leverages adversarial training to learn the policy from demonstrations directly.

However, it is important to note that even though IL and RLfD are similar, there are fundamental differences between them. RLfD methods still assume access to the reward feedbacks from the environment even though they have the assistance from expert demonstrations, while IL methods do not rely on any reward signal [11, 18] or it constructs the reward function from demonstrations itself [21].

2.3. Knowledge Representation and Integration. Various typical patterns have been explored to represent prior knowledge, such as fuzzy methods [22, 23], rules [24–26], decision trees [27, 28], neural networks [11], and graphs [29, 30]. The advantage of fuzzy methods and rules is that they are naturally interpretable. However, it requires considerable human efforts to manually define the forms of rules and they are limited to represent complex relationships. In contrast, neural networks have powerful representation ability, but the lack of interpretability impedes their adoption. Graphs and decision trees are ideal tools for interpretable knowledge extraction and representation which can automatically extract knowledge from data. Compared to trees, Bayesian networks provide a more concise probabilistic representation as graph models, which are more in line with the human form of learning and reasoning.

There is also some research on the integration of knowledge into RL in different forms. The knowledge guided policy network (KoGuN) method [26] employs fuzzy rules as the knowledge controller. Fuzzy rules are difficult to extract knowledge from the data, and the membership function must be defined manually. Compared with fuzzy rules, Bayesian networks can extract the probabilistic knowledge with minimal human efforts. The requesting confidence-moderated policy advice (RCMP) algorithm [31] also utilizes uncertainty to guide the RL, where the uncertainty used in this algorithm is obtained by computing the variance of multiple Q-value vectors provided by a multiheaded Q-network. Then, the RCMP algorithm requires action advices from the online expert when it has high decision uncertainty. Therefore, this algorithm requires continuous instructions from an online expert.

3. Preliminary

3.1. Reinforcement Learning. RL aims to solve a sequential decision-making problem, where an RL agent optimizes its policy by interacting with the environment following a Markov decision process (MDP) [32]. A standard MDP $\mathcal{M}$ is defined by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma \rangle$. Particularly, $\mathcal{S}$ and $\mathcal{A}$ are the state space and action space with sizes are $|\mathcal{S}|$ and $|\mathcal{A}|$, respectively; $\mathcal{R}$ represents the reward distribution function, with $r_t = r(s_t, a_t)$ is the immediate reward for taking action a_t in state s_t at timestep t; $\mathcal{P}$ denotes the transition probability function, with $\Pr(s_{t+1}|s_t, a_t)$ indicates the probability of transitioning from s_t to s_{t+1} upon action a_t; $\gamma \in (0, 1]$ denotes the discount factor.

As shown in Figure 1, given a policy π, the RL agent chooses an action a_t according to $\pi(s_t)$ and then transits to the next state s_{t+1} following $Pr(s_{t+1}|s_t, a_t)$ and receives an instant reward r_t. We define $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$ as the total discounted reward at s_t with discounted factor γ. The objective of an RL agent is to obtain the (near) optimal policy π^* that maximizes the expectation of R_t. Assuming that the policy network is parameterized by ϕ, the value function $V^{\pi_\phi}(s)$ is usually used to evaluate the policy π_ϕ, where $V^{\pi_\phi}(s)$ can be defined as

$$V^{\pi_\phi}(s) = \mathbb{E}^{\pi_\phi}\left[R_t | s_t = s\right], \tag{1}$$

and the action value function $Q^{\pi_\phi}(s, a)$ is defined as

$$Q^{\pi_\phi}(s, a) = \mathbb{E}^{\pi_\phi}\left[R_t | s_t = s, a_t = a\right], \tag{2}$$

where $\mathbb{E}^{\pi_\phi}[\cdot]$ denotes the expectation with respect to π_ϕ.

The policy-based RL methods update the policy parameter ϕ via gradient ascent given by

$$\phi \longleftarrow \phi + \alpha \nabla J(\phi), \tag{3}$$

where α is the learning rate and $J(\phi)$ is the total expected reward that can be estimated by

$$\nabla J(\phi) = \mathbb{E}_{s\sim\mathcal{S}, a\sim\mathcal{A}}\left[\nabla_\phi \log \pi_\phi(a|s) A^{\pi_\phi}(s, a)\right]. \tag{4}$$

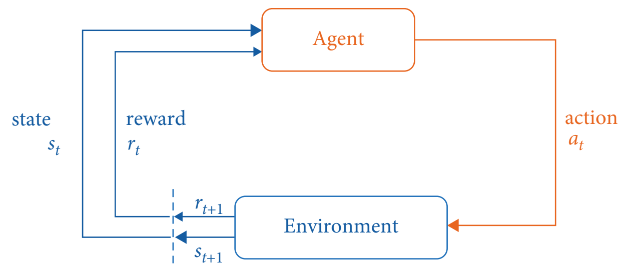

Figure 1: The standard reinforcement learning setup.

Subtracting Q^{π_ϕ} by V^{π_ϕ} gives the advantage function A^{π_ϕ} used in equation (4):

$$A^{\pi_\phi}(s,a) = Q^{\pi_\phi}(s,a) - V^{\pi_\phi}(s), \tag{5}$$

where $A^{\pi_\phi}(s,a)$ reflects the expected additional reward that the agent will receive after taking action a at state s.

To evaluate the generalization capability of demonstrations [7], we should firstly define different MDPs within the RLfD paradigm, where the source MDP $\mathcal{M}_s$: $\langle \mathcal{S}_s, \mathcal{A}_s, \mathcal{R}_s, \mathcal{P}_s, \gamma\rangle$ is used to collect the expert demonstrations and $\mathcal{M}_t$: $\langle \mathcal{S}_t, \mathcal{A}_t, \mathcal{R}_t, \mathcal{P}_t, \gamma\rangle$ is the target MDP that needs to be solved. In RLfD, an RL agent interacts with the environment following the target MDP $\mathcal{M}_t$ and is also provided with expert demonstrations generated by the expert policy π_E from source MDP $\mathcal{M}_s$. In RL, the generalization settings can be various, where $\mathcal{M}_s$ and $\mathcal{M}_t$ can differ by state space $\mathcal{S}$, action space $\mathcal{A}$, reward function $\mathcal{R}$, or system dynamics $\mathcal{P}$.

As a generalization of the standard MDP, the partial observable Markov decision process (POMDP) [33] extends MDP to the partial observable environment settings. In POMDP, the agent only receives an observation o_t with distribution $p(o_t|s_t)$ at each time step t. Similar to the standard MDP, the aim of the POMDP is to maximize the expected total reward that the RL agent receives. Moreover, the other core issue in POMDP is to improve the robustness of the trained policy to the stochastic disturbance of the environment.

3.2. Bayesian Networks

3.2. Bayesian Networks. Bayesian networks [9] belong to probabilistic graphical models (PGMs) that can be defined as $\langle \mathcal{G}, \mathcal{O}\rangle$ where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is the directed acyclic graph, with $\mathcal{V}$ is the set of nodes (variables) and edges $\mathcal{E}$, and $\mathcal{O}$ is the probability function. Depending on whether the variables are discrete or continuous, Bayesian networks can be classified into discrete Bayesian networks and Gaussian Bayesian networks. In addition to Gaussian distribution, alternative techniques such as modified exponential distribution and Rayleigh distribution can also be used to deal with continuous attributes [34]. Since only discrete Bayesian networks are employed in this paper, we use Bayesian networks to represent discrete Bayesian networks in the following paper for convenience. To utilize a Bayesian network, both the structure $\mathcal{G}$ and probability function $\mathcal{O}$ of the Bayesian network should be obtained, where $\mathcal{O}$ is quantified by a conditional probability table (CPT) that can be parameterized by θ. Depending on the characteristics of the task to be solved, the topology $\mathcal{G}$ can either be defined based on the causality of nodes or learned from data. For most RL tasks, since the state inputs and action outputs are known, the causal relationship between states and actions can be directly described by a Bayesian network structure (see also Figure 2). Thus, we focus on estimating the optimal parameter θ^* of the probability function and the probabilistic inference of Bayesian networks.

3.2.1. Parameter Estimation

3.2.1. Parameter Estimation. The parameter estimation process of Bayesian networks aims to learn the probability function $\mathcal{O}$ of all the nodes, where each node in Bayesian networks denotes a variable [35]. Providing the structure $\mathcal{G}$ of a Bayesian network, the conditional independence of all the nodes can be learned from data. Given a dataset $\mathcal{D}$ consists of fully observed samples of a Bayesian network, the maximum likelihood estimation (MLE) method is usually used to accomplish the parameter estimation process. Suppose a Bayesian network has n nodes $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ and its probability function $\mathcal{O}$ is parameterized by θ. For the node X_i in $\mathbf{X}$, we assume that it has r_i candidate values and its parent nodes $\text{parent}(X_i)$ have q_i candidate combinations. Each parameter θ_{ij}^k that represents the conditional probability between node X_i and its parent nodes $\text{parent}(X_i)$ when $X_i = k$ and $\text{parent}(X_i) = j$ can be written as

$$\theta_{ij}^k = P(X_i = k|\text{parent}(X_i) = j), \tag{6}$$

where $i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, q_i$; $k = 1, 2, \ldots, r_i$.

According to the property of probability, the accumulated sum of θ_{ij}^k over candidate values of X_i satisfies

$$\sum_{k=1}^{r_i} \theta_{ij}^k = \sum_{k=1}^{r_i} P(X_i = k|\text{parent}(X_i) = j) = 1. \tag{7}$$

The MLE method aims at learning the optimal parameter θ^* by maximizing the likelihood between the parameter θ and the dataset $\mathcal{D}$, which can be written as

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} L\left(\boldsymbol{\theta}|\mathcal{D}^E\right) = \arg\max_{\theta_{ij}^k} \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} m_{ij}^k \log\, \theta_{ij}^k, \tag{8}$$

where $L(\theta|\mathcal{D})$ is the likelihood function of θ and m_{ij}^k is the number of samples that satisfies $\text{parent}(X_i)$ when $X_i = k$ and $\text{parent}(X_i) = j$ in the dataset $\mathcal{D}$.

By using the Lagrange multiplier method, the (near) optimal θ^* can be obtained as follows:

$$\theta_{ij}^{k*} = \begin{cases} \dfrac{1}{r_i}, & \text{if } \displaystyle\sum_{k=1}^{r_i} m_{ij}^k = 0, \\[2ex] \dfrac{m_{ij}^k}{\sum_{k=1}^{r_i} m_{ij}^k}, & \text{if } \displaystyle\sum_{k=1}^{r_i} m_{ij}^k > 0. \end{cases} \tag{9}$$

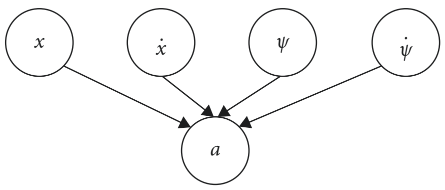

Figure 2: Structure of the Bayesian network defined for the CartPole task.

Recently, some advanced Bayesian network parameter estimation methods are also proposed for limited data [36] or uncertain data [37]. As we have enough deterministic data and the MLE method has high estimation accuracy and wide application, we choose MLE as the method for parameter estimation in this paper.

3.2.2. Probabilistic Inference. The probabilistic inference of Bayesian networks is to estimate the posterior probability on target variables by giving the learned CPTs and observed variables (also called evidence variables), which can be divided into approximate inference and exact inference. Exact inference methods aim to precisely calculate the probability distribution of variables and are suitable for Bayesian networks with simple structures. Approximate inference methods improve the computational efficiency at precision, which is suitable for Bayesian networks with complex structures.

Given the structure of a Bayesian network example shown in Figure 2, its joint probability distribution can be written as

$$P(x,\dot{x},\psi,\dot{\psi},a) = P(x)\cdot P(\dot{x})\cdot P(\psi)\cdot P(\dot{\psi})\cdot P(a|x,\dot{x},\psi,\dot{\psi}), \tag{10}$$

where $P(\cdot)$ denotes the probability distribution function.

Since the structures of Bayesian networks used in this paper are relatively uncomplicated, we can choose exact inference methods without having to sacrifice the accuracy for computational efficiency. As one of the representative exact inference methods, variable elimination (VE) can decompose the joint probability distribution, and the Bayesian network represents into a series of conditional probability products and accomplishes the inference process by integration. Therefore, giving the goal of obtaining the marginal probability $P(a)$, the VE method eliminates variables x, $\dot{x}$, ψ, and $\dot{\psi}$ in equation (10) as follows:

$$\begin{aligned} P(a) &= \sum_{\dot{\psi}}\sum_{\psi}\sum_{\dot{x}}\sum_{x} P(x,\dot{x},\psi,\dot{\psi},a) \\ &= \sum_{\dot{\psi}}\sum_{\psi}\sum_{\dot{x}}\sum_{x} P(x)\cdot P(\dot{x})\cdot P(\psi)\cdot P(\dot{\psi})\cdot P(a\mid x,\dot{x},\psi,\dot{\psi}). \end{aligned} \tag{11}$$

4. Methodology

In this section, firstly a novel state abstraction algorithm called node influence with Wasserstein distance (NIW) is proposed. Given the learned abstract states, the probabilistic knowledge extraction method with Bayesian networks is introduced in Section 4.1. Then, our RLBNK method that incorporates such probabilistic knowledge into RL is presented in Section 4.2. More specificity, two variant extensions of the RLBNK method, RLBNK-concat and RLBNK-switch, are designed for different knowledge integration approaches. Finally, we analyse and discuss the advantages of our RLBNK method in Section 4.3.

4.1. Extracting Probabilistic Knowledge by Bayesian Networks. In previous RLfD methods, demonstrations are certain instances of human knowledge for a specific task. In order to improve their generalization capability and robustness, higher-level knowledge should be extracted from demonstrations first. Since the number of demonstrations is usually insufficient to cover the entire statespace of a task, and human knowledge is naturally coarse and probabilistic, providing uncertainty in instructions is essential to utilize demonstrations well. As Bayesian networks have the advantages of extracting and representing probabilistic knowledge and are interpretable, we choose this pattern for the knowledge representation.

4.1.1. State Abstraction via NIW Algorithm. Since Bayesian networks only take discrete variables as input and output, the state abstraction should be obtained before the probabilistic knowledge extraction process, which can be done by discretization. In addition to being used to build Bayesian networks, discrete states have the advantage of being easier to understand and closer to conceptual and semantic representations than continuous states. Furthermore, the discrete state can contribute to the robustness of the learned policy compared to the original continuous state.

For convenience, we take the CartPole task as an example here and other tasks used in this paper are similar. The state vector of the CartPole task is $[x,\dot{x},\psi,\dot{\psi}]$, which represents the position and the velocity of the cart, and the angle and the angular velocity of the pole, respectively. In order to acquire the state abstraction, each state element is semantically divided into Negative, Small, and Positive. The discretization process follows equation (12), where δ_i denotes the parameter of discretization for each state element s_i in state vector. For example, δ_x, $\delta_{\dot{x}}$, δ_ψ, and $\delta_{\dot{\psi}}$ indicates the discretization parameter for each state element in $[x,\dot{x},\psi,\dot{\psi}]$ for the CartPole task:

$$S_i = \begin{cases} \text{Small}, & \text{if } |s_i| < \delta_i, \\ \text{Positive}, & \text{if } s_i \geq \delta_i, \\ \text{Negative}, & \text{if } s_i \leq -\delta_i. \end{cases} \tag{12}$$

Different δ_i for discretization would result in different representations of states, which can significantly affect the learning and inference process of Bayesian networks. Previous work has shown that abstract concepts can be learned from similarity-based approach [38], where the δ_i is determined by the similarity. Oller et al. [39] proposes a concept learning method via clustering to implicitly find.

However, this unsupervised approach does not consider the causal relationships between variables.

In Bayesian networks, the optimal state abstraction parameter δ_i should enable the most efficient prediction and inference capacity, which can be measured by the node influence [40, 41]. The node influence value stands for the discrepancy of conditional and marginal probabilities of the target probability distribution, which indicates the inference ability between variables. Based on this idea, we propose a novel state abstraction algorithm called node influence with Wasserstein distance (NIW) to find the optimal δ_i. NIW quantifies the relationship between two causal variables by describing the variability of the target probability distribution. A larger NIW value indicates a stronger inference capability between variables. We calculate the NIW value as follows:

$$\mathrm{NIW} = \frac{1}{u}\cdot\sum_{i=1}^{u}\eta\cdot D_{\mathscr{W}}\left(P(Y), tPn(Y|X=x_i)\right), \tag{13}$$

where X is the parent node of Y (See Figure 3), u is the number of discretized states, $D_{\mathscr{W}}(\cdot,\cdot)$ is the Wasserstein distance metric, and η is the ratio of the samples that satisfy $X = x_i$.

The Wasserstein distance can be calculated by

$$D_{\mathscr{W}}(p,q) = \inf_{\gamma\in\Pi[p,q]}\iint\gamma(x,y)\mathrm{d}(x,y)\mathrm{d}x\mathrm{d}y, \tag{14}$$

where $\gamma(x,y)$ satisfies

$$\begin{aligned}\int\gamma(x,y)\mathrm{d}y &= p(x),\\ \int\gamma(x,y)\mathrm{d}x &= q(y),\end{aligned} \tag{15}$$

and $\mathrm{d}(x,y)$ satisfies

$$\mathrm{d}(x,y) = \|x-y\|_2 = \sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}. \tag{16}$$

In contrast to the Kullback–Leibler divergence metric and the Jensen–Shannon divergence metric, the Wasserstein distance [42] metric can measure not only the distance between two overlapping distributions, but also the distance between two nonoverlapping distributions, which provides more useful information for evaluating the relationships of variables in Bayesian networks. By calculating the NIW values corresponding to a series of different δ_i, we can determine that the one corresponding to the maximum NIW value is the optimal δ_i.

4.1.2. Knowledge Extraction via Bayesian Networks. After determining δ_i by calculating the NIW value for discretization, the probabilistic knowledge can be extracted from data via Bayesian networks, where the knowledge extraction process is also referred to as the parameter estimating of Bayesian networks. The workflow of probabilistic knowledge extraction is shown in Figure 4. Given the discretization parameters and the original dataset $\mathscr{D}^E$ which contains continuous state variables, the original state should be firstly abstracted to $\mathscr{D}^E_{\text{abstract}}$ following equation (12). As the structure of the Bayesian network is known, the parameter of Bayesian network can be estimated according to equation (9) with abstract dataset $\mathscr{D}^E_{\text{abstract}}$.

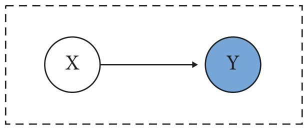

FIGURE 3: An example for the NIW value calculation.

The pseudocode of the knowledge extraction process for this section is shown in Algorithm 1.

Remark 1. According to equation (11) and equation (13), the computational complexity of Algorithm 1 can be estimated as $\mathcal{O}(n^2 2^k)$, where n is the number of nodes in the Bayesian network and k is the maximum number of parent nodes.

4.2. Incorporating Probabilistic Knowledge into Reinforcement Learning. As the Bayesian network represents the knowledge extracted from demonstrations, we use the knowledge module $\mathbb{K}_{\theta^*}$ to refer to it for convenience, where θ^* is learned following Algorithm 1. The knowledge module $\mathbb{K}_{\theta^*}$ outputs the decision confidence vector $\mathbf{p}$ that indicates the uncertainty estimation of the decisions, therefore to determine the extent to which the decision should be trusted. Formally, the output vector $\mathbf{p}$ of the probabilistic knowledge module $\mathbb{K}_{\theta^*}$ is based on the current state s following equation (11) and it can be written as

$$p = \mathbb{K}_{\theta^*}(s) = \left[p_{a_1}, p_{a_2}, \ldots, p_{a_{|\mathscr{A}|}}\right], \tag{17}$$

where p_{a_i} is the decision confidence over action a_i and the sum of all the p_{a_i} satisfies: $\sum_{i=1}^{|\mathscr{A}|} p_{a_i} = 1$.

Definition 1. In RL paradigm, the knowledge extracted by Bayesian networks can be formally defined by a tuple $\langle S_{\mathbb{K}}, \mathbb{K}_\theta\rangle$, where $\mathscr{S}_{\mathbb{K}}\subseteq\mathscr{S}$ is the state space that the knowledge module $\mathbb{K}_\theta$ works and $\mathbb{K}_\theta$ is a mapping from $\mathscr{S}_{\mathbb{K}}$ to action space $\mathscr{A}$ with high decision confidence.

Even though the knowledge module $\mathbb{K}_{\theta^*}$ plays the role of probabilistic knowledge extraction and representation, the knowledge extracted from demonstrations is still coarse and needs to be further extended and refined. Therefore, a knowledge refine module $\mathbb{F}_\phi$ should be introduced, which should at least take the decision confidence vector $\mathbf{p}$ as the input and outputs the refined decision confidence vector $\mathbf{p}'$. As a flexible universal approximator, a neural network can be combined with other patterns, including Bayesian networks, to form hybrid policies π_{hybrid}. Thus, we use a neural network-based refine module $\mathbb{F}_\phi$ here to undertake the role of knowledge refinement and propose two alternative RLBNK methods: RLBNK-concat and RLBNK-switch to approximate the refine module.

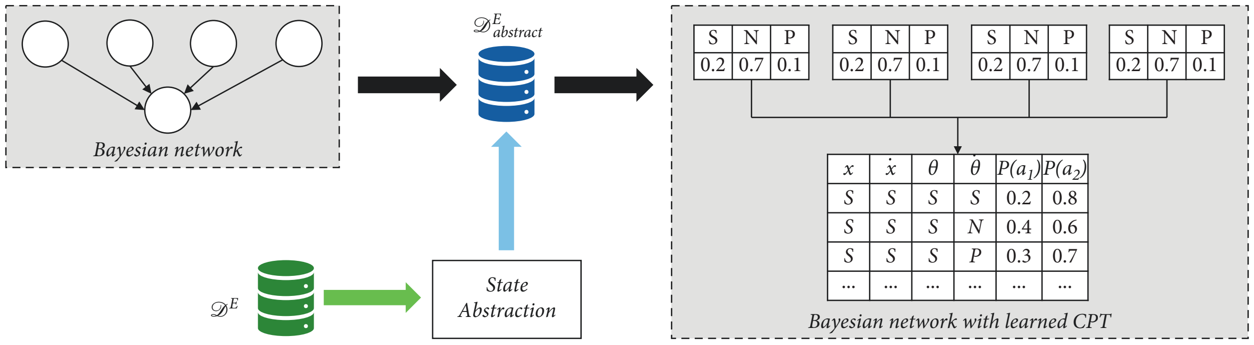

FIGURE 4: The probabilistic knowledge extraction process. The parameter estimating of the Bayesian network is based on the $\mathscr{D}^E_{\text{abstract}}$ abstracted from the original dataset $\mathscr{D}^E$ via the NIW algorithm.

```
    Input: original demonstration dataset $\mathscr{D}^E$, Structure $\mathscr{G}$ of the Bayesian network
    Output: probabilistic knowledge represented by the Bayesian network
(1) Calculate the NIW value for each discrete parameter $\delta_i$ following equation (13)
(2) Choose the optimal discrete parameter $\delta_i$ corresponding to the maximum NIW value
(3) for each state-action pair in $\mathscr{D}^E$ do
(4)     Data discretization based on equation (12) with discrete parameter $\delta_i$
(5)     Store discretized data into $\mathscr{D}^E_{\text{abstract}}$
(6) end
(7) Parameter learning with $\mathscr{D}^E_{\text{abstract}}$ following equation (9)
(8) Save the learned parameter $\theta^*$ as a conditional probability table (CPT)
```

ALGORITHM 1: Probabilistic knowledge extraction via Bayesian networks.

4.2.1. RLBNK-Concat. With the decision confidence vector $\mathbf{p}$ provided by the knowledge module $\mathbb{K}_{\theta^*}$, the first idea of incorporating knowledge into the RL process is to directly concatenate the vector $\mathbf{p}$ to the current state s as the input of the refine module $\mathbb{F}_\phi$. This idea indiscriminately considers both the current state and the decision confidence. By concatenating these two vectors as the input of the refine module, the output refined action preference vector $\mathbf{p}'$ can be obtained from the output of the refine module following

$$\mathbf{p}' = \mathbb{F}_\phi(\mathbf{p}, s) = \left[p'_{a_1}, p'_{a_2}, \ldots, p'_{a_{|\mathscr{A}|}}\right]. \tag{18}$$

For this RLBNK-concat method, we define its whole policy can be represented as $\pi^c_{\text{hybrid}} = \mathbb{K}_{\theta^*} \otimes \mathbb{F}_\phi$, where the policy π^c_{hybrid} will be optimized within the RL paradigm. Since the parameter θ^* is learned via Algorithm 1, only the parameter ϕ of the refine module will be optimized during the policy optimization process. Although the RLBNK-concat method is straightforward and feasible, it does not fully leverage the decision confidence $\mathbf{p}$ provided by the knowledge module, which results in the refine module $\mathbb{F}_\phi$ having to function in the domain with size $|\mathscr{S}| + |\mathscr{A}|$, while the original size of the state space is $|\mathscr{S}|$. However, when the RL agent encounters states in which it has a high decision confidence based on prior knowledge, it can rely solely on the prior knowledge to complete the decision-making process without further learning.

4.2.2. RLBNK-Switch. As shown in Figure 5, to better utilize the decision confidence provided by the knowledge module $\mathbb{K}_{\theta^*}$, we propose RLBNK-switch by comparing the action confidence p_{a_i} with the threshold $\wedge$ to determine the source of decisions following equation (19). More specifically, we can choose whether the action should be taken from the knowledge module or from the refine module according to decision confidence values p_{a_i}. If the decision confidence is high, the decision will be made based on the prior knowledge module $\mathbb{K}_{\theta^*}$. Otherwise, the agent can switch to the refine module $\mathbb{F}_\phi$ to make the decision, where the refine module will be further optimized by RL. Therefore, comparing to RLBNK-concat, the RL agent only learns the policy in states that are uncovered by the knowledge module. The switching process can be expressed as

$$\mathbf{p}' = \begin{cases} \mathbf{p}, & \text{if } \max(\mathbf{p}) > \wedge, \\ \mathbb{F}_\phi(s), & \text{else}, \end{cases} \tag{19}$$

where $\max(\cdot)$ is used to return the maximum element of the input vector.

For RLBNK-switch, we define its whole hybrid policy can be represented as $\pi^s_{\text{hybrid}} = \mathbb{K}_{\theta^*} \odot \mathbb{F}_\phi$. After obtaining the refined action preference vector, the action a_i should be taken if the corresponding decision confidence p'_{a_i} is the maximum element in the output vector $\mathbf{p}'$. Then, the whole policy π^s_{hybrid} will be optimized. Since the parameter θ^* is fixed, only the neural network refine module $\mathbb{F}_\phi$ will be

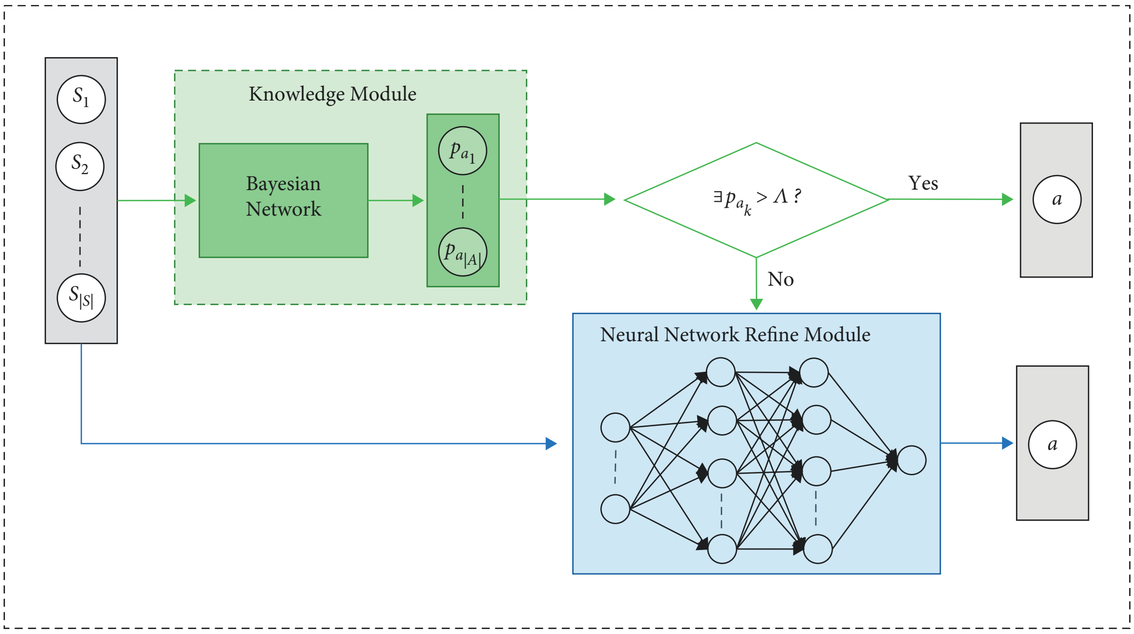

FIGURE 5: Architecture of the RLBNK-switch method. The knowledge module represented by the Bayesian network is combined with neural network-based refine module according to the decision confidence **p** over the current state.

optimized following the policy optimization procedure in RL based on equation (3).

Remark 2. Assuming that the learned knowledge module $\mathbb{K}_{\theta^*}$ is the optimal policy in state space (domain) $\mathcal{S}_{\mathbb{K}}$, because of the switch mechanism of the RLBNK-switch method, the hybrid policy π^s_{hybrid} is optimal in state space $\mathcal{S}_{\mathbb{K}}$ but is nonoptimal in state space $\mathcal{S} - \mathcal{S}_{\mathbb{K}}$. From a holistic point of view, the hybrid policy π^s_{hybrid} has an optimal initialization for partial state space, which makes this RLBNK-switch method have the same feasibility as the normal neural network-based RL algorithms.

As the proposed RLBNK method can be regarded as a general policy framework where it can be represented by $\pi^c_{\text{hybrid}} = \mathbb{K}_{\theta^*} \otimes \mathbb{F}_\phi$ (for RLBNK-concat) or $\pi^s_{\text{hybrid}} = \mathbb{K}_{\theta^*} \odot \mathbb{F}_\phi$ (for RLBNK-switch), the RLBNK method is able to combine with any policy-based RL algorithm to optimize the parameter ϕ of the refine module $\mathbb{F}_\phi$. As the proximal policy optimization (PPO) [43] algorithm is considered as a baseline RL algorithm, we apply it as the base algorithm in this paper to demonstrate the effectiveness of RLBNK.

The PPO algorithm has two variant versions and the most commonly used version is the one with clipped surrogate objective, which forms the policy gradient using the advantage function A^{π_ϕ} as introduced in equation (5) and minimizes the clipped-ratio loss $L^{\text{PPO}}(\phi)$ over samples collected by $\pi_{\phi_{\text{old}}}$. The clipped-ratio loss can be written as

$$L_t^{\text{PPO}}(\phi) = \mathbb{E}_{(s,a)\sim\pi_{\phi_{\text{old}}}}\left[\min\left(\rho(\phi)A_t^{\pi_\phi}, \text{clip}(\rho(\phi), 1-\varepsilon, 1+\varepsilon)A_t^{\pi_\phi}\right)\right], \tag{20}$$

where the clipping coefficient ϵ aims to prevent large updates. The probability ratio $\rho_t(\phi)$ used in equation (20) is introduced to measure the changed probability of the chosen action a in state s under the updated policy π_ϕ and the old policy $\pi_{\phi_{\text{old}}}$, which can be written as

$$\rho(\phi) = \frac{\pi_\phi(a|s)}{\pi_{\phi_{\text{old}}}(a|s)}. \tag{21}$$

For the policy network in PPO, the overall loss function at time step t is defined by the combination of the surrogate loss $L^{\text{PPO}}(\phi)$, the value loss $L^{\text{VF}}(\phi)$, and the entropy K. The weights of these items are adjusted by coefficients c_1 and c_2:

$$L_t(\phi) = \widehat{\mathbb{E}}_t\left[L_t^{\text{PPO}}(\phi) + c_1 L_t^{VF}(\phi) - c_2 K\left[\pi_\phi\right](s)\right]. \tag{22}$$

The weights of neural networks can be updated as follows:

$$\phi \longleftarrow \phi + \alpha\frac{\partial L_t(\phi)}{\partial \phi}. \tag{23}$$

The pseudocode of RLBNK is shown in Algorithm 2.

4.3. Performance Analysis and Discussion.

4.3. Performance Analysis and Discussion. The RLBNK method can be regarded as the neurosymbolic AI where the Bayesian network is the symbolic representation of knowledge while the refine module is represented by the neural network. Symbolism is expected to provide extra knowledge constrains for the learning process to help improve the learning efficiency, which can also prevent the well-known catastrophic forgetting of neural networks and the difficulty of extrapolation nondistributed data to improve the robustness of the algorithm [26, 44].

4.3.1. Efficiency Analysis. Formally, for the MDP defined by $\langle\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma\rangle$, the size of its policy space is $|\mathcal{A}|^{|\mathcal{S}|}$.

```
    Input: knowledge module $\mathbb{K}_{\theta^*}$ with parameter $\theta$ learned from Algorithm 1, randomly initialized refine module $\mathbb{F}_\phi$ with
  parameter $\phi$, buffer $\mathscr{B}$, update interval $T_{\text{update}}$, threshold $\wedge$, and clipping parameter $\epsilon$.
    Output: learned (near) optimal hybrid policy $\pi^*_{\text{hybrid}}$.
(1)  timestep $t = 0$
(2)  for Episode $E = 1$: $E_{\max}$ do
(3)      Initialize the state $s_0$
(4)      while $s_t$ is not the terminal state do
(5)          timestep $t = t + 1$
(6)          Compute the decision confidence vector $\mathbf{p}_t = \mathbb{K}_{\theta^*}(s_t)$
(7)          Refine the decision confidence $\mathbf{p}_t$ to get $\mathbf{p}_t'$ based on equation (18) (for RLBNK-concat)
(8)          $a_t = \arg\max(\mathbf{p}_t')$
(9)          Execute the action $a_t$, then receive the instant reward $r_t$ and transit to the next state $s_{t+1}$
(10)         if the action $a_t$ is derived from $\mathbb{F}_\phi$ then
(11)             Store $(s_t, a_t, r_t, \mathbb{F}_{\phi_{\text{old}}}(a_t|s_t))$ in the $\mathscr{B}$
(12)         end
(13)         $s_t \longleftarrow s_{t+1}$
(14)         if $t$ mod $T_{\text{update}} = 0$ then
(15)         Train $\phi$ of the refine module $\mathbb{F}_\phi$ following equation (21)–(23)
(16)             Clear buffer $\mathscr{B}$
(17)             timestep $t = 0$
(18)         end
(19)     end
(20) end
```

ALGORITHM 2: Pseudocode of the RLBNK method.

Assuming that the knowledge module provides high decision confidence values in state set $\mathcal{S}_{\mathbb{K}}$, where $\mathcal{S}_{\mathbb{K}} \subseteq \mathcal{S}$, the policy space of the refine module $\mathbb{F}$ for RLBNK-switch is reduced from $|\mathscr{A}|^{|\mathcal{S}|}$ to $|\mathscr{A}|^{|\mathcal{S}-\mathcal{S}_{\mathbb{K}}|}$. Therefore, the uncertainty-based state space partitioning can make RLBNK-switch theoretically enjoy better data efficiency performance. Additionally, the knowledge module can cover the policy space $|\mathscr{A}|^{|\mathcal{S}_{\mathbb{K}}|}$, which is fixed to prevent the catastrophic forgetting as well as reducing the overall policy space that needs to be learn. And the knowledge represented by Bayesian networks provides better generalization and robustness over the neural network-based method because of the state abstraction and the probabilistic property of Bayesian networks. For RLBNK-switch, the RL algorithm is employed to learn a policy for the state space $\mathcal{S} - \mathcal{S}_{\mathbb{K}}$. Therefore, the gradient estimator also turns from equation (4) into

$$\nabla J(\phi) = \mathbb{E}_{s\sim(\mathcal{S}-\mathcal{S}_{\mathbb{K}}),a\sim A}\left[\nabla_\phi \log \pi_\phi(a|s) A^{\pi_\phi}(s,a)\right], \quad (24)$$

which also avoids an integral over the full state space to make the learning more efficient. Moreover, for RLBNK-concat, because of the concatenation operation, the policy space is increased from $|\mathscr{A}|^{|\mathcal{S}|}$ to $|\mathscr{A}|^{|\mathcal{S}|+|\mathscr{A}|}$. Therefore, we expect that RLBNK-switch method demonstrates a better data efficiency performance than RLBNK-concat.

4.3.2. Robustness Analysis. Conditional independence used in Bayesian networks is the basic and robust form of knowledge. The Bayesian network classifier is robust, and we can learn the parameters of conditional distribution even with relatively few training examples [35]. Also, the variance that Bayesian networks provide makes them act robust. Besides, in our paper, the knowledge are constrained by the threshold $\wedge$, which also improves the robustness. The state abstraction (discretization) NIW method also plays an important role to improve the robustness. Discrete values are about intervals of numbers which are more concise to specify, easier to use, and comprehend as they are closer to a concept-level representation than continuous ones [45, 46]. From the perspective of machine learning, state abstraction reduces the risk of overfitting by minimizing structural risk and eliminates noisy samples by simplifying the data, both of which enhance the robustness and generalization capability.

5. Experiments

In this section, we conduct experiments to evaluate our RLBNK method. More specifically, for the experiments below, we aim to evaluate our proposed RLBNK method to confirm the following:

(i) Our RLBNK method contributes to the data efficiency of RL under the normal reward setting and even sparse reward settings.

(ii) The knowledge extracted from demonstrations through Bayesian networks can be generalized to similar tasks, providing instructive guidance for the RLBNK method to obtain effective hybrid policies.

(iii) With the help of the knowledge learned by Bayesian networks, the hybrid policy π_{hybrid} learned by the RLBNK method can robustly handle noisy observations from the environment.

All the experiments in this paper are conducted in the Ubuntu 16.04 system with PyTorch 1.7. Our algorithms are based on the open-source PPO-PyTorch [47] implementation and the probabilistic graphical model toolkit pgmpy [48]. We test our algorithms on the OpenAI Gym [49] environment and the PLE [50] environment. Below we briefly describe the tasks used in our experiments (also see Figure 6).

CartPole. In the CartPole system, a cart moves along a friction-less track and the pole is attached by an unactuated joint to the cart. The goal of this task is to balance the pole vertically upward as long as possible.

Catcher. In the Catcher task, the paddle has to catch the falling fruit with three different actions (moving left, moving right, and doing nothing), the RL agent has access to the position and speed of both the player and the fruit.

FlappyBird. FlappyBird is a side-scrolling game where the bird takes actions (flapping or doing nothing) to fly through gaps between pairs of pipes. The agent receives the reward once the bird passes through a pipe and the episode ends when the bird hits pipes or gets out of the screen.

5.1. Simulation Settings. To ensure the fairness of our experiments, we keep all the hyperparameters the same as the original implementation as recommended in the corresponding literature. For CartPole and Catcher task, 2000 state-action pairs (s, a) are collected by an expert policy $\pi_E$to form the original expert demonstration dataset $\mathscr{D}^E$, and for FlappyBird task, 150 state-action pairs are collected via the same way. Specially, the update interval of networks T_{update} is set to 2000, and the clipping parameter ϵ for policy optimization is set to 0.2. All the neural networks used in this paper have 2 hidden layers, each containing 64 neurons. The optimal discretization parameters δ_i for state abstraction during the knowledge extraction process are are shown in the tables in Appendix, where the parameter corresponding to the maximum NIW value is the optimal parameter for subsequent experiments. For RLBNK-switch, the knowledge module threshold parameter $\wedge$ is set to 0.8 as default. For each algorithm and each task, we train 5 policies with different seeds and the shaded region for each curve in the following results denotes the standard deviation of the average evaluation.

5.2. Data Efficiency of the RLBNK Method. To evaluate the validity of the proposed RLBNK method, we first conduct experiments on three tasks mentioned above under the normal reward setting. To further demonstrate the effectiveness of the proposed RLBNK method, we also set up CartPole tasks with variant sparse reward settings. The performance under different reward settings is shown in Figures 7 and 8, respectively.

5.2.1. Performance Comparison under the Normal Reward Setting. Curves in Figure 7 illustrate the mean and variance of the cumulative reward in each episode for the training process of RLBNK-switch, RLBNK-concat, baseline PPO [43], and DQfD [16] in these tasks. The Expert curve denotes the performance of the expert policy π_E used to collect demonstrations, and the Imitation curve is the performance of the policy trained using demonstrations via behavior cloning [11].

From Figure 7, we can observe that both proposed RLBNK-switch and RLBNK-concat outperform other baseline methods in most cases and RLBNK-switch demonstrates a jump-start for all three tasks at the beginning of each training process. Both proposed algorithms obtain higher rewards within fewer training episodes. Especially, the performance of RLBNK-switch in all three tasks learns a good (even near optimal) policy within 200 episodes. Compared to the baseline algorithm PPO that explores the environment from scratch, our method is superior by leveraging the knowledge extracted from demonstrations. In contrast, although the DQfD method utilizes the same demonstration data as RLBNK-switch and RLBNK-concat, it performs mediocrely in all cases except for the CartPole task where it outperforms than RLBNK-concat. We assume this is because the reward setting in Catcher and FlappyBird is relatively sparser than the CartPole task and this will be further analysed in the next experiment. This result confirms that the proposed RLBNK method can effectively utilize the knowledge and achieve superior performance.

5.2.2. Performance Comparison under Sparse Reward Settings. We further demonstrate the superiority of RLBNK-switch and RLBNK-concat under sparse reward conditions. To facilitate experimental validation, we propose a sparse reward setup: multistep cumulative rewards are given at sparse time steps. We choose the CartPole task to simulate this setup and provide T-step cumulative rewards at every T time step (the rewards are only provided at T, $2T$, $3T$,. . .). Figure 8 shows the experimental results under different sparse settings.

From Figure 8, we can remark that our RLBNK-switch converges within around 200 episodes and demonstrates smaller variance a consistent performance for all three sparse settings ranging from 25 to 100. The baseline PPO is hard to learn an effective policy under sparse reward settings since the PPO agent has less chance to obtain reward signals in the early pure exploration phase of learning. For DQfD, even if it achieves a good preference under normal reward setting in CartPole task, it acts the worse learning process in all sparse settings. We believe that one possible reason is that the priority sampling mechanism used by the DQfD algorithm hinders the Q-network updates under sparse reward conditions. This priority sampling mechanism gives more priority to demonstrations during the Q-network update process. However, due to sparse reward settings, the DQfD agent has difficulty obtaining positive samples from the environment itself, so the Q-network in DQfD may still be optimized with pure demonstrations for most of the time, even though the agent is interacting with the environment. Since demonstrations only cover part of the state space, it cannot optimize the Q-network well

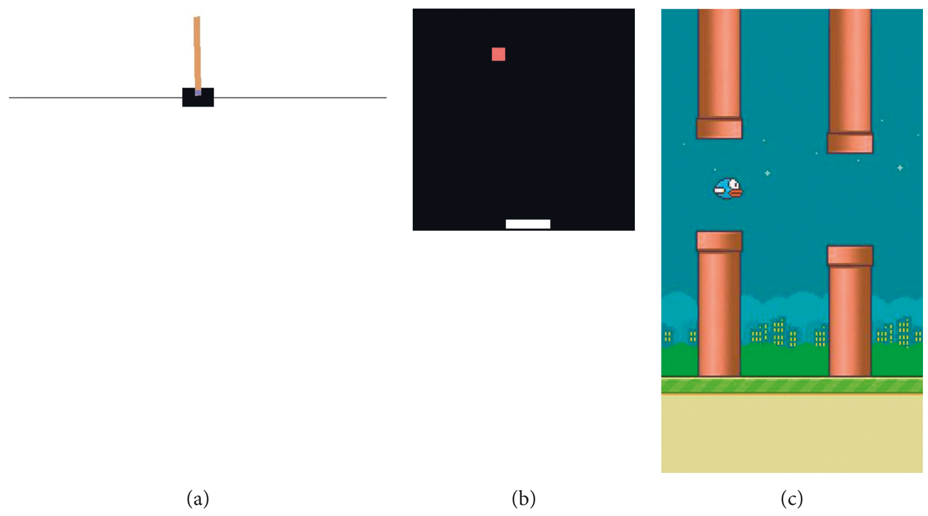

FIGURE 6: The benchmark tasks used in this paper. The CartPole task is from the OpenAI Gym environment, and the Catcher and FlappyBird tasks are from the PLE environment. (a) CartPole. (b) Catcher. (c) FlappyBird.

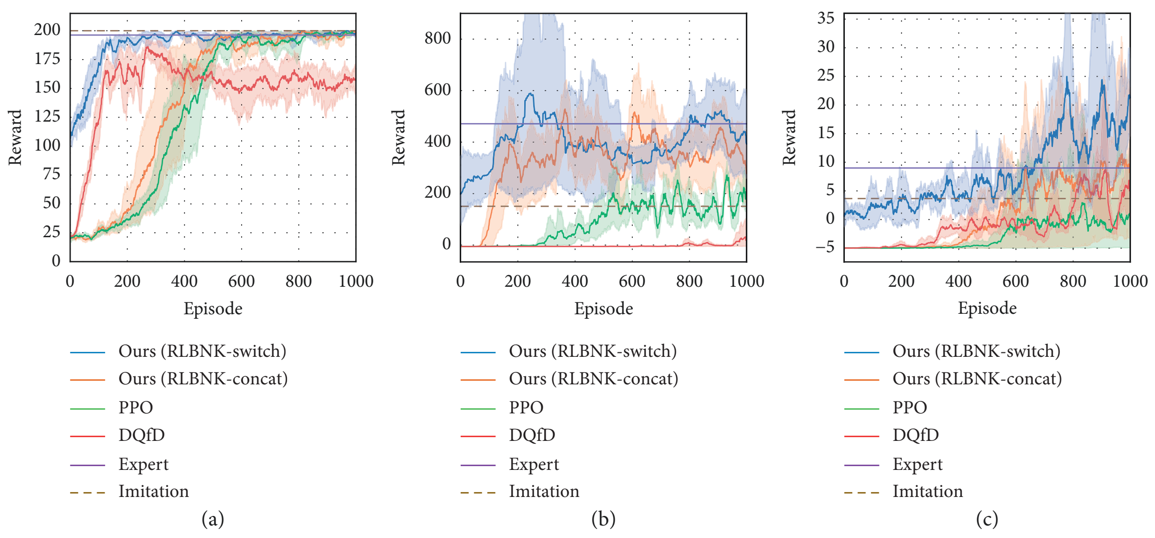

FIGURE 7: Comparison of RLBNK-switch and RLBNK-concat to the baseline PPO, DQfD, expert policy, and pure imitation learning under the normal reward setting. Plots show the training performance over the number of episodes. (a) CartPole. (b) Catcher. (c) FlappyBird.

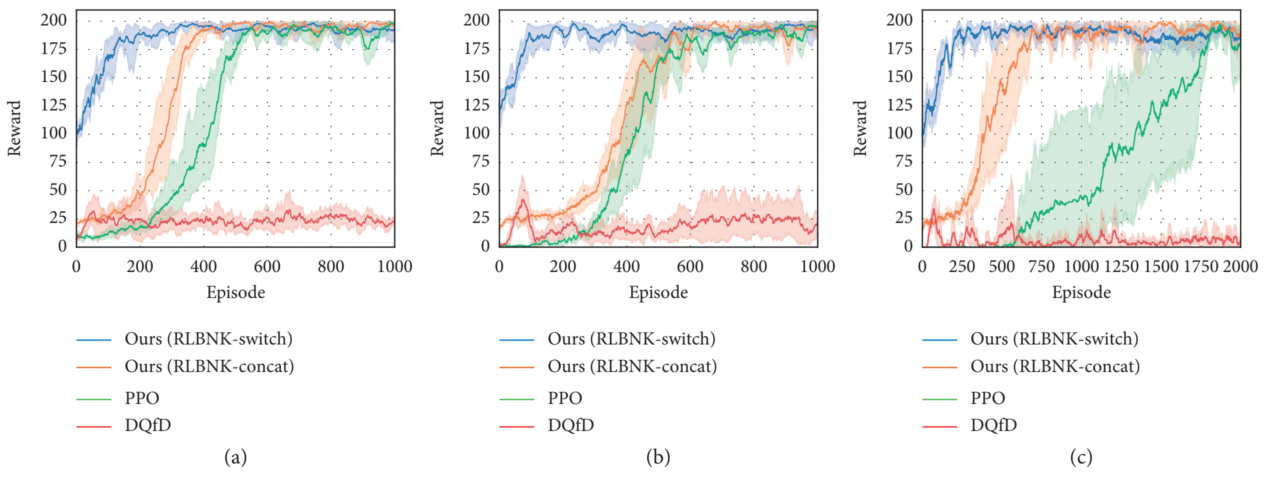

FIGURE 8: Experimental results for CartPole task under different sparse reward settings, where T denotes the sparse interval of receiving rewards for the agent. Plots show the training performance over the number of episodes. (a) $T = 25$. (b) $T = 50$. (c) $T = 100$.

Table 1: The changed system dynamics settings between the source MDP $\mathscr{M}_s$ and the target MDP $\mathscr{M}_t$.

The changed item	Item setting in source MDP	Item setting in target MDP
Pole length (m)	1.0	3.0
Cart mass (kg)	1.0	10.0

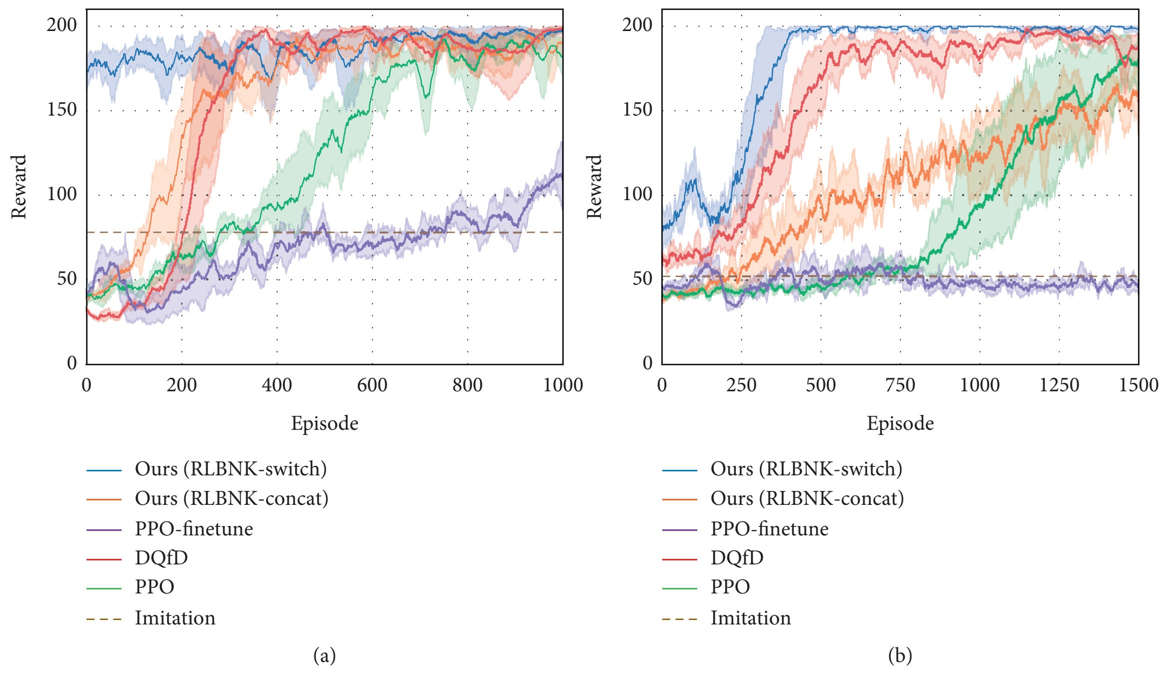

Figure 9: Comparison of RLBNK-switch and RLBNK-concat to the PPO-finetune, baseline PPO, DQfD, and imitation learning in two generalization settings. Plots show the training performance over the number of episodes. (a) Pole length generalization. (b) Cart mass generalization.

enough to obtain a well-performed policy. Moreover, from the learning curves shown in Figure 8, for DQfD and baseline PPO, the task becomes harder as the sparse factor T increases, while both RLBNK-switch and RLBNK-concat are less influenced.

5.3. Evaluation of the Generalization Capability. In this section, we perform experiments to examine the generalization capability of the RLBNK method. Here, we focus on the generalization settings that $\mathscr{M}_s$ and $\mathscr{M}_t$ share the same state space $\mathscr{S}$, action space $\mathscr{A}$, and reward function $\mathscr{R}$ but differ by the system dynamics: $\mathscr{P}_s \neq \mathscr{P}_t$. Specifically, we adopt the CartPole task here and change the length of the pole and the mass of the cart for the generalization settings as shown in Table 1. Note that given the demonstrations collected in the source MDP $\mathscr{M}_s$, our aim is to solve the target MDP $\mathscr{M}_t$.

In this experiment, we carry out several baselines, including PPO [43], PPO-finetune, DQfD [16], and Imitation (via supervised behavior cloning [11]). The PPO-finetune curve denotes the performance of the RL policy pretrained by PPO in the source MDP $\mathscr{M}_s$ and then fine-tuned in the target MDP $\mathscr{M}_t$. The PPO curve illustrates the performance of the baseline PPO directly trained in the target MDP $\mathscr{M}_t$. DQfD utilizes the demonstrations collected in the source MDP and explores in the target MDP. In the Imitation curve, the corresponding policy is trained with the expert demonstrations collected from the source MDP $\mathscr{M}_s$ via supervised behavior cloning. The curve shows its performance in the target MDP $\mathscr{M}_t$.

From Figure 9, it can be observed that in both generalization settings, directly imitating the demonstrations collected from the source MDP $\mathscr{M}_s$ cannot achieve a good performance in the target MDP (as shown in the Imitation curve). In contrast, RLBNK-switch achieves the best performance and with the help of the knowledge learned from source MDP, and both RLBNK-switch and RLBNK-concat outperform the baseline PPO algorithm. Since the policy in the PPO-finetune method is initialized as a well-trained policy in the source MDP and fine-tunes in the target MDP, it is a powerful method that achieves comparable performance to RLBNK-concat in the pole length generalization settings and even surpasses the RLBNK-concat in the cart mass generalization settings. In contrast, the DQfD method demonstrates the worse performance in both settings. One possible reason is that after the pretraining process, the demonstrations collected from the source MDP, although not suitable for the target MDP, are still used indiscriminately to update the Q-network, which hinders its optimization process in the target MDP. The empirical results in both pole length generalization and cart mass generalization

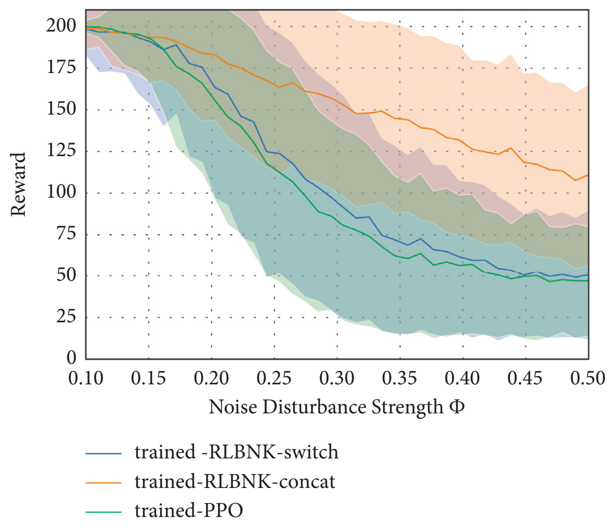

Figure 10: The cumulative reward (mean ± standard deviation with 500 rollouts) of RLBNK-switch and RLBNK-concat trained policies versus the trained PPO baseline policy when tested in disturbed CartPole task. Plots show the performance of each policy over the disturbance strength Φ.

Table 2: Discretization parameter comparison for the CartPole task.

Position $\mathbf{x}$	$\delta_{\mathbf{x}}$	0.05	0.10	0.15	0.20	0.25
	NIW	0.064	0.066	**0.068**	0.058	0.043
Velocity $\dot{x}$	$\delta_{\dot{x}}$	0.2	0.4	0.6	0.8	1.0
	NIW	0.028	0.061	0.118	**0.129**	0
Angle ψ	δ_{ψ}	0.04	0.05	0.06	0.07	0.08
	NIW	0.054	0.067	**0.095**	0.043	0.065
Angular velocity $\dot{\psi}$	$\delta_{\dot{\psi}}$	0	0.02	0.04	0.06	0.08
	NIW	**0.272**	0.262	0.249	0.172	0.181

Table 3: Discretization parameter comparison for the Catcher task.

Distance $\mathbf{x}$	δ_{x}	7	9	11	13	15
	NIW	0.079	0.095	**0.117**	0.104	0.103
Height y	δ_{y}	34	36	38	40	42
	NIW	0.013	0.019	**0.022**	0.017	0.017
Velocity v	δ_{v}	3	4	5	6	7
	NIW	0.101	0.088	**0.114**	0.105	0

Table 4: Discretization parameter comparison for the FlappyBird task.

Distance y	δ_{y}	0	5	10	15	20
	NIW	**0.192**	0.163	0.090	0.128	0.120
Distance x	δ_{x}	20	40	60	80	100
	NIW	0.028	0.048	**0.056**	0.030	0.034
Velocity v	δ_{v}	0	2	4	6	8
	NIW	0.206	**0.235**	0.208	0.189	0.020

show the strong evidence that the proposed RLBNK-switch achieves superior performance in generalization settings and the RLBNK-concat also demonstrates a comparable result to the PPO-finetune method. RLBNK-switch and RLBNK-concat can not only improve the data efficiency but also can generalize to tasks with different system dynamics.

5.4. Robustness against Stochastic Disturbances. To make the learned policy achieve robustness against stochastic observation disturbances is one of the goals of POMDP. We extensively evaluate the robustness of the RLBNK method in the CartPole task by injecting stochastic disturbance ζ to the state s, and the observation o_t satisfies

$$o_t = s_t + \zeta, \tag{25}$$

where ζ is sampled uniformly from the set U: $U(0, \Phi) = \{\zeta: \|\zeta\|_\infty \leq \Phi\}$ and the disturbance strength Φ denotes the upper bound of the stochastic disturbance. To evaluate the robustness, we firstly conduct baseline PPO, RLBNK-switch, and RLBNK-concat in the CartPole task to obtain their well-performed policies in the environment without noise following the settings introduced in Section 5.1. Then, 500 rollouts are conducted for each trained policy under the environment with a specific noise disturbance strength range from 0.10 to 0.50 to obtain the mean and standard deviation of the cumulative reward.

Figure 10 shows the performance of these policies against stochastic disturbance ζ. We can observe that as the disturbance strength Φ increases, the performance of all the learned policies becomes progressively worse. However, our RLBNK method demonstrates better robustness than the learned baseline neural network-based PPO policy for almost all ranges of disturbance strength. Especially, the RLBNK-concat performs significantly better than the RLBNK-switch. We argue that the reason for this phenomenon is due to the fact that the Bayesian network in RLBNK-switch only functions in the state space $|\mathcal{S}_{\mathbb{K}}|$, so its robustness works only in this part of the state space. For RLBNK-concat, even though directly concatenating the state s and decision confidence vector $\mathbf{p}$ enlarges the state space for the policy to search, the Bayesian network can provide the robustness for the entire state space.

6. Conclusion

In this paper, we develop a novel RLfD method called RLBNK that employs Bayesian networks to extract probabilistic knowledge from expert demonstrations to assist in RL, which provides an alternative perspective of exploiting demonstrations in RLfD. Compared with other RLfD methods, RLBNK utilizes Bayesian networks to extract probabilistic knowledge from demonstrations, which not only enables interpretability of presentation data but also enhances the generalization of the demonstrations. We further extend the RLBNK method to RLBNK-concat and RLBNK-switch and use PPO as the basic policy optimization paradigm. Extensive experiments are conducted on different tasks and the results validate that by utilizing the knowledge module represented by Bayesian networks and the knowledge refine module, both RLBNK-concat and RLBNK-switch outperform other baseline methods in normal reward and sparse reward settings and provides a jump-start at the beginning of the training. More importantly, RLBNK demonstrates a superior performance in generalization settings. Besides, the policy trained by RLBNK is more robust to the environment noise comparing to the policy trained by RL with neural network function approximators. In future work, we will scale our RLBNK to pixel-based decision-making tasks by incorporating feature dimension reduction methods such as variational autoencoders (VAEs).

Appendix

Tables 2–4 list the calculated node influence with Wasserstein distance metric (NIW) value with different discretized parameters on datasets collected from three tasks introduced in 5.1. As NIW value reflects the prediction capacity, the threshold parameter corresponding to the maximum NIW value is preferred and will be used in our experiments.

Data Availability

The data used to support the findings of this study are included within the article.

Disclosure

Yichuan Zhang is the first author.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant Nos. 61825305 and 61703418.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] S. Carta, A. Corriga, A. Ferreira, A. S. Podda, and D. R. Recupero, "A multi-layer and multi-ensemble stock trader using deep learning and deep reinforcement learning," *Applied Intelligence*, vol. 51, no. 2, pp. 889–905, 2021.

[3] X. Zhao, L. Zhang, Z. Ding, L. Xia, J. Tang, and D. Yin, "Recommendations with negative feedback via pairwise deep reinforcement learning," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018*, Y. Guo and F. Farooq, Eds., pp. 1040–1048, ACM, London, UK, August 2018.

[4] S. Schaal, "Learning from demonstration," in *Proceedings of the Advances in Neural Information Processing Systems 9, NIPS*, M. Mozer, M. I. Jordan, and T. Petsche, Eds., MIT Press, Denver, CO, USA, pp. 1040–1046, December 1996.

[5] M. Vecerík, T. Hester, J. Scholz et al., "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," 2017, http://arxiv.org/abs/1707.08817.

[6] X. Liang, T. Wang, L. Yang, and E. Xing, "CIRL: controllable imitative reinforcement learning for vision-based self-driving," in *Proceedings of the 15th European Conference Computer Vision-ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., pp. 604–620, Springer, Munich, Germany, September 2018.

[7] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters, "An algorithmic perspective on imitation learning," *Found. Trends Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018.

[8] L. von Rueden, S. Mayer, K. Beckh et al., "Informed machine learning—a taxonomy and survey of integrating prior

knowledge into learning systems," *IEEE Transactions on Knowledge and Data Engineering*, p. 1, 2021.

[9] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Elsevier, Amsterdam, Netherlands, 2014.

[10] L. C. Cobo, K. Subramanian Jr., C. L. Isbell, A. D. Lanterman, and A. L. Thomaz, "Abstraction from demonstration for efficient reinforcement learning in high-dimensional domains," *Artificial Intelligence*, vol. 216, pp. 103–128, 2014.

[11] M. Bain and C. Sammut, "A framework for behavioural cloning," in *Machine Intelligence 15, Intelligent Agents*, K. Furukawa, D. Michie, and S. Muggleton, Eds., pp. 103–129, Oxford University Press, Oxford, UK, 1995.

[12] D. Silver, A. Huang, C. J. Maddison et al., "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[13] C. Atkinson, B. McCane, L. Szymanski, and A. Robins, "Pseudo-rehearsal: achieving deep reinforcement learning without catastrophic forgetting," *Neurocomputing*, vol. 428, pp. 291–307, 2021.

[14] A. Hussein, E. Elyan, M. M. Gaber, and C. Jayne, "Deep reward shaping from demonstrations," in *Proceedings of the 2017 International Joint Conference on Neural Networks, IJCNN 2017*, pp. 510–517, IEEE, Anchorage, AK, USA, May 2017.

[15] S. Reddy, A. D. Dragan, and S. Levine, "SQIL: imitation learning via reinforcement learning with sparse rewards," in *In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020*, Addis Ababa, Ethiopia, April 2020.

[16] T. Hester, M. Vecerík, O. Pietquin et al., "Deep q-learning from demonstrations," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*, S. A. McIlraith and K. Q. Weinberger, Eds., pp. 3223–3230pp. 3223–, New Orleans, LA, USA, February 2018.

[17] T. P. Lillicrap, J. J. Hunt, A. Pritzel et al., "Continuous control with deep reinforcement learning," in *Proceedings of the 4th International Conference on Learning Representations, ICLR 2016*, Y. Bengio and Y. LeCun, Eds., San Juan, Puerto Rico, May 2016.

[18] F. Codevilla, M. Müller, A. M. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *Proceedings of the 2018 IEEE International Conference on Robotics and Automation, ICRA 2018*, pp. 1–9, IEEE, Brisbane, Australia, May 2018.

[19] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010*, Y. W. Teh and D. M. Titterington, Eds., pp. 661–668pp. 661–, Sardinia, Italy, May 2010.

[20] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza et al., "Generative adversarial nets," in *Proceedings of the Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., pp. 2672–2680, Montreal, Canada, December 2014.

[21] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proceedings of the Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016*, D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, Eds., pp. 4565–4573pp. 4565–, Barcelona, Spain, December 2016.

[22] Y. Zhou, H. Liu, J. Cao, and S. Li, "Composite learning fuzzy synchronization for incommensurate fractional-order chaotic systems with time-varying delays," *International Journal of Adaptive Control and Signal Processing*, vol. 33, no. 12, pp. 1739–1758, 2019.

[23] Y. Zhou, H. Wang, and H. Liu, "Generalized function projective synchronization of incommensurate fractional-order chaotic systems with inputs saturation," *International Journal of Fuzzy Systems*, vol. 21, no. 3, pp. 823–836, 2019.

[24] Z. Gao, F. Lin, Y. Zhou, H. Zhang, K. Wu, and H. Zhang, "Embedding high-level knowledge into dqns to learn faster and more safely," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 13608-13609, Vancouver, Canada, 2020.

[25] M. E. Taylor and P. Stone, "Cross-domain transfer for reinforcement learning," in *Proceedings of the Twenty-Fourth International Conference (ICML 2007)*, Z. Ghahramani, Ed., pp. 879–886, ACM, Corvallis, OR, USA, June 2007.

[26] P. Zhang, J. Hao, W. Wang et al., "Kogun: accelerating deep reinforcement learning via integrating human suboptimal knowledge," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, C. Bessiere, Ed., pp. 2291–2297, IJCAI 2020, Yokohama, Japan, 2020.

[27] O. Bastani, Y. Pu, and A. Solar-Lezama, "Verifiable reinforcement learning via policy extraction," in *Proceedings of the Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018*, S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., pp. 2499–2509, Montréal, Canada, December 2018.

[28] P. Kontschieder, M. Fiterau, A. Criminisi, and S. R. Bulò, "Deep neural decision forests," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016*, S. Kambhampati, Ed., pp. 4190–4194, New York, NY, USA, July 2016.

[29] C. Drummond, "Accelerating reinforcement learning by composing solutions of automatically identified subtasks," *Journal of Artificial Intelligence Research*, vol. 16, pp. 59–104, 2002.

[30] P. Madumal, T. Miller, L. Sonenberg, and F. Vetere, "Explainable reinforcement learning through a causal lens," in *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, the Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020*, vol. 34, pp. 2493–2500, New York, NY, USA, February 2020.

[31] F. L. da Silva, P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "Uncertainty-aware action advising for deep reinforcement learning agents," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 4, pp. 5792–5799, New York, NY, USA, February 2020.

[32] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, USA, 2018.

[33] G. E. Monahan, "State of the art-a survey of partially observable Markov decision processes: theory, models, and algorithms," *Management Science*, vol. 28, no. 1, pp. 1–16, 1982.

[34] P. Yildirim and D. Birant, "Naive bayes classifier for continuous variables using novel method (NBC4D) and distributions," in *Proceedings of the 2014 IEEE International Symposium on Innovations in Intelligent Systems and*

Applications, INISTA 2014, pp. 110–115, IEEE, Alberobello, Italy, June 2014.

[35] D. Koller and N. Friedman, *Probabilistic Graphical Models-Principles and Techniques*, MIT Press, Cambridge, MA, USA, 2009.

[36] J. Yang, Y. Wang, S. Pei, and Q. Hu, "Monotonicity induced parameter learning for bayesian networks with limited data," in *Proceedings of the 2018 International Joint Conference on Neural Networks, IJCNN 2018*, pp. 1–8, IEEE, Rio de Janeiro, Brazil, July 2018.

[37] S. Wasserkrug, R. Marinescu, S. Zeltyn, E. Shindin, and Y. A. Feldman, "Learning the parameters of bayesian networks from uncertain data," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 12190–12197, Vancouver, Canada, 2021.

[38] K. Lamberts and D. Shanks, *Knowledge Concepts and Categories*, Psychology Press, Hove, UK, 2013.

[39] M. Oller, J. Wu, Z. Wu, J. B. Tenenbaum, and A. Rodriguez, "See, feel, act: hierarchical learning for complex manipulation skills with multisensory fusion," *Science Robotics*, vol. 4, no. 26, 2019.

[40] J. R. Koiter, "Visualizing inference in Bayesian networks," Master thesis, Delft University of Technology, Delft, Netherlands, 2006.

[41] J. Li, B. Dai, X. Li, X. Xu, and D. Liu, "A dynamic bayesian network for vehicle maneuver prediction in highway driving scenarios: framework and verification," *Electronics*, vol. 8, no. 1, p. 40, 2019.

[42] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International Journal of Computer Vision*, vol. 40, no. 2, pp. 99–121, 2000.

[43] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, http://arxiv.org/abs/1707.06347.

[44] I. Donadello, L. Serafini, and A. S. d'Avila Garcez, "Logic tensor networks for semantic image interpretation," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017*, C. Sierra, Ed., pp. 1596–1602, Melbourne, Australia, August 2017.

[45] S. García, J. Luengo, J. A. Sáez, V. López, and F. Herrera, "A survey of discretization techniques: taxonomy and empirical analysis in supervised learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 4, pp. 734–750, 2013.

[46] H. Liu, F. Hussain, C. L. Tan, and M. Dash, "Discretization: an enabling technique," *Data Mining and Knowledge Discovery*, vol. 6, no. 4, pp. 393–423, 2002.

[47] N. Barhate, "Minimal pytorch implementation of proximal policy optimization," 2021, https://github.com/nikhilbarhate99/PPO-PyTorch.

[48] A. Ankan and A. Panda, "PGMPY: probabilistic graphical models using python," in *Proceedings of the 14th Python in Science Conference (SCIPY 2015)*, Austin, TX, USA, 2015.

[49] G. Brockman, V. Cheung, L. Pettersson et al., "OpenAI gym," *CoRR*, 2016.

[50] N. Tasfi, "Pygame learning environment," 2016, https://github.com/ntasfi/PyGame-Learning-Environment.