Efficient Bayesian Policy Reuse With a Scalable Observation Model in Deep Reinforcement Learning

Jinmei Liu, *Graduate Student Member, IEEE*, Zhi Wang, *Member, IEEE*, Chunlin Chen, *Senior Member, IEEE*, and Daoyi Dong, *Fellow, IEEE*

***Abstract*— Bayesian policy reuse (BPR) is a general policy transfer framework for selecting a source policy from an offline library by inferring the task belief based on some observation signals and a trained observation model. In this article, we propose an improved BPR method to achieve more efficient policy transfer in deep reinforcement learning (DRL). First, most BPR algorithms use the episodic return as the observation signal that contains limited information and cannot be obtained until the end of an episode. Instead, we employ the state transition sample, which is informative and instantaneous, as the observation signal for faster and more accurate task inference. Second, BPR algorithms usually require numerous samples to estimate the probability distribution of the tabular-based observation model, which may be expensive and even infeasible to learn and maintain, especially when using the state transition sample as the signal. Hence, we propose a scalable observation model based on fitting state transition functions of source tasks from only a small number of samples, which can generalize to any signals observed in the target task. Moreover, we extend the offline-mode BPR to the continual learning setting by expanding the scalable observation model in a plug-and-play fashion, which can avoid negative transfer when faced with new unknown tasks. Experimental results show that our method can consistently facilitate faster and more efficient policy transfer.**

***Index Terms*— Bayesian policy reuse (BPR), continual learning, deep reinforcement learning (DRL), observation model, transfer learning.**

NOMENCLATURE

Notation	Description
s	State.
a	Action.
r	Reward.
$\mathcal{P}$	Transition function.
π	Policy.
γ	Discount factor.
M	Maximum number of steps.
U	Episode total discounted reward function.
K	Number of episodes.
τ	Task instance.
$\mathcal{T}$	Task set.
Π	Policy library.
σ	Observation signal.
β	Belief model.
P	Probability distribution.
$\mathcal{D}$	State transition sample dataset.
ϑ	Neural network parameters.
$\mathcal{N}$	Gaussian function.
κ	Kernel function.
δ	Signal variance of the RBF.
l	Characteristic length scale of the RBF.

Manuscript received 19 March 2022; revised 28 October 2022 and 1 March 2023; accepted 26 May 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62006111 and Grant 62073160, in part by the Australian Research Council's Future Fellowship funding scheme under Project FT220100656, and in part by the Natural Science Foundation of Jiangsu Province of China under Grant BK20200330. *(Corresponding authors: Zhi Wang; Chunlin Chen.)*

Jinmei Liu, Zhi Wang, and Chunlin Chen are with the Department of Control Science and Intelligent Engineering, School of Management and Engineering, and the Research Center for Novel Technology of Intelligent Equipment, Nanjing University, Nanjing 210093, China (e-mail: jinmeiliu@smail.nju.edu.cn; zhiwang@nju.edu.cn; clchen@nju.edu.cn).

Daoyi Dong is with the School of Engineering and Information Technology, University of New South Wales, Canberra, ACT 2600, Australia (e-mail: daoyidong@gmail.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2023.3281604.

Digital Object Identifier 10.1109/TNNLS.2023.3281604

I. INTRODUCTION

REINFORCEMENT learning (RL) [1] is a general optimization framework of how an artificial agent learns an optimal policy to maximize the cumulative reward by interacting with its environment. RL has been used to find optimal controllers [2], [3], [4] and realize human–computer interaction [5], and has a broad prospect in other practical applications [6]. With recent advances, deep RL (DRL) has achieved state-of-the-art performance on various tasks [7], [8], [9], such as video games [10], board games [11], robotics [12], autonomous driving [13], and quantum information [14]. However, DRL algorithms are usually sensitive to the choice of hyper-parameters [15], [16] and typically require numerous samples to converge to good policies [17], [18], which can be computationally intensive in real-world applications.

To address this concern, transfer RL [19] is usually introduced to reduce the number of samples required for learning a target task by reusing previously acquired knowledge from relevant source tasks [20], [21], [22]. Reusing policies are widely investigated kind of approaches for this topic [23], [24], [25], [26], [27], [28], since they are intuitive, direct, and do not rely on value functions that may be difficult or unavailable to transfer.

In this article, we focus on the problem of transferring policies from multiple source tasks in DRL. Bayesian policy reuse (BPR) [29] is a general transfer framework for responding to

a target task by selecting the most appropriate policy from an offline library based on some observation signals and a trained observation model. Moreover, BPR has been successfully applied to handle nonstationary opponents in multiagent systems [30], [31], [32], [33]. However, BPR algorithms have several limitations: 1) they usually use the episodic return as the observation signal that is only a scalar containing limited information, and the learning can be too slow as they must wait until the end of a full episode before updating the task belief; 2) they typically require numerous samples to estimate the probability distribution of the tabular-based observation model, which could be expensive and even infeasible to learn and maintain, especially using the state transition sample as the signal; and 3) they need to apply a fixed set of policies on all source tasks to obtain the tabular observation model in an offline manner, impeding the applicability to real-world continual learning settings.

We address the above limitations in the article. First, we employ the state transition sample (s, a, r, s'), the four-tuple composed of state, action, reward, and the next state, as the observation signal that reveals more task information since the state transition function $\mathcal{P}(s', r|s, a)$ completely characterizes the task's dynamics. Then, we use only a small number of samples to fit the state transition functions of source tasks, which are utilized to compute the scalable observation model that can generalize to any signals observed in the target task. In particular, we adopt two typical approaches, the nonparametric Gaussian process (GP) and the parametric neural network (NN), to estimate these functions, while in principle any distribution matching technique or probabilistic model can be adopted for this estimation. GP exhibits better sample efficiency and uncertainty measurements on the predictions, while NN can scale to extremely high-dimensional tasks. Finally, although there are several BPR+ algorithms [30], [31], [32], [33], [34] that can incorporate new models online, they still need to retrain the observation model offline when adding a new source policy into the library. Instead, we extend the offline-mode BPR to continual learning settings in a truly online manner by expanding the scalable observation model in a plug-and-play fashion when incorporating new source policies. Experimental results show that our method achieves more efficient policy reuse, and realizes effective continual learning with avoiding negative transfer.

In summary, our main contributions include: 1) employing the state transition sample as the observation signal and proposing a scalable observation model to achieve efficient policy reuse in DRL compared to the basic BPR algorithms [29], [30], [31], [32], [33], [34] and 2) extending the offline BPR to continual learning in an online manner compared to BPR+ algorithms [30], [31], [32], [33], [34], which avoids negative transfer and realizes better applicability to real-world applications.

The remainder of the article is organized as follows. Section II introduces the background including preliminaries of BPR and the related work. Section III presents our method in detail. Section IV shows the experimentation, and Section V presents concluding remarks and future work.

II. Background

A. Preliminaries of BPR

BPR provides an efficient framework for an agent to perform well by selecting the most appropriate policy from the policy library Π to reuse when facing a target task. Formally, a task $\tau \in \mathcal{T}$ is defined as a Markov decision process (MDP), and a policy $\pi \in \Pi$ outputs an appropriate action a given the state s. Then, the return is defined as the accumulated discounted reward $U = \sum_{i=1}^{M} \gamma^i r_i$, which is received from interacting with the environment under the guidance of policy π over an episode of M steps, where r_i is the instantaneous reward, and γ is the discount factor. BPR uses an observation model $\mathrm{P}(\sigma|\tau, \pi)$, which is a probability distribution over the observation signal $\sigma \in \Sigma$, to describe possible results when policy π behaves on task τ. The belief model $\beta(\mathcal{T})$, which is a probability distribution over $\mathcal{T}$, describes the similarity between the target task τ_0 and the source tasks $\mathcal{T}$. BPR initializes the belief model $\beta^0(\mathcal{T})$ with a prior probability and updates it based on the observation model and the observation signal by using Bayes' rule

$$\beta^t(\tau) = \frac{\mathrm{P}(\sigma^t|\tau,\pi^t)\beta^{t-1}(\tau)}{\sum_{\tau'\in\mathcal{T}} \mathrm{P}(\sigma^t|\tau',\pi^t)\beta^{t-1}(\tau')}. \tag{1}$$

To trade-off between exploration and exploitation, BPR uses the probability of expected improvement based on the task belief to select policies. It reuses the most potential policy $\hat{\pi}$ in the policy library Π by maximizing the expected utility as

$$\hat{\pi} = \arg\max_{\pi\in\Pi} \sum_{\tau\in\mathcal{T}} \beta(\tau) \int_{U\in R} U\mathrm{P}(U|\tau,\pi)\mathrm{d}U \tag{2}$$

where $\mathrm{P}(U|\tau, \pi)$ is the performance model, a probability distribution over the utility U that describes how policy π behaves on the task τ. The notations used in the article are summarized in Nomenclature section.

B. Related Work

Knowledge transfer has received increasing attention recently and a wide variety of methods have been studied in the RL community [19], [35]. Lazaric et al. [36] transferred the state transition samples from the source to the target task by calculating the compliance (similarity) between tasks using the Bayes theorem. Brys et al. [27] provided an inter-task mapping based on state spaces to measure the similarity between two tasks, and transferred the value functions using reward shaping. Song et al. [37] transferred the value functions by measuring the distance between the source and the target tasks based on the expected models of the MDPs. Laroche and Barlier [38] estimated the reward function of the target task by reusing the experience instances of a source task. Mustafa et al. [39] presented an assured metacognitive RL-based autonomous control framework to learn to choose reward functions that satisfy desired specifications and achieve significant performance across a variety of circumstances. However, transfer methods based on value functions or samples usually rely on well-estimated models of the MDPs or some prior knowledge of the target task for similarity measurement, which

leads to high computational complexity and could be infeasible to transfer in practice.

Instead, the other kind of approach attempts to directly transfer policies learned in source tasks, which eliminates the requirement on the prior knowledge of the target task or the assumed models of the MDPs. Fernández et al. [40] proposed probabilistic policy reuse that transfers source policies to bias the agent's exploration strategy in the target task. Li and Zhang [41] developed an online method of optimally selecting source policies by formulating it as a multiarmed bandit (MAB) problem. Li et al. [42] proposed a context-aware multipolicy reuse approach by employing the option framework to select the most appropriate source policy according to some contexts (e.g., a subset of states). Yang et al. [43] formulated the multipolicy transfer as an option learning problem, which serves as a complementary optimization objective of policy learning in the target task. These policy reuse approaches focus on facilitating learning the optimal policy for the target task, other than a quick response to an unknown task or rapid convergence of the target policy. Nevertheless, it is often required for RL to act online and respond quickly, in terms of rapid convergence, to novel tasks in real-world scenarios such as applications involving interactions with humans.

Rosman et al. [29] contributed BPR to quickly select source policies from an offline library using the Bayesian inference based on some observation signals and a trained observation model. BPR prefers to quickly select a pre-learned policy from a fixed library, which does not guarantee that the appropriate policy will be learned. Especially when the agent is faced with new unknown tasks, it may even result in negative transfer [44]. Furthermore, BPR is successfully applied to handle nonstationary opponents in multiagent systems. Hernandez-Leal et al. [30] proposed an extension, BPR+, to enable online learning of new models when the learning agent detects that the current policies are not performing optimally. Zheng et al. [32], [34] proposed deep BPR+ by extending BPR+ with a NN. A rectified belief model using the NN approximator is introduced to achieve accurate policy detection, and a distilled policy network is proposed as the policy library to store and reuse policies efficiently. Gao et al. [33] investigated how to play with unknown opponents in bilateral negotiation games based on deep BPR+.

In contrast to the family of BPR algorithms, we use more informative observation signals such as the state transition sample, and we propose a scalable observation model to efficiently update the task belief. Although BPR+ algorithms can incorporate new models online, they still need to retrain the observation model in an offline manner when adding a new source policy into the library. Instead, our scalable observation model allows us to extend our method to continual learning settings conveniently in a truly online manner.

III. Our Method

In this section, we propose an improved BPR method that aims at more efficient policy transfer in DRL. First, we formulate the scalable observation model by fitting the state transition function from limited samples. Next, we describe in detail how to use GP and NN to estimate the observation model. Then, we extend BPR from the conventional offline mode to the continual learning setting, which can be established naturally using the scalable observation model. Finally, we present the detailed algorithm.

A. Scalable Observation Model

In general, BPR maintains an observation model from some observation signals to update a task similarity measure, i.e., a belief, over source tasks for positive policy reuse. Naturally, the choice of the observation signal is crucial, since it determines the granularity of the policy selection frequency and the effectiveness of the policy reuse. The most widely used observation signal in BPR is the episodic return U when fully executing an associated policy. However, using the episodic return as the observation signal has two weaknesses. One is that we must wait until the end of a full episode to obtain the signal. Some applications have very long episodes, so delaying observing the signal until the end of the episode is too slow. The other is that the episodic return is only a scalar that contains limited information. For example, two policies with large differences may receive similar episodic returns in the same task, or one policy can also obtain similar episodic returns in two tasks with large differences. Instead, we use the state transition sample (s, a, r, s') as the observation signal, which is supposed to reveal more task information since the state transition function $\mathcal{P}(s', r|s, a)$ completely characterizes the task's dynamics.

Suppose that we have n source tasks, and we train one source policy in each task. The observation signal could be accrued by the agent by storing the history of all state transition samples encountered during the execution of all n source policies in all n source tasks. The observation model $\mathrm{P}(\sigma|\tau, \pi)$, in this case, is a tabular-based empirical estimate of the expected state transition function of the MDPs, which could be expensive and even infeasible to learn and maintain. It requires a large amount of state transition samples to estimate the probability distribution of the observation model, and will encounter the "curse of dimensionality" in large or continuous state-action spaces. Additionally, this may not generalize well, especially in cases with sparse sampling. When encountering a new sample that has not been recorded by the observation model, the Bayesian update of the task belief can easily be inaccurate in the policy reuse phase. Since the performance of BPR highly depends on the inference of the task belief, a scalable approach is necessary for building the observation model using limited state transition samples as the signal.

To address the above challenges, we propose a scalable observation model based on fitting the state transition functions of sources tasks $\mathcal{P}_j(s', r|s, a)$ from a small number of state transition samples $\mathcal{D}_j = \{(s_i^j, a_i^j, r_i^j, s'^j_i)\}_{i=1}^{N_j}$, where N_j is the number of samples in the jth source task. For the convenience of expression, we denote the input and output of this supervised learning instance of fitting the state transition function as $x = (s, a)$ and $y = (r, s')$. Assume that we have

learned the state transition functions of the n source tasks as $\{\mathcal{P}_j(y^j|x^j)\}_{j=1}^n$. Then, in the target task, we obtain the observation signal, i.e., a couple of state transition samples $\sigma^t = \{(x_i^0, y_i^0)\}_{i=1}^{N_0}$, by applying a selected policy π_t on the target task, where N_0 is the number of samples in the target task. The observation model $\mathrm{P}(\sigma|\tau, \pi)$ can be naturally constructed using the fit task dynamics as

$$\mathrm{P}(\sigma^t|\tau_j, \pi_t) = \Pi_{i=1}^{N_0} \mathcal{P}_j\left(y_i^0|x_i^0\right), \quad j = 1, \ldots, n. \tag{3}$$

Intuitively, $\mathcal{P}_j(y_i^0|x_i^0)$ indicates the likelihood of the learned task dynamics $\mathcal{P}_j$ fitting the sample (x_i^0, y_i^0), or the degree of samples in the target task τ_0 resembling those in the source task τ_j. We only need to store a small number of samples to learn the state transition functions of source tasks, which are utilized to compute the scalable observation model that can generalize to any signals observed in the target task.

B. Estimation of Observation Model

Since $\{\mathcal{P}_j\}_{j=1}^n$ are unknown, we only have access to an estimation of the observation model in (3). To obtain an approximation of the unknown densities, we employ two typical supervised learning approaches, the nonparametric GP [45] and the parametric NN [46], to fit the state transition function for each source task. These two methods are suitable for different situations with respective advantages.

First, we adopt GP due to its sample efficiency and the ability to provide uncertainty measurements on the predictions. GP is a nonparametric and Bayesian approach to regression that has been successfully adopted in many existing works in the RL community [47], [48], [49], [50]. For example, PILCO [48] used the GP to approximate the state transition function, which was employed to assist the long-term planning and policy evaluation in the context of model-based RL.

Given a test point x_*, the jth GP returns a Gaussian distribution over the output's mean, i.e., $\hat{y}_* \sim \mathcal{N}(\mu_{\mathrm{GP}_j}(x_*), \mathrm{cov}_{\mathrm{GP}_j}(x_*))$, as

$$\begin{aligned} \mu_{\mathrm{GP}_j}(x_*) &= \kappa_{*j}\left(\kappa_{jj} + \varepsilon^2 I\right)^{-1} Y_j \\ \mathrm{cov}_{\mathrm{GP}_j}(x_*) &= \kappa_{**} - \kappa_{*j}\left(\kappa_{jj} + \varepsilon^2 I\right)^{-1} \kappa_{j*} \end{aligned} \tag{4}$$

where (X_j, Y_j) are the training samples $\{(x_i^j, y_i^j)\}_{i=1}^{N_j}$ from the jth source task, and κ is the kernel function such that $\kappa_{jj} = \kappa(X_j, X_j)$, $\kappa_{*j} = \kappa(x_*, X_j)$, $\kappa_{j*} = \kappa(X_j, x_*)$, $\kappa_{**} = \kappa(x_*, x_*)$. The most commonly used kernel function is the Gaussian kernel, also known as the radial basis function (RBF)

$$\kappa\left(x, x'\right) = \delta^2 \exp\left(-\frac{\left\|x - x'\right\|_2^2}{2l^2}\right) \tag{5}$$

where δ and l are the hyper-parameters of the RBF kernel, representing its signal variance and characteristic length scale. For more commonly used kernels, please refer to [45]. Considering each sample as an independent Gaussian distribution, the observation model in (3) can be re-arranged using the n fit GPs $\{\mathrm{GP}_j\}_{j=1}^n$ as

$$\mathrm{P}\left(\sigma^t|\tau_j, \pi_t\right) = \Pi_{i=1}^{N_0} \mathcal{N}\left(y_i^0; \mu_j\left(x_i^0\right), \mathrm{cov}_j\left(x_i^0\right) + \varepsilon_{\mathrm{GP}}^2\right) \tag{6}$$

where μ_j and cov_j denote μ_{GP_j} and $\mathrm{cov}_{\mathrm{GP}_j}$ for simplicity, and $\varepsilon_{\mathrm{GP}}^2$ is a constant additional variance of the Gaussian distribution, which is used to enhance generalization of GP.

Second, we adopt NN as another technique to fit the state transition function since it can scale to extremely high-dimensional problems. Having been widely investigated in supervised learning tasks, NN can extract useful information from very large data sets to build extremely complex models. Naturally, we parameterize the state transition function of a given source task as

$$\hat{Y}_j = g_{\vartheta_j}(X_j) \tag{7}$$

where $g_{\vartheta_j}(\cdot)$ is a NN parameterized by weights ϑ_j, and $\hat{Y}_j$ is the predicted output of the network. Each NN is trained in a supervised learning way to minimize the loss function, e.g., the mean squared error (MSE), as

$$\mathcal{L}(\vartheta_j) = (Y_j - \hat{Y}_j)^2. \tag{8}$$

With the n fit NNs, the observation model in (3) represents each sample as an independent Gaussian distribution, such that

$$\mathrm{P}\left(\sigma^t|\tau_j, \pi_t\right) = \Pi_{i=1}^{N_0} \mathcal{N}\left(y_i^0; g_{\vartheta_j}\left(x_i^0\right), \varepsilon_{\mathrm{NN}}^2\right) \tag{9}$$

where the preset constant $\varepsilon_{\mathrm{NN}}^2$ is the variance of the Gaussian distribution.

For the sake of simplicity, let $f_j(x_i^0)$ and ξ_j^2 denote the mean and variance of the Gaussian distribution in the observation model, respectively. Then, we can obtain a uniform formation for the observation models in (6) and (9) as

$$\mathrm{P}\left(\sigma^t|\tau_j, \pi_t\right) = \Pi_{i=1}^{N_0} \mathcal{N}\left(y_i^0; f_j\left(x_i^0\right), \xi_j^2\right), \quad j = 1, \ldots, n. \tag{10}$$

With the estimated observation model, the task belief β^t in (1) can be efficiently updated using the Bayes' rule as

$$\beta^t(\tau_j) = \frac{\prod_{i=1}^{N_0} \mathcal{N}\left(y_i^0; f_j\left(x_i^0\right), \xi_j^2\right) \beta^{t-1}(\tau_j)}{\sum_{j'=1}^{n} \prod_{i=1}^{N_0} \mathcal{N}\left(y_i^0; f_{j'}\left(x_i^0\right), \xi_{j'}^2\right) \beta^{t-1}(\tau_{j'})}. \tag{11}$$

C. Extension to Continual Learning

The conventional BPR algorithm is performed in an *offline* manner. It has to acquire a collection of pre-learned behaviors in advance, and also needs to apply a fixed set of policies on all source tasks to obtain a tabular-based observation model offline. Nevertheless, when encountering a new task, it is likely that none of the policies in the library is suitable, which leads to negative transfer. Although the BPR+ algorithms can incorporate new models online, when adding a new source policy into the library, they still need to re-estimate the observation model by applying all source policies on the new task and applying the new policy on all tasks. This can be a big barrier to the practicality of the algorithm due to the expensive and inefficient update of the observation model. Moreover, different from the learned policies, the source tasks may be inaccessible in the new learning process of the target task, in which case the observation model is infeasible to update.

To achieve artificial general intelligence, RL agents should constantly build more complex skills and scaffold their knowledge about the world in a *continual learning* manner. In the

article, we extend our method to the continual learning setting such that in the policy reuse phase, the learning agent can consult the stored library first, and either retrieve the most suitable policy from the library or expand a new policy into the library. When incorporating a new policy, we only need another GP or NN to fit the state transition function of the corresponding new source task $\mathcal{P}_{n+1}$ using a small number of samples from that task, independent of the policies and fit GPs or NNs from the existing source tasks. The proposed observation model in (10) is naturally scalable to the continual expansion of a new source task in a plug-and-play fashion as

$$\mathrm{P}\left(\sigma^t|\tau_{n+1},\pi_t\right)=\Pi_{i=1}^{N_0}\mathcal{N}\left(y_i^0;\, f_{n+1}\left(x_i^0\right),\xi_{n+1}^2\right). \tag{12}$$

Note that the conventional BPR also maintains a performance model, i.e., a distribution of returns from each policy on the source tasks, which is utilized together with the task belief to select the most appropriate policy for reusing. As we observe the state transition function that completely characterizes the task's dynamics, the scalable observation model is informative to capture the task similarity effectively. We can merely rely on the task belief in (11) to choose reuse policies, and hence we abandon the use of the performance model for better applicability in the continual learning setting.

D. Algorithm

Based on the above statements, we present the general form of our method in Algorithm 1. It mainly consists of two phases: the reuse phase that selects the most appropriate policy from the library in Lines 2–13, and the learning phase which expands a new optimal policy into the library in Lines 14–17.

First, the policy reuse phase is performed when facing the target task. At update step t (the step size is N_0), the agent selects a reuse policy π^t from the library $\{\pi_j\}_{j=1}^n$ according to the task belief β^{t-1}, and receives the observation signal σ^t after applying the selected policy π^t on the target task τ_0. Next, the scalable observation model $\mathrm{P}(\sigma^t|\tau_j,\pi^t)$ in (10) can be efficiently estimated by applying the obtained signal σ^t on the fit GPs or NNs from the source tasks, which is used to update the task belief in (11). The two processes are iteratively alternated within a learning episode until the goal or the maximum number of steps is reached. At the end of the episode, we check whether the target task is sufficiently different from all source tasks according to the average return $\bar{U}$ over k episodes. If $\bar{U}$ is below a preset threshold, the agent will switch to the learning phase in subsequent episodes where a new optimal policy π_{n+1} is learned for the target task using any DRL algorithm. The obtained new policy π_{n+1} is expanded into the policy library, together with the target task τ_0 being extended as a new source task τ_{n+1}. Accordingly, a new GP or NN is generated to fit the dynamics of the new source task using a small number of samples collected from the learning phase. Specifically, for GP, we store a small number of samples for the new source task to calculate the probability distribution of the observation signals when fitting the new GP. For NN, we need to use the collected samples to train a new NN model to fit the state transition function for the new source task. Finally, the agent switches back to the reuse phase for the next policy transfer problem.

Algorithm 1 Efficient BPR With Scalable Observation Model

Input: Source task set $\mathcal{T}$, source policy library Π and GPs/NNs, number of episodes K, the maximum number of steps M.

1 Initialize belief $\beta^0(\mathcal{T})$ with a uniform distribution
2 **if** *execute a reuse phase* **then**
3 **while** *episode* $\leq K$ **do**
4 **while** *step* $\leq M$ & *not reaching the goal* **do**
5 Select a policy $\pi^t \in \Pi$ based on β^{t-1}
6 Apply π^t on target task τ_0 and receive observation signal σ^t
7 Estimate $\mathrm{P}(\sigma^t|\mathcal{T},\pi^t)$ in (10) by applying σ^t on source GPs or NNs
8 Update task belief β^t in (11) using the estimated $\mathrm{P}(\sigma^t|\mathcal{T},\pi^t)$
9 **end**
10 **if** *a sufficiently different task is detected* **then**
11 Switch to the learning phase
12 **end**
13 **end**
14 **else if** *execute a learning phase* **then**
15 Learn a new policy π_0 and fit a new GP/NN for τ_0
16 Expand π_0 as π_{n+1} into the library
17 Switch to the reuse phase
18 **end**

Note that in some practical applications, we can empirically choose (s,a,r) or (s,a,s') alone as the observation signal for ease of use, in which case the output y of the task dynamics function in (3) is the reward r or the next state s'. Furthermore, N_0 determines the granularity of the policy selection frequency, in this article, we update the policy reusing strategy every time step for better sample efficiency, i.e., $N_0=1$. In summary, our algorithm estimates the observation model from limited samples to realize efficient policy transfer. Meantime, using the scalable observation model, our algorithm can be easily extended to the continual learning setting, which makes it more practical in real-world scenarios.

IV. Experiments

In this section, we evaluate our method on five different domains with increasing levels of complexity. Through these experiments, we aim to build problem settings that are representative of the types of transfer learning that RL agents may encounter in real-world scenarios. Our experiments are mainly divided into three parts.

1) First, we implement our GP-based method in three relatively simple domains to demonstrate that it can achieve efficient policy transfer when faced with target tasks that are similar to source tasks. The results are shown in Section IV-A.
2) Second, we implement our NN-based method in two high-dimensional complex domains from MuJoCo, to verify that our method enables efficient policy transfer

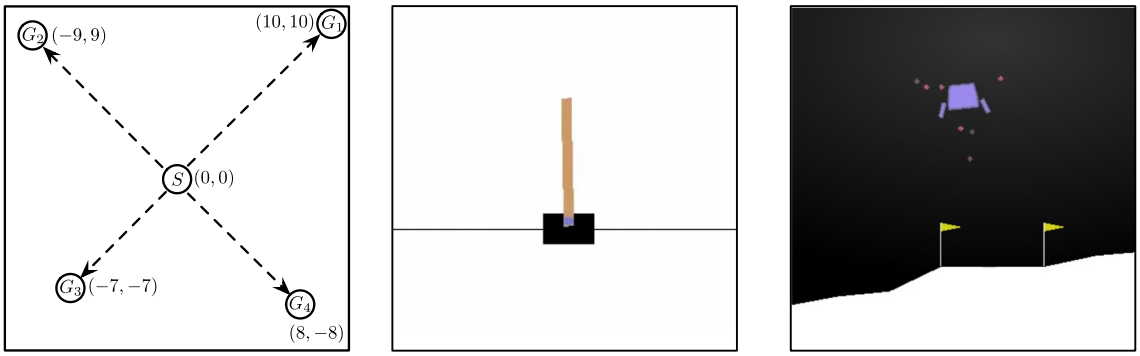


Fig. 1. Three types of dynamic environments. (a) 2-D navigation domain, S is the start point and G is the goal point. (b) CartPole domain. (c) LunarLander domain.

for more sophisticated tasks. The results are shown in Section IV-B.

3) Finally, we apply our method to all domains to prove that compared to the basic BPR algorithm, it can achieve continual learning and thus avoid negative transfer when faced with target tasks that are very different from the source tasks, as shown in Section IV-C.

In Sections IV-A and IV-B, we compare our method to three baseline approaches for transferring policies: the family of BPR algorithms using episodic return as the observation signal [29], [34], the probabilistic policy reuse with DRL (PR-DRL) algorithm [26], [40], [51], and the optimal policy selection with DRL (OPS-DRL) algorithm [41]. In Section IV-C, we evaluate our method in comparison to the basic BPR algorithm [29], and the settings of the BPR algorithm are consistent with those in Sections IV-A and IV-B. For a fair comparison, we make some improvements to the baselines so that they can be directly compared with our method. The details of the three baselines are given in the Appendix.

All experimental results are averaged over ten trials. The shaded area represents the 95% confidence interval for evaluation curves, and the standard errors are presented for numerical results. All the algorithms are implemented with Python 3.5 running on Ubuntu 16 with 48 Intel Xeon E5-2650 2.20 GHz CPU processors, 193-GB RAM, and an NVIDIA Tesla GPU of 32-GB memory.

A. Results for Efficient Policy Transfer Based on GP

In the experiment of this section, we choose three representative types of dynamic environments, as shown in Fig. 1, and the details are as follows.

1) 2-D Navigation: We first consider the navigation domain where an agent must move to a goal position in a 2-D surface, which is a continuous-state and continuous-action problem. The states are the current 2-D positions and the actions are 2-D vectors clipped to be in the range of $[-1, +1]$. Episodes terminate when the agent is within 0.5 of the goal or reaches the maximum number of steps $M = 100$. The reward is the negative Euclidean distance to the goal minus a controlled cost that is positively related to the scale of actions. We consider four source tasks, where they have the same starting points $(0, 0)$ and different goal positions as $(10, 10)$, $(-9, 9)$, $(-7, -7)$, and $(8, -8)$, respectively. In this domain, tasks only differ in reward functions. Therefore, we employ the tuple (s, a, r) as the observation signal to fit the GP and estimate the scalable observation model.

2) CartPole: We next consider a classical control problem with a continuous state space and a discrete action space, described in [52] and implemented experiments in OpenAI Gym [53]. The states are 4-D vectors, and the actions are two discrete values of 0 and 1. Episodes terminate when the pole falls or reaches the maximum number of steps $M = 100$. In order to encourage the agent to balance the pole, a positive reward of $+1$ is given at every step when the angle between the pole and vertical line is smaller than a small threshold. Otherwise, the reward is 0. The system is controlled by applying a constant force $F = 10$ newtons to the cart, and the agent can apply full force to the cart in either direction, i.e., two possible actions. We set up two source tasks by adding a constant force F' with a fixed direction on the cart. The agent needs to balance the pole with the interference of $F' = 5$ newtons in one source task, and with $F' = -5$ in the other task. In this domain, the tuple (s, a, s') is set as the observation signal to infer the most appropriate policy for the target task.

3) LunarLander: We choose a relatively complex task, the LunarLanderContinuous-v2 domain from OpenAI-Gym, which involves landing a spacecraft safely on a lunar surface. This is a high-dimensional continuous control problem with sparse reward, and is representative of real-world problems where it is considerably difficult to learn accurate dynamics. The states are 8-D vectors. The actions are 2-D vectors clipped to be in the range of $[-1, +1]$, which are used to control the powers of the main and side engines. Episodes finish if the lander crashes or comes to rest or reaches the maximum number of steps $M = 1000$. In this case, it can take a large number of steps to reach the goal, and the reward is significantly delayed until the end of the long episode. We set up three source tasks that represent three typical scenarios by applying an additional constant power with a fixed direction on the main engine of the spacecraft. The additive powers

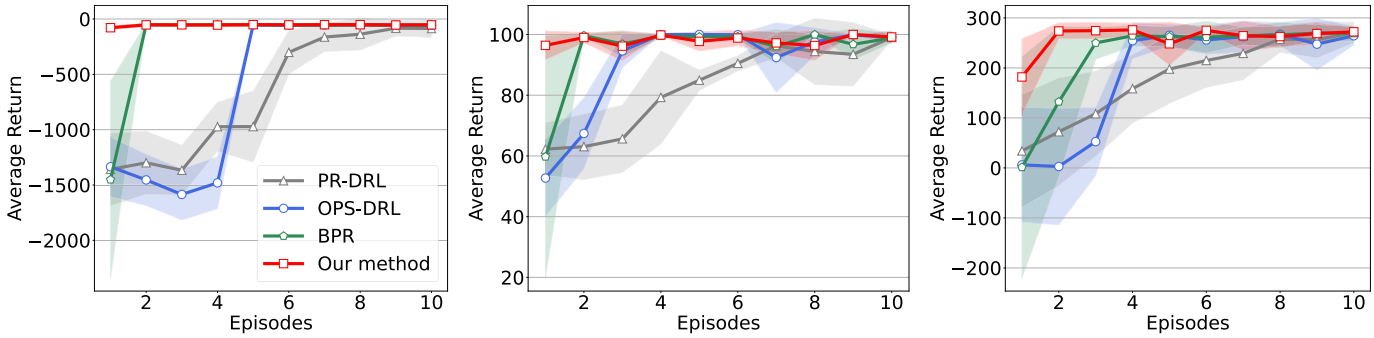


Fig. 2. Average return per episode of all tested methods in: (a) 2-D navigation; (b) CartPole; and (c) LunarLander.

TABLE I
AVERAGE RETURN OVER ALL EPISODES OF ALL TEST METHODS IMPLEMENTED IN THREE DOMAINS

Method	PR-DRL	OPS-DRL	BPR	Our method
2-D Navigation	-674.19 ± 581.44	-618.02 ± 709.72	-193.39 ± 505.16	$\mathbf{-55.33 \pm 16.53}$
CartPole	82.88 ± 16.41	90.36 ± 17.16	94.59 ± 17.46	$\mathbf{98.08 \pm 3.53}$
LunarLander	179.88 ± 105.79	187.87 ± 126.69	224.09 ± 121.28	$\mathbf{259.75 \pm 41.98}$

are $(0.5, 0)$, $(-0.5, 0)$, and $(0, 0.5)$, respectively. A successful transfer agent should learn to transfer skills from the source tasks and try to avoid the potential risk of landing the spacecraft. Here, the moon's surface is generated randomly in each episode, so the environmental dynamics are learned on noisy data. To solve this problem, during the experiment, we set up a constant random seed for the environment. In this setting, tasks mainly differ in the state transition functions. Therefore, we can employ the tuple (s, a, s') as the observation signal to infer the task belief.

In this experiment, we select some target tasks that are relatively close to the source tasks in each domain, which ensures that there are suitable policies for the target tasks in the policy library.

1) In 2-D Navigation domain, we select 12 target tasks, and their goal positions are $(10.5, 10)$, $(10, 9.5)$, $(-8.5, 9)$, $(-9, 9.5)$, $(-6.5, -7)$, $(-7, -7.5)$, $(7.5, -8)$, $(8, -7.5)$, $(10, 10)$, $(-9, 9)$, $(-7, -7)$, and $(8, -8)$.
2) In CartPole domain, we select six target tasks, and their values of the additional force F' are set as 4.5, 5, 5.5, -5.5, -5, and -4.5.
3) In LunarLander domain, we select nine target tasks, and their value of additive powers are $(0.45, 0)$, $(-0.45, 0)$, $(0, 0.45)$, $(0.55, 0)$, $(-0.55, 0)$, $(0, 0.55)$, $(0.5, 0)$, $(-0.5, 0)$, and $(0, 0.5)$.

In each experiment, the number of learning episodes in the target task, K, is set as 10, which is supposed to be sufficient for all tested methods to converge to the most appropriate policy with the highest return. The hyperparameters are set as: $\delta = 1$ and $l = 2$ for the RBF kernel of the GPR, and $\varepsilon_{\text{GP}}^2 = 0.1$ for the Gaussian distribution of all test domains. And in this experiment, we use vanilla policy gradient (REINFORCE) [54], [55] and twin delayed deep deterministic policy gradient (TD3) algorithm [56] to learn optimal policies for source tasks.

We present the primary results of our GP-based method and all baseline approaches on three relatively simple experimental domains, and all the results are averaged over multiple target tasks. Fig. 2 shows the average return per learning episode, and Table I reports the numerical results in terms of average return over all learning episodes. BPR obtains better policy transfer performance than PR-DRL and OPS-DRL, which is supposed to benefit from leveraging the efficiency of the Bayesian inference framework. It is clear that our method achieves much more efficient policy transfer in all tasks compared to baselines. From Fig. 2(a) and (b), it is observed that the received return of our method in the first episode is nearly equal to that of the best policies for target tasks, since our method can converge to the most appropriate policy using a few steps within the first episode. In more complex domains, our method can also achieve much better jump-start performance compared to all baselines, as illustrated in Fig. 2(c). Furthermore, from the numerical results in Table I, our method generally obtains the largest average return over all learning episodes in all domains. In addition, it can be observed from the statistical results that our method mostly obtains smaller confidence intervals and standard errors than the baselines. This phenomenon indicates that our method can provide more stable source task selection and knowledge transfer.

We also study how the number of state transition samples (s, a, r, s') used for fitting the GPs affects the performance of our method, i.e., identifying the relationship between the sample size and the accuracy of policy detection during the policy reuse phase in our method. We use 100, 200, 500, 1000, and 2000 samples, respectively, for fitting state transition functions using GPs in these domains to observe the performance of our method. Fig. 3 shows the average

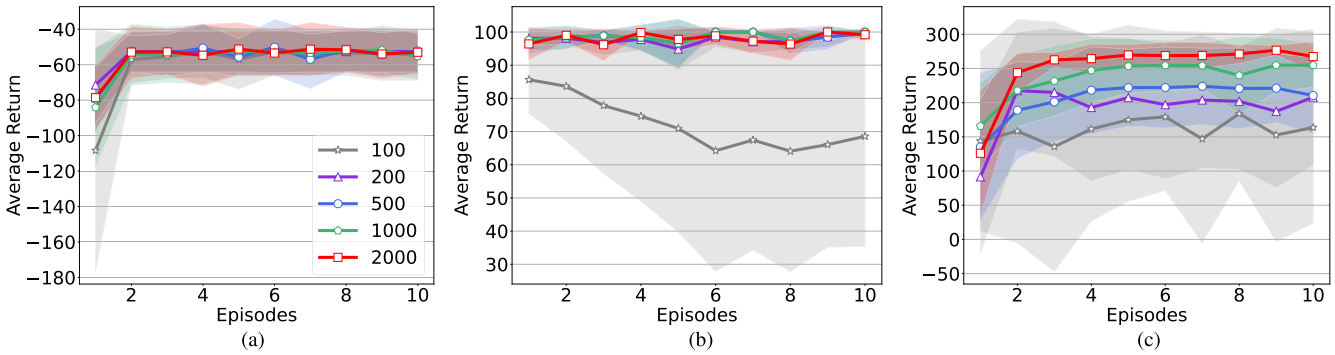


Fig. 3. Average return per episode of different sample sizes used for GPs in: (a) 2-D navigation; (b) CartPole; and (c) LunarLander.

TABLE II
AVERAGE RETURN OVER ALL EPISODES OF DIFFERENT SAMPLE SIZES USED FOR GPS IN THREE DOMAINS

Sample size	100	200	500	1000	2000
2-D Navigation	-59.87 ± 30.06	$\mathbf{-54.46 \pm 14.09}$	-55.75 ± 16.01	-56.53 ± 19.33	-55.33 ± 16.53
CartPole	72.31 ± 29.62	97.65 ± 3.54	98.53 ± 3.72	$\mathbf{98.70} \pm 3.50$	98.08 ± 3.53
LunarLander	186.43 ± 105.72	192.43 ± 125.87	206.37 ± 70.66	237.42 ± 53.50	$\mathbf{251.97 \pm 54.57}$

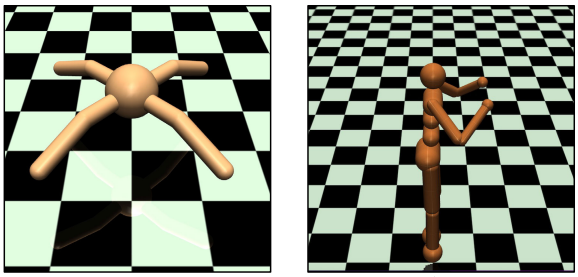


Fig. 4. Two types of dynamic environments. (a) Ant-Navigation domain. (b) Humanoid-Navigation domain.

return per learning episode, and Table II reports the numerical results in terms of average return over all learning episodes. In all domains, our method obtains a significant performance improvement when the sample size goes from 100 to 200. Then, as the sample size continues to grow, the performance of our method is slightly improved only, and the degree of performance improvement is decreasing. Especially in the 2-D Navigation and CartPole domains, the performance of our method remains roughly the same. It demonstrates that by using only a small number of samples to fit the state transition function, our method is capable of accurately estimating the scalable observation model and inferring the task belief for efficient policy transfer.

B. Results for Efficient Policy Transfer Based on NN

In the experiment of this section, we choose two high-dimensional dynamic environments from MuJoCo [57], as shown in Fig. 4, and the details are as follows.

1) Ant-Navigation: The first experiment consists of a variation of the Ant-v3, which makes an ant agent reach the specified 2-D positions. The states are 113-D vectors, and the actions are 8-D vectors clipped to be in the range of $[-1, +1]$. In our experiment, the ant agent only needs to be within 0.2 of the goal from the starting point or reach the maximum number of steps $M = 100$ regardless of speed. The reward consists of three parts: the negative Euclidean distance to the goal, a controlled cost that is positively related to the scale of actions, and a contact cost that is positively related to the scale of contact forces. We set up four source tasks with different goal positions and gears on the legs of the ant agent, where the gear is used to specify 3-D force and torque axes by scaling the length of the actuator. The goal positions are $(1, 1)$, $(-1, 1)$, $(-1, -1)$, $(1, -1)$, and the corresponding gears are 50, 100, 150, 200, respectively. When faced with target tasks similar to the source ones, we need to select the most appropriate policy from the library. To solve this problem, the tuple (s, a, r, s') is used as the observation signal.

2) Humanoid-Navigation: We design the fifth experiment to verify that our method can solve very high-dimensional problems. We adopt a variation of the Humanoid-v3, which makes a humanoid agent reach the specified 2-D position. It is very challenging to solve this high-dimensional problem with a continuous state-action space. The states are 378-D vectors, and the actions are 17-D vectors clipped to be in the range of $[-0.4, +0.4]$. In this experiment, the humanoid agent will end the episode when it reaches the specified range of the goal or the maximum number of steps $M = 1000$, or it falls. The reward contains four parts: the negative Euclidean distance to the goal, an alive bonus when the z-coordinate of the agent is in the specified range, a large bonus when the agent reaches its goal, and a control and impact cost. We set up four source tasks where the goal positions are $(0.6, 0.6)$, $(-0.55, 0.55)$, $(0.5, -0.5)$, and $(-0.45, -0.45)$, and the corresponding specified ranges are 0.4, 0.35, 0.3, and 0.25, respectively. To solve this problem, we use the tuple (s, a, r) as the observation signal.

Analogous to Section IV-A, we select some target tasks that are relatively close to the source tasks in each domain.

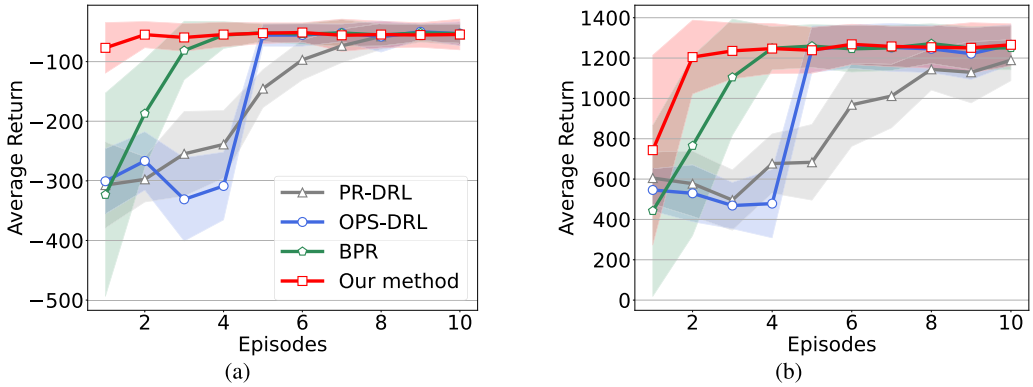


Fig. 5. Average return per episode of all tested methods in: (a) Ant-Navigation and (b) Humanoid-Navigation.

TABLE III
AVERAGE RETURN OVER ALL EPISODES OF ALL TEST METHODS IMPLEMENTED IN TWO DOMAINS

Method	PR-DRL	OPS-DRL	BPR	Our method
Ant-Navigation	-157.79 ± 109.36	-153.05 ± 128.53	-96.46 ± 107.31	$\mathbf{-56.94 \pm 25.78}$
Humanoid-Navigation	848.15 ± 295.14	950.91 ± 384.62	1108.84 ± 354.02	$\mathbf{1196.59 \pm 243.20}$

1) In Ant-Navigation domain, we select 12 target tasks, and their goal positions and gears are $(1.1, 1.1, 60)$, $(0.9, 0.9, 40)$, $(1.1, -1.1, 90)$, $(0.9, -0.9, 110)$, $(-1.1, 1.1, 160)$, $(-0.9, 0.9, 140)$, $(-1.1, -1.1, 190)$, $(-0.9, -0.9, 210)$, $(1, 1, 50)$, $(1, -1, 100)$, $(-1, 1, 150)$, and $(-1, -1, 200)$.
2) In Humanoid-Navigation domain, we select 12 target tasks, and their goal positions and specified ranges are $(0.55, 0.55, 0.4)$, $(0.65, 0.65, 0.4)$, $(-0.5, 0.5, 0.35)$, $(-0.6, 0.6, 0.35)$, $(0.45, -0.45, 0.3)$, $(0.55, -0.55, 0.35)$, $(-0.5, -0.5, 0.25)$, $(-0.4, -0.4, 0.25)$, $(0.6, 0.6, 0.4)$, $(-0.55, 0.55, 0.35)$, $(0.5, -0.5, 0.3)$, and $(-0.45, -0.45, 0.25)$.

In this experiment, the number of learning episodes in the target task, K, is set as 10. The hyper-parameters are set as: $\varepsilon^2_{\text{NN}} = 0.1$ for the Ant-Navigation domain and $\varepsilon^2_{\text{NN}} = 1$ for the Humanoid-Navigation domain. To solve complex problems, we use the twin delayed deep deterministic policy gradient (TD3) algorithm [56] to learn optimal policies for source tasks.

We present the primary results of our NN-based method and all baseline methods on two complex experimental domains, and all the results are averaged over multiple target tasks. Fig. 5 shows the average return per learning episode, and Table III reports the numerical results in terms of average return over all learning episodes. BPR obtains better policy transfer performance than PR-DRL and OPS-DRL, and our method achieves much more efficient policy transfer in all tasks compared to BPR. In domains with high-dimensional, our method can also achieve much better jump-start performance compared to all baselines, as illustrated in Fig. 5(a) and (b). Furthermore, from the numerical results in Table III, our method generally obtains the largest average return over all learning episodes in all domains. In addition, it can be observed from the statistical results that our method mostly obtains smaller confidence intervals and standard errors than the baselines. This phenomenon indicates that our method can provide more stable source task selection and knowledge transfer in high-dimensional domains.

Moreover, we study how the number of state transition samples (s, a, r, s') used for fitting the NNs affects the performance of our method. We use 1000, 5000, 10 000, 50 000, and 100 000 samples, respectively, for fitting the state transition functions using NNs in all domains to observe the performance of our method. Fig. 6 shows the average return per learning episode, and Table IV reports the numerical results in terms of average return over all learning episodes. In all domains, as the sample size continues to grow, the performance of our method is improved, but the degree of performance improvement is decreasing. When the sample size reaches 50 000, the performance is almost optimal. It demonstrates that using the limited number of samples to fit the state transition function, our method based on NN is capable of accurately estimating the scalable observation model and inferring the task belief for efficient policy transfer.

Overall, it is verified that our method achieves a more accurate inference of the task belief and converges more quickly to the most appropriate policy for the target task. Using the scalable observation model with more informative observation signals, our method can efficiently update the task belief and achieve better performance compared to all baselines in all experimental domains.

C. Results for Continual Learning

In this experiment, we evaluate our method and the baseline approach in continual learning settings where the agent is faced with a new unknown target task that largely differs from any of the source tasks.

1) In the 2-D Navigation domain, we select four target tasks, and their goal positions are $(0, 10)$, $(0, -9)$, $(-8, 0)$, and $(9, 0)$.

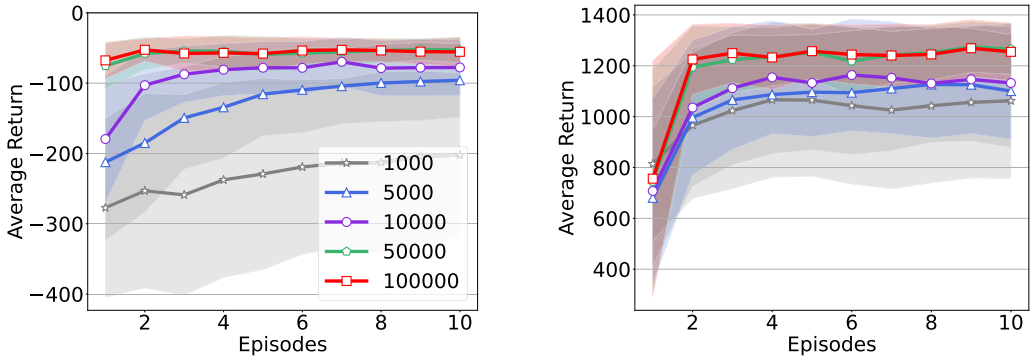


Fig. 6. Average return per episode of different sample sizes used for NNs in: (a) Ant-Navigation and (b) Humanoid-Navigation.

TABLE IV
AVERAGE RETURN OVER ALL EPISODES OF DIFFERENT SAMPLE SIZES USED FOR NNS IN TWO DOMAINS

Sample size	1000	5000	10000	50000	100000
Ant-Navigation	-230.84 ± 132.05	-130.40 ± 81.34	-91.06 ± 56.40	-57.28 ± 22.61	$\mathbf{-56.34 \pm 20.95}$
Humanoid-Navigation	1016.49 ± 311.26	1048.26 ± 273.24	1086.82 ± 276.53	1190.30 ± 234.84	$\mathbf{1197.40 \pm 236.57}$

TABLE V
FINAL RECEIVED RETURNS IMPLEMENTED IN ALL DOMAINS WITH CONTINUAL LEARNING SETTINGS

Domain	2-D Navigation	CartPole	LunarLander	Ant-Navigation	Humanoid-Navigation
BPR	-895.15 ± 292.44	34.0 ± 10.34	-33.25 ± 87.71	-288.05 ± 62.82	343.61 ± 110.79
Our method	$\mathbf{-39.82 \pm 5.39}$	$\mathbf{96.44 \pm 2.90}$	$\mathbf{-24.96 \pm 5.29}$	$\mathbf{-70.08 \pm 11.30}$	$\mathbf{1347.82 \pm 89.98}$

2) In the CartPole domain, we select two target tasks, and their values of the additional force F' are 8 and -8.
3) In the LunarLander domain, we select two target tasks, and their value of additive powers are $(0.5, 0.5)$ and $(-0.5, 0.5)$.
4) In the Ant-Navigation domain, we select four target tasks, and their goal positions and gears are $(1, 0, 50)$, $(-0.9, 0, 100)$, $(0, -0.8, 150)$, and $(0, 0.9, 200)$.
5) In the Humanoid-Navigation domain, we select four target tasks, and their goal positions and specified ranges are $(0.2, 0.7, 0.3)$, $(-0.65, 0.2, 0.3)$, $(0.25, -0.6, 0.3)$, and $(0, -0.8, 0.3)$.

Different from the previous settings, our method is required to learn a new policy for the target task in an online fashion, other than selecting an existing source policy from the offline library, when a sufficiently different target task is detected. In this setting, we compare our method to the BPR and we deploy the agents to some target tasks that are not close to the source ones, and the final received returns are shown in Table V. It is observed that BPR performs worse in the five domains since it is unable to select an appropriate policy from the offline library to respond to a new unknown task. In contrast, our method can learn a new optimal policy for the target task by expanding the source library in a continual learning manner. In the policy reuse phase, our method can detect the unknown tasks according to the received return, and switch to the learning phase to learn a new optimal policy for the target task, thus effectively avoiding negative transfer.

V. CONCLUSION AND FUTURE WORK

In the article, we proposed a general improved BPR framework to implement more efficient policy transfer in DRL, which can be implemented by any DRL algorithm, whether they are model-based or model-free, on-policy, or off-policy. We introduced a scalable observation model by using the nonparametric GP or parametric NN to fit the state transition function of source tasks in a model-based way, which achieves more efficient policy reuse with informative and instantaneous observation signals. GP is merited for its sample efficiency and the ability to provide uncertainty measurements on the predictions, while NN is preferred for extremely high-dimensional tasks. Moreover, we extended our method to continual learning settings conveniently in a plug-and-play fashion to avoid the negative transfer.

While we use GP and NN to estimate the observation model, our method is a general framework and can be easily combined with any distribution matching technique or probabilistic model. Thus, a potential direction for future work is to use more powerful techniques to estimate the observation model, such as more advanced GPs that can scale to high-dimensional domains [58], [59], [60], [61], or Bayesian NNs. In addition, we can also consider taking different kernel functions for states and actions according to specific situations. Alternatively, a potential solution is to use the return distribution of distributional RL approaches to avoid introducing any additional models. Another direction is to improve the settings of continual learning, for example, using policy distillation to

Algorithm 2 Probabilistic Policy Reuse With DRL

Input: Source policy library $\{\pi_j\}_{j=1}^n$, number of episodes K, the maximum number of steps M.

$W_j \leftarrow 0$, the reuse gain associated with π_j,
$V_j \leftarrow 0$, the number of times π_j is used.
Temperature parameter ν and the incremental size $\Delta\nu$
while *episode* $\leq K$ **do**
  while $j \leq n$ **do**
    $p_j \leftarrow e^{\nu \times \mathrm{W}_j} / \sum_{j'}^{n} e^{\nu \times \mathrm{W}_{j'}}$
  end
  Select the reuse policy π_j according P.
  $U \leftarrow 0$
  while *step* $\leq M$ & *not reaching the goal* **do**
    Apply π_j on target task τ_0 and receive the reward r_t.
    $U = U + \gamma^t r_t$
  end
  $W_j \leftarrow \frac{W_j \times V_j + U}{V_j + 1}$
  $V_j \leftarrow V_j + 1$
  $\nu = \nu + \Delta\nu$
end

Algorithm 3 Optimal Policy Selection With DRL

Input: Source policy library $\{\pi_j\}_{j=1}^n$, number of episodes K, the maximum number of steps M.

$W_j \leftarrow 0$, the reuse gain associated with π_j,
$V_j \leftarrow 0$, the number of times π_j is used.
while *episode* $\leq K$ **do**
  Select the reuse policy π_j according:
  $j = \arg\max_{1 \leq j \leq n} \left(W_j + \sqrt{\frac{2\ln\left(\sum_{j=1}^{n} V_j + 1\right)}{V_j + 1}} \right)$.
  $U \leftarrow 0$
  while *step* $\leq M$ & *not reaching the goal* **do**
    Apply π_j on target task τ_0 and receive the reward r_t.
    $U = U + \gamma^t r_t$
  end
  $W_j \leftarrow \frac{\mathrm{W}_j \times \mathrm{V}_j + U}{\mathrm{V}_j + 1}$
  $V_j \leftarrow V_j + 1$
end

speed up the learning process for unknown tasks. Furthermore, another significant and challenging direction is to provide some theoretical guarantees for the family of BPR algorithms such as the quantitative analysis of sample efficiency.

APPENDIX
BASELINES: BPR, PR-DRL, AND OPS-DRL

A. Bayesian Policy Reuse

The family of BRP algorithms [29], [30], [31], [32], [33], [34] typically uses the episodic return as the observation signal, and needs to apply all source policies on all source tasks to estimate the tabular-based observation model in an offline manner. However, it requires a large amount of episode return samples to estimate the probability distribution of the observation model. In our experiments, for better applicability, we employ a variant of the BPR algorithms, which models the probability distribution of the observation model as a Gaussian distribution. For each task-policy pair (τ, π), we apply the source policy π on the source τ, and repeat it one hundred times. Then, we take the mean of episode returns μ_U, and artificially choose an appropriate variance ε_U^2. In this way, we can obtain the observation model as $\mathrm{P}(\sigma \mid \tau, \pi) \sim \mathcal{N}(\mu_U, \varepsilon_U^2)$.

B. Policy Reuse with DRL

The probabilistic policy reuse algorithm [40] improves its exploration by exploiting the source policies probabilistically. It updates the probability depending on the reuse gains, which are obtained concurrently during the learning process. The original probabilistic policy reuse (PR) algorithm was implemented by the Q-learning, referred to as PRQ-learning, which can be applied to simple domains with a discrete state-action space only. Here, we adopt a version of PRQ-learning that builds on DRL, similar to the experimental settings in [26], [51]. Thus, the resulting approach is called PR-DRL which is directly comparable to our method. Algorithm 2 shows its pseudocode, where the initial value of the temperature parameter ν is 0, and the value of the incremental size $\Delta\nu$ is 0.05.

C. Optimal Policy Selection With DRL

The optimal policy selection (OPS) algorithm [41] formulates online source policy selection as a MAB problem and augments Q-learning with policy reuse. Analogous to the setting of PR-DRL, we adopt a version of OPS that builds on DRL and obtain another baseline approach OPS-DRL. Note that the original OPS algorithm initializes the reuse gains of source policies by applying all the source policies on the target task. For a fair comparison, we set the initial reuse gains of source policies in the OPS-DRL as zeros. The pseudocode is shown in Algorithm 3. Moreover, PR-DRL and OPS-DRL learn a new policy for the target task while reusing source policies from the policy library. For a fair comparison with our method, we assume that the two baselines only reuse source policies without learning a new one.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.
[2] M. Mazouchi, Y. Yang, and H. Modares, "Data-driven dynamic multiobjective optimal control: An aspiration-satisfying reinforcement learning approach," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 11, pp. 6183–6193, Nov. 2022.
[3] Y. Yang, B. Kiumarsi, H. Modares, and C. Xu, "Model-free λ-policy iteration for discrete-time linear quadratic regulation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 2, pp. 635–649, Feb. 2023.
[4] Y. Yang, M. Mazouchi, and H. Modares, "Hamiltonian-driven adaptive dynamic programming for mixed H_2/H_∞ performance using sum-of-squares," *Int. J. Robust Nonlinear Control*, vol. 31, no. 6, pp. 1941–1963, Apr. 2021.
[5] R. Chavarriaga, P. W. Ferrez, and J. D. R. Millán, "To err is human: Learning from error potentials in brain-computer interfaces," IDIAP Res. Inst., Martigny, Switzerland, Tech. Rep., 2007.

[6] Z. Wang, C. Chen, H. Li, D. Dong, and T. Tarn, "Incremental reinforcement learning with prioritized sweeping for dynamic environments," *IEEE/ASME Trans. Mechatronics*, vol. 24, no. 2, pp. 621–632, Apr. 2019.

[7] H. Li, Q. Zhang, and D. Zhao, "Deep reinforcement learning-based automatic exploration for navigation in unknown environment," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 6, pp. 2064–2076, Jun. 2020.

[8] Z. Wang, C. Chen, and D. Dong, "Lifelong incremental reinforcement learning with online Bayesian inference," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 8, pp. 4003–4016, Aug. 2022.

[9] Q. Wei, H. Ma, C. Chen, and D. Dong, "Deep reinforcement learning with quantum-inspired experience replay," *IEEE Trans. Cybern.*, vol. 52, no. 9, pp. 9326–9338, Sep. 2022.

[10] O. Vinyals et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, Nov. 2019.

[11] D. Silver et al., "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017.

[12] J. Hwangbo et al., "Learning agile and dynamic motor skills for legged robots," *Sci. Robot.*, vol. 4, no. 26, Jan. 2019, Art. no. eaau5872.

[13] Z. Huang, J. Wu, and C. Lv, "Efficient deep reinforcement learning with imitative expert priors for autonomous driving," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jan. 26, 2022, doi: 10.1109/TNNLS.2022.3142822.

[14] J.-A. Li et al., "Quantum reinforcement learning during human decision-making," *Nature Hum. Behav.*, vol. 4, no. 3, pp. 294–307, Jan. 2020.

[15] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, 2018, pp. 3207–3214.

[16] S. Majumdar, S. Khadka, S. Miret, S. Mcaleer, and K. Tumer, "Evolutionary reinforcement learning for sample-efficient multiagent coordination," in *Proc. Int. Conf. Mach. Learn.*, vol. 119, 2020, pp. 6651–6660.

[17] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, and R. Fergus, "Improving sample efficiency in model-free reinforcement learning from images," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, 2021, pp. 10674–10681.

[18] E. Mitchell, R. Rafailov, X. B. Peng, S. Levine, and C. Finn, "Offline meta-reinforcement learning with advantage weighting," in *Proc. Int. Conf. Mach. Learn.*, vol. 139, 2021, pp. 7780–7791.

[19] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *J. Mach. Learn. Res.*, vol. 10, no. 7, pp. 1633–1685, 2009.

[20] Z. Xu, K. Wu, Z. Che, J. Tang, and J. Ye, "Knowledge transfer in multitask deep reinforcement learning for continuous control," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 15146–15155.

[21] M. Abdolshah, H. Le, T. K. George, S. Gupta, S. Rana, and S. Venkatesh, "A new representation of successor features for transfer across dissimilar environments," in *Proc. Int. Conf. Mach. Learn.*, vol. 139, 2021, pp. 1–9.

[22] J. Pan, X. Wang, Y. Cheng, and Q. Yu, "Multisource transfer double DQN based on actor learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2227–2238, Jun. 2018.

[23] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine, "Learning invariant feature spaces to transfer skills with reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, 2017.

[24] E. Parisotto, J. L. Ba, and R. Salakhutdinov, "Actor-mimic: Deep multitask and transfer reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, 2016.

[25] J. Yang, B. Petersen, H. Zha, and D. Faissol, "Single episode policy transfer in reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, 2020.

[26] A. Barreto et al., "Successor features for transfer in reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4058–4068.

[27] T. Brys, A. Harutyunyan, M. E. Taylor, and A. Nowé, "Policy transfer using reward shaping," in *Proc. Int. Conf. Auto. Agents Multiagent Syst.*, 2015, pp. 181–188.

[28] W. Yu, C. K. Liu, and G. Turk, "Protective policy transfer," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 10595–10602.

[29] B. Rosman, M. Hawasly, and S. Ramamoorthy, "Bayesian policy reuse," *Mach. Learn.*, vol. 104, no. 1, pp. 99–127, Jul. 2016.

[30] P. Hernandez-Leal, B. Rosman, M. E. Taylor, L. E. Sucar, and E. M. de Cote, "A Bayesian approach for learning and tracking switching, non-stationary opponents," in *Proc. Int. Conf. Auto. Agents Multiagent Syst.*, 2016, pp. 1315–1316.

[31] T. Yang, J. Hao, Z. Meng, C. Zhang, Y. Zheng, and Z. Zheng, "Towards efficient detection and optimal response against sophisticated opponents," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 623–629.

[32] Y. Zheng, Z. Meng, J. Hao, Z. Zhang, T. Yang, and C. Fan, "A deep Bayesian policy reuse approach against non-stationary agents," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 962–972.

[33] X. Gao, S. Chen, Q. Sun, Y. Zheng, and J. Hao, "A Bayesian policy reuse approach for bilateral negotiation games," in *Proc. AAAI Conf. Artif. Intell., Workshop Reinforcement Learn. Games*, 2022.

[34] Y. Zheng et al., "Efficient policy detecting and reusing for non-stationarity in Markov games," *Auto. Agents Multi-Agent Syst.*, vol. 35, no. 1, pp. 1–29, Apr. 2021.

[35] Y. Tao, S. Genc, J. Chung, T. Sun, and S. Mallya, "Repaint: Knowledge transfer in deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, vol. 139, 2021, pp. 10141–10152.

[36] A. Lazaric, M. Restelli, and A. Bonarini, "Transfer of samples in batch reinforcement learning," in *Proc. 25th Int. Conf. Mach. Learn. (ICML)*, 2008, pp. 544–551.

[37] J. Song, Y. Gao, H. Wang, and B. An, "Measuring the distance between finite Markov decision processes," in *Proc. Int. Conf. Auto. Agents Multiagent Syst.*, 2016, pp. 468–476.

[38] R. Laroche and M. Barlier, "Transfer reinforcement learning with shared dynamics," in *Proc. AAAI Conf. Artif. Intell.*, vol. 31, 2017, pp. 2147–2153.

[39] A. Mustafa, M. Mazouchi, S. Nageshrao, and H. Modares, "Assured learning-enabled autonomy: A metacognitive reinforcement learning framework," *Int. J. Adapt. Control Signal Process.*, vol. 35, no. 12, pp. 2348–2371, Dec. 2021.

[40] F. Fernández, J. García, and M. Veloso, "Probabilistic policy reuse for inter-task transfer learning," *Robot. Auto. Syst.*, vol. 58, no. 7, pp. 866–871, Jul. 2010.

[41] S. Li and C. Zhang, "An optimal online method of selecting source policies for reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, 2018, pp. 3562–3570.

[42] S. Li, F. Gu, G. Zhu, and C. Zhang, "Context-aware policy reuse," in *Proc. Int. Conf. Auto. Agents Multiagent Syst.*, 2019, pp. 989–997.

[43] T. Yang et al., "Efficient deep reinforcement learning via adaptive policy transfer," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 3094–3100.

[44] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[45] M. Seeger, "Gaussian processes for machine learning," *Int. J. Neural Syst.*, vol. 14, no. 2, pp. 69–106, 2004.

[46] E. Akleman, "Deep learning," *Computer*, vol. 53, no. 9, p. 17, Sep. 2020.

[47] C. E. Rasmussen and M. Kuss, "Gaussian processes in reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 4, 2003.

[48] M. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *Proc. Int. Conf. Mach. Learn.*, 2011, pp. 465–472.

[49] F. Doshi-Velez and G. Konidaris, "Hidden parameter Markov decision processes: A semiparametric regression approach for discovering latent task parametrizations," in *Proc. Int. Joint Conf. Artif. Intell.*, 2016, p. 1432.

[50] F. Berkenkamp, M. Turchetta, A. P. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 2, 2018, pp. 909–919.

[51] Z. Wang, H. Li, and C. Chen, "Incremental reinforcement learning in continuous spaces via policy relaxation and importance weighting," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 6, pp. 1870–1883, Jun. 2020.

[52] S. Geva and J. Sitte, "A cartpole experiment benchmark for trainable controllers," *IEEE Control Syst.*, vol. 13, no. 5, pp. 40–51, Oct. 1993.

[53] G. Brockman et al., "OpenAI gym," 2016, *arXiv:1606.01540*.

[54] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 229–256, May 1992.

[55] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1329–1338.

[56] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596.

[57] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 5026–5033.
[58] L. Csató, E. Fokoué, M. Opper, B. Schottky, and O. Winther, "Efficient approaches to Gaussian process classification," *Proc. Adv. Neural Inf. Process. Syst.*, vol. 12, 1999.
[59] L. Csató and M. Opper, "Sparse on-line Gaussian processes," *Neural Comput.*, vol. 14, no. 3, pp. 641–668, Mar. 2002.
[60] N. Lawrence, M. Seeger, and R. Herbrich, "Fast sparse Gaussian process methods: The informative vector machine," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 15, 2002.
[61] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *J. Mach. Learn. Res.*, vol. 1, pp. 211–244, Sep. 2001.

Jinmei Liu (Graduate Student Member, IEEE) received the B.E. degree in automation and the M.E. degree in control science and intelligent engineering from Nanjing University, Nanjing, China, in 2020 and 2023, respectively, where she is currently pursuing the Ph.D. degree with the Department of Control Science and Intelligent Engineering, School of Management and Engineering.

Her research interests include reinforcement learning and machine learning.

Zhi Wang (Member, IEEE) received the B.E. degree in automation from Nanjing University, Nanjing, China, in 2015, and the Ph.D. degree in machine learning from the Department of Systems Engineering and Engineering Management, City University of Hong Kong, Hong Kong, China, in 2019.

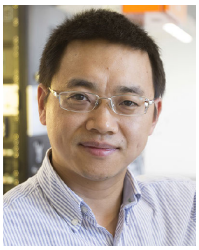
He is currently an Associate Professor with the Department of Control Science and Intelligent Engineering, School of Management and Engineering, Nanjing University. He holds Visiting positions at the University of New South Wales, Canberra, ACT, Australia, and the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China. His current research interests include reinforcement learning, machine learning, and robotics.

Dr. Wang served as the Associate Editor for IEEE International Conference on Systems, Man, and Cybernetics in 2021, 2022, and 2023, and the Associate Editor for IEEE International Conference on Networking, Sensing, and Control in 2020.

Chunlin Chen (Senior Member, IEEE) received the B.E. degree in automatic control and the Ph.D. degree in control science and engineering from the University of Science and Technology of China, Hefei, China, in 2001 and 2006, respectively.

He was a Visiting Scholar at Princeton University, Princeton, NJ, USA, from 2012 to 2013. He had Visiting positions at the University of New South Wales, Canberra, ACT, Australia, and the City University of Hong Kong, Hong Kong, China. He is currently a Full Professor and the Vice Dean of School of Management and Engineering, Nanjing University, Nanjing, China. His recent research interests include reinforcement learning, mobile robotics, and quantum control.

Dr. Chen is the Chair of Technical Committee on Quantum Cybernetics, IEEE Systems, Man and Cybernetics Society.



Daoyi Dong (Fellow, IEEE) received the B.E. degree in automatic control and the Ph.D. degree in engineering from the University of Science and Technology of China, Hefei, China, in 2001 and 2006, respectively.

He was an Alexander von Humboldt Fellow at AKS, University of Duisburg-Essen, Duisburg, Germany. He was with the Institute of Systems Science, Chinese Academy of Sciences, Beijing, China, and with Zhejiang University, Hangzhou, China. He had Visiting positions at Princeton University, NJ, USA; RIKEN, Wako-Shi, Japan; and The University of Hong Kong, Hong Kong. He is currently an Associate Professor at the University of New South Wales, Canberra, ACT, Australia. His research interests include quantum control and machine learning.

Dr. Dong was awarded the ACA Temasek Young Educator Award from the Asian Control Association and was a recipient of Future Fellowship, the International Collaboration Award, and the Australian Post-Doctoral Fellowship from the Australian Research Council, and a Humboldt Research Fellowship from the Alexander von Humboldt Foundation of Germany. He was a Member-at-Large, the Board of Governors, and the Associate Vice President for Conferences and Meetings, IEEE Systems, Man and Cybernetics Society. He served as an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS from 2015 to 2021. He is currently an Associate Editor of the IEEE TRANSACTIONS ON CYBERNETICS and a Technical Editor of the IEEE/ASME TRANSACTIONS ON MECHATRONICS.