TAG: Teacher-Advice Mechanism With Gaussian Process for Reinforcement Learning

Ke Lin, Duantengchuan Li, Yanjie Li, *Member, IEEE*, Shiyu Chen, Qi Liu, Jianqi Gao, Yanrui Jin, and Liang Gong, *Member, IEEE*

***Abstract*— Reinforcement learning (RL) still suffers from the problem of sample inefficiency and struggles with the exploration issue, particularly in situations with long-delayed rewards, sparse rewards, and deep local optimum. Recently, learning from demonstration (LfD) paradigm was proposed to tackle this problem. However, these methods usually require a large number of demonstrations. In this study, we present a sample efficient teacher-advice mechanism with Gaussian process (TAG) by leveraging a few expert demonstrations. In TAG, a teacher model is built to provide both an advice action and its associated confidence value. Then, a guided policy is formulated to guide the agent in the exploration phase via the defined criteria. Through the TAG mechanism, the agent is capable of exploring the environment more intentionally. Moreover, with the confidence value, the guided policy can guide the agent precisely. Also, due to the strong generalization ability of Gaussian process, the teacher model can utilize the demonstrations more effectively. Therefore, substantial improvement in performance and sample efficiency can be attained. Considerable experiments on sparse reward environments demonstrate that the TAG mechanism can help typical RL algorithms achieve significant performance gains. In addition, the TAG mechanism with soft actor-critic algorithm (TAG-SAC) attains the state-of-the-art performance over other LfD counterparts on several delayed reward and complicated continuous control environments.**

***Index Terms*— Gaussian process, reinforcement learning (RL), teacher-advice mechanism.**

NOMENCLATURE

$\mathcal{M}$	Markov decision process (MDP).
$\mathcal{S}, \mathcal{A}$	State space and action space in an MDP.
$\boldsymbol{s}$	State in environments.
$\boldsymbol{a}$	Action.
$P(\cdot \mid \boldsymbol{s}, \boldsymbol{a})$	State transition probability under $\boldsymbol{s}, \boldsymbol{a}$.
r	Reward function.
γ	Discount factor.
π_θ	Policy parameterized with $\boldsymbol{\theta}$.
$\pi_\theta(\boldsymbol{s})$	Action given by π_θ at state $\boldsymbol{s}$.
$\pi_\theta(\cdot \mid \boldsymbol{s})$	Action distribution of π_θ at state $\boldsymbol{s}$.
$\boldsymbol{\theta}$	Parameters of a function.
ρ	Initial state distribution in an MDP.
τ	Trajectory.
$P_{\pi_\theta}(\tau)$	Probability of a trajectory τ under π_θ. Sometimes, $P_{\pi_\theta}(\tau)$ is abbreviated as $\pi_\theta(\tau)$.
$R(\tau)$	Discounted cumulative reward of τ.
$J(\boldsymbol{\theta})$	Expected episode return under $\boldsymbol{\theta}$.
V^π	State value function under policy π.
Q^π	State–action value function under policy π.
Σ	Covariance matrix.
$\mathcal{GP}$	Gaussian process model.
$k(\cdot \mid \cdot)$	Kernel function.
$\mathbf{K}$	Kernel matrix.
$\mathrm{Var}(\boldsymbol{s})$	Variance of $\boldsymbol{s}$.
$\mathrm{Cov}(\boldsymbol{s}, \boldsymbol{s}')$	Covariance between $\boldsymbol{s}$ and $\boldsymbol{s}'$.
$f(\boldsymbol{s})$	Suggested action given by $\mathcal{GP}$ at state $\boldsymbol{s}$.
$\sigma^2(\boldsymbol{s})$	Maximum variance given by $\mathcal{GP}$ at state $\boldsymbol{s}$.
$\mathcal{D}$	Dataset.
σ_n^2	Noise level in $\mathcal{GP}$.
$\mathbf{I}$	Identity matrix.
$U(\cdot \mid \cdot)$	Uniform distribution.
ξ	Random variable from U.
$\mathcal{B}_\delta(\boldsymbol{s})$	Neighborhood of $\boldsymbol{s}$.
π_f	Teacher policy (model).
π_θ^h	Hybrid policy (the guided policy).
T_{gp}	Threshold for the confidence of the suggested action with $\mathcal{GP}$.
T_{eu}	Threshold for the confidence of the suggested action with Euclidean distance.
T_r	Threshold for the average episode return.
$\kappa(\pi_\theta)$	Average episode return of π_θ in a test environment.
β	Threshold for sampling action in π_f.
$\pi_\theta^{h_{\mathrm{eu}}}$	Hybrid policy with Euclidean distance.
$\hat{Q}_\varphi(\boldsymbol{s}, \boldsymbol{a})$	Estimation of the Q parameterized with $\boldsymbol{\varphi}$.
μ_θ	Policy network parameterized with $\boldsymbol{\theta}$. It is the neural network version of π_θ.
$\mathcal{S}_g$	Set of state–action pairs from π_f.
$\mathcal{S}_{\mathrm{ng}}, \mathcal{S}_n$	Sets of normal state–action pairs.
$\boldsymbol{\tau}_g$	Set of trajectories from π_f.

Manuscript received 13 July 2021; revised 14 November 2022; accepted 23 March 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 61977019 and Grant U1813206; and in part by the Shenzhen Fundamental Research Program under Grant JCYJ20180507183837726, Grant JCYJ20220818102415033, and Grant JSGG20201103093802006. *(Corresponding author: Yanjie Li.)*

Ke Lin, Yanjie Li, Shiyu Chen, Qi Liu, and Jianqi Gao are with the Department of Control Science and Engineering, Harbin Institute of Technology (Shenzhen), Shenzhen 518055, China (e-mail: autolyj@hit.edu.cn).

Duantengchuan Li is with the School of Computer Science, Wuhan University, Wuhan 430072, China.

Yanrui Jin and Liang Gong are with the School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2023.3262956.

Digital Object Identifier 10.1109/TNNLS.2023.3262956

I. INTRODUCTION

REINFORCEMENT learning (RL), especially deep RL (DRL), enables intelligent agents to make smart decisions in environments and learn specific skills by interacting with

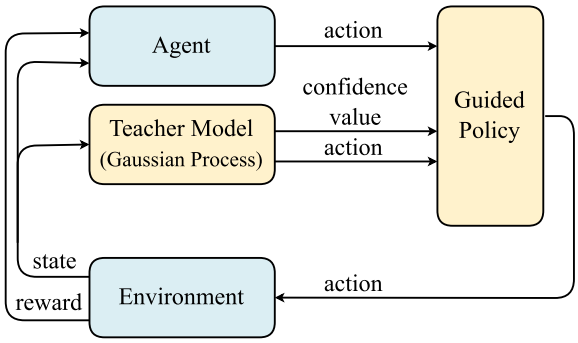

Fig. 1. Brief overview of the TAG for DRL.

environments. Therefore, DRL has been employed in a wide range of scenarios, e.g., games [1], [2], recommender systems [3], [4], [5], robot control [6], [7], autonomous driving [8], and financial trade [9].

Generally, typical RL algorithms can be divided into two categories: policy-based methods (e.g., REINFORCE [10] and policy gradient (PG) [11]) and value-based methods (e.g., Q-learning [12]). For PG, to reduce the estimated variance during the sampling, the baseline technique was introduced into PG, and the vanilla PG (VPG) was proposed [13]. In addition, with the aim of improving the training stability of the policy network, Schulman et al. [14] proposed the trust region policy optimization (TRPO) algorithm, where Kullback–Leibler (KL) divergence was employed to constrain the update of the policy network. After that, an efficient version of TRPO was proposed named proximal policy optimization (PPO) [15], in which the gradient truncation mechanism was applied in the training stage. On the other hand, with the popularity of deep learning, deep neural networks were introduced into Q-learning, and deep Q-network (DQN) was put forward by Mnih et al. [16]. In DQN, human-level control in Atari games was achieved. Then, many improved versions of DQN have been proposed successively, such as double-DQN [17] and duel-DQN [18]. Lillicrap et al. [19] extended DQN to the continuous action space and proposed the deep deterministic PG (DDPG). In addition, Fujimoto et al. [20] put forward an improved version of DDPG, named twin delayed DDPG (TD3), aiming to alleviate the problem of dramatically overestimating the Q values. Moreover, soft actor-critic (SAC) [21] was proposed almost simultaneously with TD3, characterized by maximum entropy regularization.

Despite the significant success, efficient exploration is still a challenging problem in typical DRL, particularly when an environment is characterized by large state spaces, long-delayed rewards, sparse rewards, and deep local optimum. To tackle this problem, in recent years, there has been an increasing interest in learning from demonstration (LfD) methods. For the LfD paradigm, one intuitive idea is to train the policy network with the demonstrations directly [22], [23], [24]. However, limited by supervised learning manner, these methods usually cannot make full use of the demonstrations and may not work well in complex sequential decision tasks. To address this deficiency, some researchers utilized the idea from generative adversarial networks (GANs) [25] and proposed a variety of generative adversarial imitation learning (GAIL) methods [26], [27], [28], [29]. This type of method aims at minimizing the distance of the distributions between expert demonstrations and the policy, and it achieves good results in many control tasks. In addition, there are also methods that combine RL with demonstrations (RLfD) [30], [31], [32], [33], [34]. Different from the GAIL-style methods, this kind of method can not only learn from the environment with reward signals but also learn from the demonstration data. This combination manner can be split in three major ways: 1) put the demonstrations into the replay buffer; 2) combine the loss function of RL and supervised learning; and 3) mimic the expert behaviors by reward shaping. However, inappropriate combinations may have the opposite effect on the training of RL algorithms. In addition, methods that just put expert demonstrations into the replay buffer do not make full use of demonstrations. Hence, they will also encounter the problem of low sample efficiency.

In this article, we proposed a sample efficient teacher-advice mechanism with Gaussian process (TAG), which can be used in typical DRL algorithms. The framework is shown in Fig. 1. With demonstrations, a teacher model is built using a Gaussian process model. The role of the teacher model is to give a suggested action. Meanwhile, the corresponding confidence value will be output. Then, the guided policy selects an action between the advice action and the action given by the policy, based on a set of criteria. Hence, the proposed model is essentially a teacher-advice framework [35]. With the help of the powerful data generalization ability of the Gaussian process model, the TAG mechanism can make full use of demonstrations, and the sample efficiency is improved. In addition, equipped with the characteristics of being able to give the confidence value, the TAG mechanism can guide the agent accurately in the exploration phase and make the agent possess a greater preference to explore high-reward states. Therefore, better exploration can be achieved by direct and accurate guidance, which makes the DRL algorithm be trained faster in complex environments. The main contribution of this work is summarized as follows.

1) We formalize the TAG mechanism, which can directly intervene in the exploration process of an agent and enable the agent to have a better exploration for the high-reward states, i.e., with a higher probability. Thus, better performance is achieved through better exploration.
2) To fully use and generalize the demonstration data, a Gaussian process is utilized to model the data, which can be used as a teacher model to guide the agent's exploration. The teacher model can not only give the advice action but also give the corresponding confidence so that it can guide the agent more accurately.
3) We theoretically prove that compared with the existing ordinary policies, the proposed guided policy in the TAG mechanism will collect high-reward samples with greater probability. Thus, the guided policy can enable RL algorithms to converge faster.
4) We evaluate the TAG mechanism on empirical experiments, which demonstrates that our mechanism can significantly improve the training efficiency for many DRL algorithms, such as VPG, PPO, DDPG, and SAC. In addition, compared with the existing LfD methods, e.g., soft actor-critic from demonstration (SACfD), GAIL-SAC, and VAIL-SAC, the TAG mechanism with soft actor-critic algorithm (TAG-SAC) can achieve the

best performance and the fastest convergence rate in several complex control environments.

The rest of this article is structured as follows. In Section II, we describe the related work. In Section III, preliminaries about RL, Gaussian policy, and Gaussian process are introduced. Section IV gives the detail of the proposed TAG mechanism and the corresponding TAG with DRL algorithms. In Section V, experiments are carried out in several different environments. Finally, the conclusion and discussion are summarized in Section VI.

II. Related Work

A. Learning From Demonstration

LfD also known as imitation learning is a learning paradigm designed to make agents learn from expert demonstrations [36]. Behavioral cloning (BC) [22] is a common method in imitation learning, in which demonstrations are directly used to train the policy in a supervised manner. However, for sequential decision problems, due to the cumulative error caused by distribution shift [37], BC only tends to perform well when enormous demonstrations are available. To solve this problem, many interactive imitation learning methods were proposed, such as DAgger [23] and Deeply AggreVaTeD [24], where the distribution shift problem is solved by expanding the dataset interactively and labeling novel states.

On the other hand, based on GAN [25], GAIL was put forward by Ho and Ermon [26]. In GAIL, two networks, a generator network and a discriminator network, are employed to reduce the distribution difference between the data obtained by the agent and the data in expert demonstrations. The goal of the discriminator network is to distinguish whether the data come from the demonstrations or the agent, while the generator network aims to confuse the discriminator network. However, GAIL is sensitive to environmental noise, which makes GAIL's training unstable. Therefore, variational adversarial imitation learning (VAIL) [27] was proposed, where the information bottleneck (IB) was introduced to avoid the vanishing gradient in the generator network. Subsequently, Zhang et al. [28] introduced the f-divergence into GAIL and proposed the f-GAIL. Afterward, Ghasemipour et al. [29] unified this kind of adversarial imitation learning algorithm into a new framework named f-MAX.

B. RL With Demonstrations

Another type of LfD method is to combine RL with expert demonstrations. Hester et al. [30] proposed deep Q-learning from demonstration (DQNfD) method, in which the demonstration data were put into the experience replay buffer. Moreover, to make better use of the demonstrations, mean squared error (mse) and $L2$ regularization loss were added to the loss function of the Q-network. With a similar scheme, Nair et al. [31] proposed OERL methods, achieving good performance in several robot manipulation tasks. Different from the DQNfD, OERL adopted the DDPG as the backbone DRL model, and a reset mechanism was used to make sure that the expert demonstrations could be utilized in long-horizon tasks. In addition, Kang et al. [32] proposed policy optimization with demonstrations (POfD) method, where they adopted the reward shaping technique by inserting a divergence item to the original reward, enforcing the occupancy measure to match between the policy and expert demonstrations. Furthermore, other methods that incorporate the supervised learning paradigm into RL include Soft-RLfD [33] and DAC [38].

C. Teacher-Advice Framework

The teacher-advice framework is usually used in safe RL [35], in which a teacher model is built to guarantee that no harmful action is performed by the agent in risk-sensitive tasks. Alshiekh et al. [39] put forward the shielded RL, in which manually defined rules were modeled as a finite-state machine, and the finite-state machine was considered as the teacher model. Cheng et al. [40] proposed an end-to-end safe RL algorithm based on the teacher-advice framework, where a control barrier function (CBF) was used to build the teacher model. For CBF, the specific function needs to be formulated before training, e.g., linear functions or quadratic functions. In addition, Neider et al. [41] also constructed a teacher model with a finite-state machine and proposed the AdvisoRL algorithm.

In addition, Radac and Precup [42] proposed a virtual reference feedback tuning method that is used to obtain a stabilizing nonlinear state-feedback controller as an initial controller for DRL, in which a teacher model is built by initial random exploration samples to guide the subsequent exploration of the agent in a more efficient style.

In this article, we propose the teacher-advice mechanism based on Gaussian process to overcome the problem of sample inefficiency and accelerate the training process in DRL algorithms.

III. Preliminaries

A. Reinforcement Learning

In RL, problems are usually modeled as a Markov decision process (MDP). An MDP can be represented by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state space and action space, respectively. The state transition probability $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ describes the probability of transition from state $\boldsymbol{s}_t$ to next state $\boldsymbol{s}_{t+1}$ under action $\boldsymbol{a}_t$. $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function. $\gamma \in [0, 1]$ is the discount factor.

In an MDP, a policy is defined as a function $\pi_\theta : \mathcal{S} \times \mathcal{A} \to [0, 1]$. Thus, the policy function parameterized by $\boldsymbol{\theta}$, denoted by $\pi_\theta(\boldsymbol{a}|\boldsymbol{s})$, describes the probability of performing action $\boldsymbol{a}$ at state $\boldsymbol{s}$. $\rho : \mathcal{S} \to [0, 1]$ gives the occurrence probability of the initial state $\boldsymbol{s}$. In addition, an episode corresponds to a trajectory $\tau = \{\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1, \boldsymbol{a}_1, \ldots\}$. Then, in an MDP, the probability of a trajectory can be defined as

$$P_{\pi_\theta}(\tau) = \rho(\boldsymbol{s}_0) \prod_{t=0}^{T} \pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t) P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t). \tag{1}$$

The goal of an MDP is to find a policy π_θ that can maximize the expected episode return

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\tau \sim P_{\pi_\theta}}[R(\tau)] \tag{2}$$

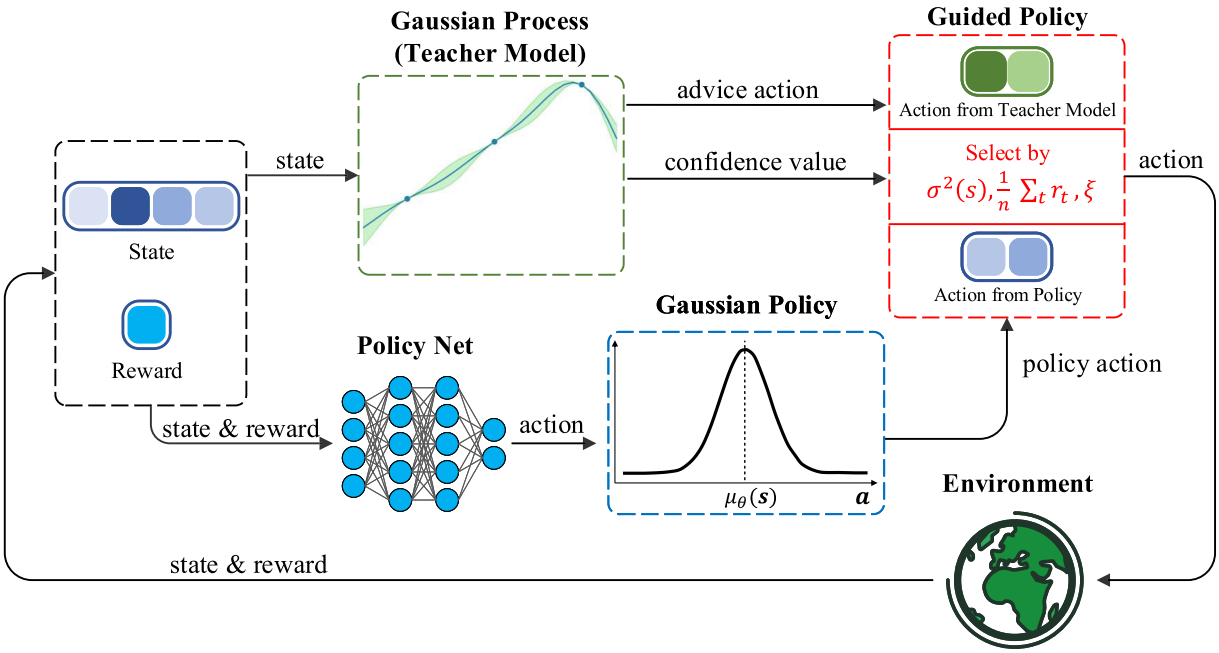

Fig. 2. Illustration of the proposed TAG mechanism for DRL algorithms.

where $R(\tau) = \sum_{t=0}^{T} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t)$. Therefore, the gradient of $J(\boldsymbol{\theta})$ can be written as

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\tau \sim P_{\pi_\theta}}[R(\tau)] \\ &= \mathbb{E}_{\tau \sim P_{\pi_\theta}}[\nabla_{\boldsymbol{\theta}} \log P_{\pi_\theta}(\tau) R(\tau)] \\ &\approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log P_{\pi_\theta}(\tau_i) R(\tau_i) \end{aligned} \tag{3}$$

where the subscript represents the ith trajectory. The objective function $J(\boldsymbol{\theta})$ is usually optimized by gradient ascent repetitively, i.e., $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$, where α denotes the step size. This iterative process will eventually make $\boldsymbol{\theta}$ converge [10].

In addition, the state value function $V^\pi(\boldsymbol{s})$ and the state–action value function $Q^\pi(\boldsymbol{s}, \boldsymbol{a})$ are introduced in RL to evaluate the performance of a policy π. They are defined as

$$V^\pi(\boldsymbol{s}) = \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t) \middle| \boldsymbol{s}_0 = \boldsymbol{s} \right] \tag{4}$$

$$Q^\pi(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t) \middle| \begin{matrix} \boldsymbol{s}_0 = \boldsymbol{s} \\ \boldsymbol{a}_0 = \boldsymbol{a} \end{matrix} \right]. \tag{5}$$

B. Gaussian Policy

The Gaussian policy is usually employed in environments with continuous state space and action space. It endows the agent with random exploration ability. A Gaussian policy can be written as

$$\pi_\theta(\boldsymbol{a}|\boldsymbol{s}) = \frac{\exp\left\{-\frac{1}{2}(\boldsymbol{a} - \mu_\theta(\boldsymbol{s}))^\top \Sigma^{-1} (\boldsymbol{a} - \mu_\theta(\boldsymbol{s}))\right\}}{(2\pi)^{m/2} |\Sigma|^{1/2}} \tag{6}$$

where μ_θ may be a neural network parameterized by $\boldsymbol{\theta}$, $\boldsymbol{\Sigma}$ is the covariance matrix, and m is the dimension of the action space.

C. Gaussian Process

A Gaussian process model can be specified by a mean function $m(\boldsymbol{s})$ and a covariance function $k(\boldsymbol{s}, \boldsymbol{s}')$ [43]. Then, a Gaussian process model can be written as $\mathcal{GP}\big(m(\boldsymbol{s}), k(\boldsymbol{s}, \boldsymbol{s}')\big)$. Normally, we set $m(\boldsymbol{s}) = \boldsymbol{0}$. In addition, $k(\boldsymbol{s}, \boldsymbol{s}')$ is a kernel function. In this article, we adopt the squared exponential function

$$k(\boldsymbol{s}, \boldsymbol{s}') = \exp\left(-\frac{d^2(\boldsymbol{s}, \boldsymbol{s}')}{2l^2}\right) \tag{7}$$

where $d(\boldsymbol{s}, \boldsymbol{s}')$ is the Euclidean distance between $\boldsymbol{s}$ and $\boldsymbol{s}'$ and l is a length-scale parameter, which can be seen as a hyperparameter. Matern kernel [43] is also used in this article, which is defined as

$$k(\boldsymbol{s}, \boldsymbol{s}') = \frac{\left(\frac{\sqrt{2\nu}}{l} d(\boldsymbol{s}, \boldsymbol{s}')\right)^\nu}{\Gamma(\nu) 2^{\nu-1}} \mathbf{K}_\nu \left(\frac{\sqrt{2\nu}}{l} d(\boldsymbol{s}, \boldsymbol{s}')\right) \tag{8}$$

where $\Gamma(\cdot)$ is the gamma function, $\mathbf{K}_\nu$ is a modified Bessel function, and ν is a hyperparameter specifying the smoothness of the function.

Given a dataset $\mathcal{D} = \{\boldsymbol{s}_i, a_i\}_{i=1,2,\ldots,n}$ and a new $\boldsymbol{s}_*$ that does not appear in the dataset, the predictive value and the corresponding variance and covariance of $\boldsymbol{s}_*$ are

$$f(\boldsymbol{s}_*) = \boldsymbol{k}(\boldsymbol{s}_*)^\top \left(\mathbf{K} + \sigma_n^2 \mathbf{I}\right)^{-1} \boldsymbol{y} \tag{9}$$

$$\mathrm{Var}(\boldsymbol{s}_*) = \mathrm{Cov}(\boldsymbol{s}_*, \boldsymbol{s}_*) \tag{10}$$

$$\mathrm{Cov}(\boldsymbol{s}_*, \boldsymbol{s}) = k(\boldsymbol{s}_*, \boldsymbol{s}) - \boldsymbol{k}(\boldsymbol{s}_*)^\top \left(\mathbf{K} + \sigma_n^2 \mathbf{I}\right)^{-1} \boldsymbol{k}(\boldsymbol{s}) \tag{11}$$

where $\boldsymbol{k}(\boldsymbol{s}_*) = [k(\boldsymbol{s}_1, \boldsymbol{s}_*), k(\boldsymbol{s}_2, \boldsymbol{s}_*), \ldots, k(\boldsymbol{s}_n, \boldsymbol{s}_*)]^\top$. $\mathbf{K}$ is the kernel matrix generated by $k(\boldsymbol{s}_i, \boldsymbol{s}_j)$. σ_n^2 refers to the noise level in the dataset, and $\boldsymbol{y} = [a_1, a_2, \ldots, a_n]^\top$.

IV. Methodology

In this section, we first describe the overview of the proposed method. Then, the teacher model built with demonstrations using the Gaussian process is presented. Third,

we introduce the details of the guided policy. Then, we analyze the advancement of the guided policy for typical DRL methods.

A. Overview of TAG

In this article, we aim to use demonstration data to accelerate the training process of normal DRL algorithms in complicated environments.

The overview of the proposed TAG mechanism is shown in Fig. 2, which mainly includes a teacher model and a guided policy. The two of them form the teacher-advice mechanism. The teacher model is constructed by a Gaussian process to output two critical elements, i.e., an advice action and the corresponding confidence value. As an action selector, the guided policy is used to choose an action from the teacher model and the policy network through the defined criteria. Then, the selected action is executed by the agent to interact with the environment.

In our framework, a neural network is modeled as the policy, named policy network, whose input is the state, and the output is the action. The optimization goal of the policy network is to maximize the agent's cumulative reward. In addition, to assist in the optimization of the policy network, a critic network is constructed, which is used to evaluate the performance of the policy. For the critic network, the input is also the state, and the output is a real number that represents the cumulative reward of the policy at a state. The optimization goal of the critic network is to minimize the mse between the predicted cumulative reward and the real cumulative reward. Thus, the data collected from interacting with the environment will be used to train the policy network and the critic network. The details of the teacher model, the guided policy, and the teacher-advice mechanism for DRL algorithms will be described in Sections IV-B–IV-E.

B. Teacher Model With Gaussian Process

The teacher model is used to give an advice action to the agent, and it is constructed on demonstration data with Gaussian process. By utilizing the Gaussian process, states and actions that do not appear in the dataset are predicted according to the existing demonstration data. Meanwhile, the variance of the prediction will be given, which is used to illustrate the confidence of the prediction.

The superiority of leveraging Gaussian process is that it can provide the teacher model with considerable generalization ability. In addition, the variance output by the Gaussian process can help to guide the agent more accurately.

The Gaussian process is a nonparametric model, but the parameters of its kernel function can still be learned by the training data $\mathcal{D} = \{\boldsymbol{s}_i, a_i\}_{i=1,2,\ldots,n}$. To optimize the parameters of the kernel function such that the probability of occurrence of all a_i is maximized given all $\boldsymbol{s}_i$, the objective can be designed to maximize the likelihood function

$$\log p(\boldsymbol{y}|\mathbf{S}, \boldsymbol{\theta}) = -\frac{1}{2}\boldsymbol{y}^\top \mathbf{K}_y^{-1}\boldsymbol{y} - \frac{1}{2}\log|\mathbf{K}_y| - \frac{n}{2}\log 2\pi \quad (12)$$

where $\mathbf{K}_y = \mathbf{K} + \sigma_n^2\mathbf{I}$ and $\boldsymbol{y} = [a_1, a_2, \ldots, a_n]^\top$. $\mathbf{S}$ is a matrix composed of inputs of the dataset, i.e., $\mathbf{S} = [\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n]^\top$. $\boldsymbol{\theta}$ is the hyperparameter in the kernel function. The derivation of (12) can refer to in [43, Eq. (2.30)]. The objective function can be optimized using gradient-based methods. Then, we can predict the action at state $\boldsymbol{s}$ and its corresponding variance by (9) and (10).

Demonstrations contain a series of state–action pairs, denoted by $\mathcal{D}_d = \{\boldsymbol{s}_i, \boldsymbol{a}_i\}_{i=1,\ldots,n}$, where the subscript i represents the ith sample. Therefore, we can use the demonstration data to build the teacher model. However, because the advice action output by the teacher model is multidimensional, we model each component of action $\boldsymbol{a} = [a^{(1)}, a^{(2)}, \ldots, a^{(m)}]^\top$ with a Gaussian process model. As a consequence, the teacher model is framed as

$$\begin{cases} a^{(1)} = f_1(\boldsymbol{s}) \\ a^{(2)} = f_2(\boldsymbol{s}) \\ \vdots \\ a^{(m)} = f_m(\boldsymbol{s}) \end{cases} \quad \begin{cases} a_{\sigma^2}^{(1)} = \mathrm{Var}_1(\boldsymbol{s}) \\ a_{\sigma^2}^{(2)} = \mathrm{Var}_2(\boldsymbol{s}) \\ \vdots \\ a_{\sigma^2}^{(m)} = \mathrm{Var}_m(\boldsymbol{s}) \end{cases} \quad (13)$$

where $a_{\sigma^2}^{(i)}$ represents the prediction variance for $a^{(i)}$. Then, we denote the action and the corresponding maximum variance output by the teacher model as

$$\begin{cases} f(\boldsymbol{s}) = [f_1(\boldsymbol{s}), f_2(\boldsymbol{s}), \ldots, f_m(\boldsymbol{s})]^\top \\ \sigma^2(\boldsymbol{s}) = \max\{\mathrm{Var}_1(\boldsymbol{s}), \mathrm{Var}_2(\boldsymbol{s}), \ldots, \mathrm{Var}_m(\boldsymbol{s})\}. \end{cases} \quad (14)$$

Then, we can define that the advice action is sampled from a uniform distribution parameterized by $f(\boldsymbol{s})$ and a vector $\boldsymbol{\delta}$, where $\boldsymbol{\delta}_i \in (0, 1)$ is a small real number

$$\boldsymbol{a} \sim U(f(\boldsymbol{s}) - \boldsymbol{\delta}, f(\boldsymbol{s}) + \boldsymbol{\delta}) \quad (15)$$

which means that the advice action belongs to a neighborhood of $f(\boldsymbol{s})$. The neighborhood is denoted as $\mathcal{B}_\delta(\boldsymbol{s})$, and it can be defined as $\mathcal{B}_\delta(\boldsymbol{s}) = \{\boldsymbol{a} \in \mathcal{A} \mid |\boldsymbol{a}^i - f_i(\boldsymbol{s})| < \delta, i = 1, 2, \ldots, m\}$.

C. Guided Policy

The guided policy π_θ^h is a hybrid policy, which is composed of a teacher model $\pi_f(\boldsymbol{s}) \sim U(f(\boldsymbol{s}) - \boldsymbol{\delta}, f(\boldsymbol{s}) + \boldsymbol{\delta})$ and a Gaussian policy $\pi_\theta(\boldsymbol{s})$, as shown in the following equation:

$$\pi_\theta^h(\boldsymbol{s}) = \begin{cases} \pi_f(\boldsymbol{s}), & \sigma^2(\boldsymbol{s}) < T_{\mathrm{gp}}, \xi < \beta, \kappa(\pi_\theta) < T_r \\ \pi_\theta(\boldsymbol{s}), & \text{else} \end{cases} \quad (16)$$

where $\sigma^2(\boldsymbol{s})$ is the maximum variance of state $\boldsymbol{s}$ predicted by the teacher model, which is used to indicate the confidence of the advice action. T_{gp} is a threshold for this confidence.

Remark 1: From (16), we can know that when the variance of the action given by the teacher model is large, i.e., $\sigma^2(\boldsymbol{s}) \geq T_{\mathrm{gp}}$, the agent will not adopt the suggestion of the teacher model but choose its own action. This means that when the teacher model does not have enough confidence in the advice action, and the agent will give up the suggestion.

In addition, ξ is a sample from a uniform distribution $U(0, 1)$, and $\beta \in (0, 1)$ is also a threshold for ξ to make sure that the teacher model guides the agent with a certain probability. This setting aims to enhance the exploration of the agent. $\kappa(\pi_\theta) = 1/M \sum_{i=1}^{M} R(\tau_i)$ is defined as the average episode return of π_θ in the test environment. T_r is a threshold to guarantee that when the performance of the agent is good enough, the teacher model does not give advice to the policy π_θ. This setting guarantees that the agent is not restricted by the teacher model during the optimization process.

Therefore, we can obtain the probability of performing action $\boldsymbol{a}$ at state $\boldsymbol{s}$ under policy π_θ^h. When $\kappa(\pi_\theta) < T_r$, we have

the following probability density function:

$$\pi_\theta^h(\boldsymbol{a}|\boldsymbol{s}) = \begin{cases} \pi_\theta(\boldsymbol{a}|\boldsymbol{s}), & \sigma^2(\boldsymbol{s}) \geq T_{\text{gp}} \\ (1-\beta)\pi_\theta(\boldsymbol{a}|\boldsymbol{s}) + \dfrac{\beta}{|\mathcal{B}_\delta(\boldsymbol{s})|}, & \sigma^2(\boldsymbol{s}) < T_{\text{gp}}, \boldsymbol{a} \in \mathcal{B}_\delta(\boldsymbol{s}) \\ (1-\beta)\pi_\theta(\boldsymbol{a}|\boldsymbol{s}), & \sigma^2(\boldsymbol{s}) < T_{\text{gp}}, \boldsymbol{a} \notin \mathcal{B}_\delta(\boldsymbol{s}) \end{cases} \tag{17}$$

where $|\mathcal{B}_\delta(\boldsymbol{s})| = (2\delta)^m$ represents the Lebesgue measure of $\mathcal{B}_\delta(\boldsymbol{s})$ and m represents the dimension of $\boldsymbol{a}$. When $\sigma^2(\boldsymbol{s}) \geq T_{\text{gp}}$, the suggested action given by the teacher model has low confidence. Thus, the agent executes the original policy. The action distribution in this case is $\pi_\theta(\boldsymbol{a}|\boldsymbol{s})$. When $\sigma^2(\boldsymbol{s}) < T_{\text{gp}}$, the confidence of the action given by the teacher model is high. However, the agent will still not take the suggested action with probability $1-\beta$. Thus, when $\boldsymbol{a} \notin \mathcal{B}_\delta(\boldsymbol{s})$, which means that the action is not the suggested action, the probability density of the action $\boldsymbol{a}$ should be $(1-\beta)\pi_\theta(\boldsymbol{a}|\boldsymbol{s})$. When $\boldsymbol{a} \in \mathcal{B}_\delta(\boldsymbol{s})$, there are two possibilities: 1) the action is directly sampled in $\mathcal{B}_\delta(\boldsymbol{s})$, whose probability density is $(\beta/\mathcal{B}_\delta(\boldsymbol{s}))$, and 2) the action is selected from the original policy, but it happens to belong to $\mathcal{B}_\delta(\boldsymbol{s})$, whose probability density is also $(1-\beta)\pi_\theta(\boldsymbol{a}|\boldsymbol{s})$. Thus, the final probability density is $(1-\beta)\pi_\theta(\boldsymbol{a}|\boldsymbol{s}) + (\beta/|\mathcal{B}_\delta(\boldsymbol{s})|)$. In addition, when $\kappa(\pi_\theta) \geq T_r$, we have

$$\pi_\theta^h(\boldsymbol{a}|\boldsymbol{s}) = \pi_\theta(\boldsymbol{a}|\boldsymbol{s}). \tag{18}$$

According to (17) and (18), when $\kappa(\pi_\theta) < T_r$ and $\sigma^2(\boldsymbol{s}) < T_{\text{gp}}$, we have

$$\begin{aligned} \int_{\mathcal{A}} \pi_\theta^h(\boldsymbol{a}|\boldsymbol{s})d\boldsymbol{a} &= \int_{\mathcal{B}_\delta(\boldsymbol{s})} \left[(1-\beta)\pi_\theta(\boldsymbol{a}|\boldsymbol{s}) + \frac{\beta}{|\mathcal{B}_\delta(\boldsymbol{s})|}\right] d\boldsymbol{a} \\ &\quad + \int_{\mathcal{A}\setminus\mathcal{B}_\delta(\boldsymbol{s})} (1-\beta)\pi_\theta(\boldsymbol{a}|\boldsymbol{s})d\boldsymbol{a} \\ &= \int_{\mathcal{A}} (1-\beta)\pi_\theta(\boldsymbol{a}|\boldsymbol{s})d\boldsymbol{a} + \int_{\mathcal{B}_\delta(\boldsymbol{s})} \frac{\beta}{|\mathcal{B}_\delta(\boldsymbol{s})|} d\boldsymbol{a} \\ &= 1-\beta+\beta = 1 \end{aligned} \tag{19}$$

and when $\kappa(\pi_\theta) \geq T_r$ or $\sigma^2(\boldsymbol{s}) \geq T_{\text{gp}}$, we have

$$\int_{\mathcal{A}} \pi_\theta^h(\boldsymbol{a}|\boldsymbol{s})d\boldsymbol{a} = \int_{\mathcal{A}} \pi_\theta(\boldsymbol{a}|\boldsymbol{s})d\boldsymbol{a} = 1. \tag{20}$$

Therefore, we get that $\int_{\mathcal{A}} \pi_\theta^h(\boldsymbol{a}|\boldsymbol{s})d\boldsymbol{a} = 1$.

D. π_θ^h for Policy-Based DRL

Consider the case where $\kappa(\pi_\theta) < T_r$. It can be seen from (17) that the state–action pairs $(\boldsymbol{s}, \boldsymbol{a})$ under the guided policy π_θ^h can be divided into three categories.

1) $\mathcal{S}_g = \{(\boldsymbol{s}, \boldsymbol{a}) | \sigma^2(\boldsymbol{s}) < T_{\text{gp}}, \boldsymbol{a} \in \mathcal{B}_\delta(\boldsymbol{s})\}$.
2) $\mathcal{S}_{\text{ng}} = \{(\boldsymbol{s}, \boldsymbol{a}) | \sigma^2(\boldsymbol{s}) < T_{\text{gp}}, \boldsymbol{a} \notin \mathcal{B}_\delta(\boldsymbol{s})\}$.
3) $\mathcal{S}_n = \{(\boldsymbol{s}, \boldsymbol{a}) | \sigma^2(\boldsymbol{s}) \geq T_{\text{gp}}, \boldsymbol{a} = \pi_\theta(\boldsymbol{a}|\boldsymbol{s})\}$.

Lemma 1: For a Gaussian policy $\pi_\theta(\boldsymbol{a}|\boldsymbol{s})$, if $\delta \leq (1/2)\sqrt{2\pi}|\Sigma|^{(1/2m)}$, then $|\mathcal{B}_\delta(\boldsymbol{s})|\pi_\theta(\boldsymbol{a}|\boldsymbol{s}) \leq 1, \forall \boldsymbol{s} \in \mathcal{S}, \forall \boldsymbol{a} \in \mathcal{A}$.

Proof: For a Gaussian policy, we have

$$\pi_\theta(\boldsymbol{a}|\boldsymbol{s}) \leq \frac{1}{(2\pi)^{m/2}|\Sigma|^{1/2}}.$$

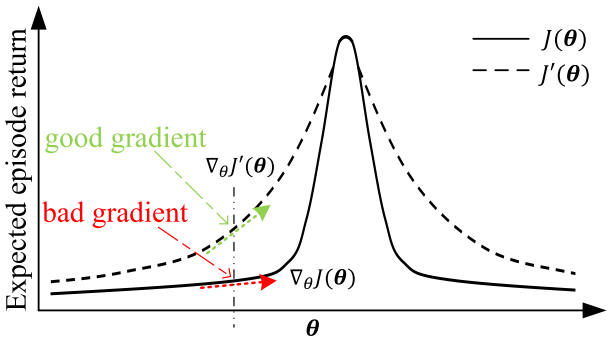

Fig. 3. General view of how (22) works. In sparse reward environments, it is difficult for the agent to collect high-reward trajectories in the early stage of training, resulting in an unfavorable gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$. The guided policy π_θ^h can increase the probability of the high-reward trajectories, so the obtained gradient is more helpful to train $\boldsymbol{\theta}$.

Hence, we can get

$$\begin{aligned} |\mathcal{B}_\delta(\boldsymbol{s})|\pi_\theta(\boldsymbol{a}|\boldsymbol{s}) &\leq |\mathcal{B}_\delta(\boldsymbol{s})| \frac{1}{(2\pi)^{m/2}|\Sigma|^{1/2}} \\ &= (2\delta)^m \frac{1}{(2\pi)^{m/2}|\Sigma|^{1/2}} \\ &\leq (2 \times \frac{1}{2}\sqrt{2\pi}|\Sigma|^{\frac{1}{2m}})^m \frac{1}{(2\pi)^{m/2}|\Sigma|^{1/2}} \\ &= 1. \end{aligned}$$

□

Lemma 2: When $\kappa(\pi_\theta) < T_r$, if $(\boldsymbol{s}, \boldsymbol{a}) \in \mathcal{S}_g$ and $\delta \leq (1/2)\sqrt{2\pi}|\Sigma|^{(1/2m)}$, then $\pi_\theta^h(\boldsymbol{a}|\boldsymbol{s}) \geq \pi_\theta(\boldsymbol{a}|\boldsymbol{s})$ holds.

Proof: By (17), for $(\boldsymbol{s}, \boldsymbol{a}) \in \mathcal{S}_g$, we have

$$\begin{aligned} \pi_\theta^h(\boldsymbol{a}|\boldsymbol{s}) &= (1-\beta)\pi_\theta(\boldsymbol{a}|\boldsymbol{s}) + \frac{\beta}{|\mathcal{B}_\delta(\boldsymbol{s})|} \\ &= \pi_\theta(\boldsymbol{a}|\boldsymbol{s}) + \left(\frac{1}{|\mathcal{B}_\delta(\boldsymbol{s})|} - \pi_\theta(\boldsymbol{a}|\boldsymbol{s})\right)\beta \\ &\geq \pi_\theta(\boldsymbol{a}|\boldsymbol{s}) \quad \text{(by Lemma 1)}. \end{aligned}$$

□

It is worth noting that samples in $\mathcal{S}_g$ can be seen as favorable pairs that are beneficial to gradient updates. This is because actions in $\mathcal{S}_g$ are from the teacher model. Moreover, actions given by the teacher model can be seen as potentially high-reward pairs. Thus, state–action pairs in $\mathcal{S}_g$ have a beneficial effect on optimizing the parameters of the policy. From Lemma 2, we can know that favorable actions will appear in π_θ^h with a greater probability than π_θ.

Consider a trajectory $\tau = \{\boldsymbol{s}_t, \boldsymbol{a}_t\}_{t=0,1,\ldots}$ that satisfies the following properties: every $(\boldsymbol{s}, \boldsymbol{a})$ in τ belongs to $\mathcal{S}_g$, and we define the set of this kind of trajectory as $\boldsymbol{\tau}_g$, which can be written as

$$\boldsymbol{\tau}_g = \{\tau \mid (\boldsymbol{s}_t, \boldsymbol{a}_t) \in \mathcal{S}_g\}. \tag{21}$$

As before, an action $\boldsymbol{a}$ in $\boldsymbol{\tau}_g$ is from the teacher model, so trajectories in $\boldsymbol{\tau}_g$ will be helpful to optimize the parameters of the policy. To simplify the notations, we denote $P_{\pi_\theta}(\tau)$ as $\pi_\theta(\tau)$. Then, we have the following proposition.

Proposition 1: $\forall \tau \in \boldsymbol{\tau}_g$, $\pi_\theta^h(\tau) \geq \pi_\theta(\tau)$ holds.

Proof: By (1), we have

$$\begin{aligned}\pi_\theta^h(\tau) &= \rho(\boldsymbol{s}_0)\prod_{t=0}^{T}\pi_\theta^h(\boldsymbol{a}_t|\boldsymbol{s}_t)P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t,\boldsymbol{a}_t)\\ &\geq \rho(\boldsymbol{s}_0)\prod_{t=0}^{T}\pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t)P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t,\boldsymbol{a}_t) \quad \text{(by Lemma 2)}\\ &= \pi_\theta(\tau).\end{aligned}$$

□

Remark 2: From Proposition 1, we can know that compared with the ordinary policy π_θ, the hybrid guided policy π_θ^h will collect high-reward samples with a greater probability.

In complicated environments, such as sparse reward and delayed reward environments, the useful reward signal given in the environment is very sparse. In this case, when the agent interacts with the environment, the low-reward and even zero-reward trajectories will account for the vast majority, while favorable trajectories, i.e., trajectories with larger positive rewards, will be in the minority. Therefore, the real useful gradient direction does not dominate in $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$. As a result, the policy network converges slowly. To alleviate this drawback, our intuition is to increase the probability of high-return trajectories. As a result, we consider replacing $\pi_\theta(\tau)$ with $\pi_\theta^h(\tau)$ in (3), so we can get

$$\nabla_{\boldsymbol{\theta}} J'(\boldsymbol{\theta}) = \int_\tau \pi_\theta^h(\tau)\nabla_{\boldsymbol{\theta}}\log\pi_\theta(\tau)R(\tau)d\tau. \tag{22}$$

From Proposition 1, it can be concluded that (22) can increase the probability of occurrence of $\tau \in \boldsymbol{\tau}_g$. As a consequence, using (22) to update the parameters $\boldsymbol{\theta}$ of the policy can accelerate the training process to a certain extent. For policy-based RL algorithms, we adopt (22) to update the parameters of the policy network using gradient ascent. Fig. 3 shows a general view of the advantages of using $\nabla_{\boldsymbol{\theta}} J'(\boldsymbol{\theta})$ for gradient ascent.

It is worth mentioning that without importance sampling, (22) is biased. However, importance sampling usually brings huge variance. Therefore, there is a tradeoff between unbiasedness and convergence rate. When $\beta = 0$, the teacher model does not guide the agent, so the gradient is unbiased, but the convergence is slow. When β increases, the gradient direction becomes biased, but the more favorable trajectories are collected, achieving faster convergence speed.

Proposition 2: If we use $\nabla_{\boldsymbol{\theta}} J'(\boldsymbol{\theta})$ to update the parameters $\boldsymbol{\theta}$ with $\boldsymbol{\theta}_{i+1} \leftarrow \boldsymbol{\theta}_i + \alpha_i \nabla_{\boldsymbol{\theta}} J'(\boldsymbol{\theta}_i)$, where $\sum_i^\infty \alpha_i = \infty$ and $\sum_i^\infty \alpha_i^2 \leq \infty$. During the iteration, once $\boldsymbol{\theta}$ satisfies the condition $T_r \leq \kappa(\pi_\theta)$, then $\boldsymbol{\theta}$ will eventually converge to a local optimum $\boldsymbol{\theta}_*$.

Proof: When $\boldsymbol{\theta}$ satisfies the condition $T_r \leq \kappa(\pi_\theta)$, according to (18), we have $\pi_\theta^h(\boldsymbol{a}|\boldsymbol{s}) = \pi_\theta(\boldsymbol{a}|\boldsymbol{s})$. Thus, we get that

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} J'(\boldsymbol{\theta}) &= \int_\tau \pi_\theta^h(\tau)\nabla_{\boldsymbol{\theta}}\log\pi_\theta(\tau)R(\tau)d\tau\\ &= \int_\tau \pi_\theta(\tau)\nabla_{\boldsymbol{\theta}}\log\pi_\theta(\tau)R(\tau)d\tau\\ &= \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}).\end{aligned}$$

Therefore, according to [10], $\boldsymbol{\theta}$ can converge to a local optimum. □

According to Proposition 2, the optimality of the TAG mechanism for policy-based DRL is guaranteed. From the above, we can combine the proposed TAG mechanism and policy-based DRL methods using $\nabla_{\boldsymbol{\theta}} J'(\boldsymbol{\theta})$, which is shown in Algorithm 1.

Algorithm 1 TAG for Policy-Based DRL

Input: Demonstration data $\mathcal{D}_d$. Initialize parameters $\boldsymbol{\theta}$ of the policy network and $\boldsymbol{\varphi}$ of the critic network. Initialize $\kappa(\pi_\theta)$, T_r, T_{gp}, β, and a data buffer $\mathcal{D}$.

Procedure:

```
Build the teacher model with D_d.
for each epoch do
    // Collect data
    for each step do
        Observe state s.
        Sample from U(0, 1) and get ξ.
        Get σ²(s) and get a_g with Eq. (14,15).
        if κ(π_θ) < T_r, σ²(s) < T_gp and ξ < β then
            Let a = a_g.        ▷ Adopt teacher's suggestion
        else
            Sample from π_θ(·|s) and get a_p.
            Let a = a_p. ▷ Action from the original policy
        end if
        The agent takes action a.
        Get next state s', reward r, and done signal d.
        Store (s, a, r, s', d) into D.
    end for
    // Update the policy
    Calculate ∇_θ J'(θ) using data in D.
    Update θ with gradient ascent.
    Update φ using MSE loss with gradient descent.
    Clear buffer D.
    Test π_θ in the test environment and get κ(π_θ).
end for
```

Output: π_θ.

E. π_θ^h for Value-Based DRL

In this section, we introduce the TAG mechanism into value-based DRL algorithms. For value-based DRL algorithms, the goal of the policy network is to find a policy that can maximize the Q value

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{s}}\left[\hat{Q}_{\boldsymbol{\varphi}}(\boldsymbol{s}, \mu_\theta(\boldsymbol{s}))\right]. \tag{23}$$

According to Proposition 1, with the TAG mechanism, trajectories in $\boldsymbol{\tau}_g$ will appear in the replay buffer with a greater probability. Hence, the teacher-advice mechanism will also benefit the training process of value-based DRL algorithms. In addition, in the process of Q-iteration, it directly iterates on the optimum Q value, which is independent of the policy [44]. Therefore, for value-based DRL algorithms with the TAG mechanism, the estimation of the value function is still unbiased. In addition, to achieve better performance, we also introduce a reward-shaping technique into the TAG mechanism, which is designed as

$$\begin{aligned}r^+(\boldsymbol{s},\boldsymbol{a}) = r(\boldsymbol{s},\boldsymbol{a})\\ + c * \mathbb{1}\left\{\sigma^2(\boldsymbol{s}) < T_{\text{gp}}, \xi < \beta, \kappa(\pi_\theta) < T_r\right\}\end{aligned} \tag{24}$$

Algorithm 2 TAG for Value-Based DRL

Input: Demonstration data $\mathcal{D}_d$. Initialize parameters $\boldsymbol{\theta}$ of the policy network and $\boldsymbol{\varphi}$ of the Q network. Initialize $\kappa(\pi_\theta)$, T_r, T_{gp}, β, and a data buffer $\mathcal{D}$.

Procedure:

```
Build the teacher model with D_d.
for each epoch do
    // Collect data
    π_θ^h interacts with the environment and get a mini-batch data d = {s_i, a_i}_{i=1,2,...}.
    Reshape rewards in d using Eq. (24).
    Store data d in D.
    // Update the policy
    Sample a batch data from D to train θ and φ.
    Test π_θ in the test environment and get κ(π_θ).
end for
```

Output: π_θ.

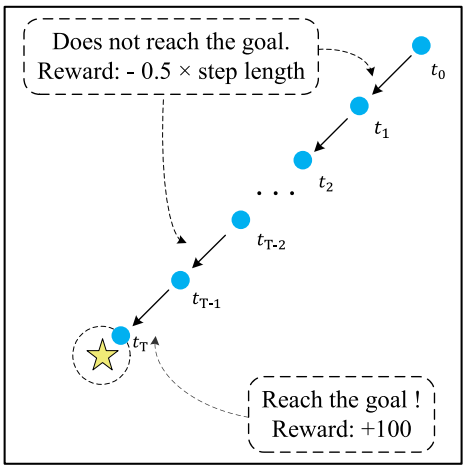

Fig. 4. In this environment, the agent's goal is to approach the target with the shortest number of steps. The reward in the environment is sparse, and the agent can only receive a positive reward when reaching the goal. The size of the environment can be changed to vary the difficulty of reaching the goal.

where c is a constant used to reward the agent for taking favorable actions. From the above, the pseudocode is shown in Algorithm 2.

V. Experiments

In this section, we evaluate the TAG mechanism in two types of environments: sparse-reach and delayed reward robot locomotion environments. The former are characterized by sparse reward, and the latter are characterized by delayed reward and complicated locomotion control. In these environments, the proposed TAG mechanism is critical. From the experiments, we aim to investigate the following research questions (RQs).

1) *RQ1:* Can the proposed TAG mechanism effectively accelerate the training process of different DRL algorithms?
2) *RQ2:* Under the same demonstration data, can our method attain better performance versus the counterparts (e.g., existing LfD methods) that also utilize demonstration data?
3) *RQ3:* What is the key ingredient in our method that introduces better empirical results? Is the introduction of Gaussian process necessary?
4) *RQ4:* Can our method achieve better results in complex environments, such as delayed reward robot locomotion environments?

A. Experimental Configuration

1) Environment: The sparse-reach environment, shown in Fig. 4, is a 2-D target approaching environment. The ultimate goal of the agent is to reach the target point. The key elements in the environment are defined as follows.

1) *State:* A 4-D vector represents the location of the target point and the location of the agent.
2) *Action:* A 2-D vector denotes the step size in x- and y-directions. The step length cannot exceed 1.
3) *Reward:* The positive reward is very sparse. Only when the agent reaches the target point, can the agent receive a positive reward signal. Otherwise, the agent will receive a negative reward $-0.5 \times$ step length.

In addition, the mujoco environment [45], a physics engine aiming to facilitate research in robotics, was also used to test the performance of algorithms. In particular, modified versions of several continuous control environments in mujoco, named delayed-mujoco, were leveraged in this article. In delayed-mujoco, the reward signal is given to the agent only after a specific number of steps.

2) Hyperparameter: For sparse-reach environments, the hyperparameters of the guided policy are given as follows: $\beta = 0.5, T_{\text{gp}} = 0.04$, and $T_r = 80$. For delayed-mujoco environments, the hyperparameters of the guided policy are $\beta = 0.5, T_{\text{gp}} = 0.0025$, and $T_r = 0.8 \times r_{\max}$, where $r_{\max}$ is the maximum estimated episode return in the demonstrations in an environment. In all RL algorithms, $\gamma = 0.99$, and the number of steps in an epoch is ten times the max episode length of the environment. For all networks, the optimizer is selected as Adam [46] with a learning rate of 0.001.

3) Implementation: For a fair comparison, policy networks of different methods in each comparative experiment have the same hidden layer structure. For sparse-reach environments, demonstration data were collected manually. Only 50 state–action pairs were collected for demonstrations. For delayed-mujoco environments, demonstration data were collected by the best-trained agent in the dense reward setting. Only 1000 state–action pairs were used to build the teacher model. In addition, the squared exponential kernel was used in sparse-reach environments, and the Matern kernel was used in delayed-mujoco environments. Moreover, all experiments were carried out on a workstation with Intel Xeon Gold 6240 CPU, NVIDIA RTX 3090 GPU, and 256-GB RAM. PyTorch, a machine learning framework, was used to implement the training of neural networks [47].

B. Sparse-Reach Experiment

RQ1: To answer the first question, we applied the proposed TAG mechanism to four existing state-of-the-art DRL algorithms, i.e., VPG [13], PPO [15], DDPG [19], and SAC [21], in sparse-reach environments; they are named TAG-VPG, TAG-PPO, TAG-DDPG, and TAG-SAC, respectively. In addition, the experimental results were compared with the above four original DRL methods without the TAG

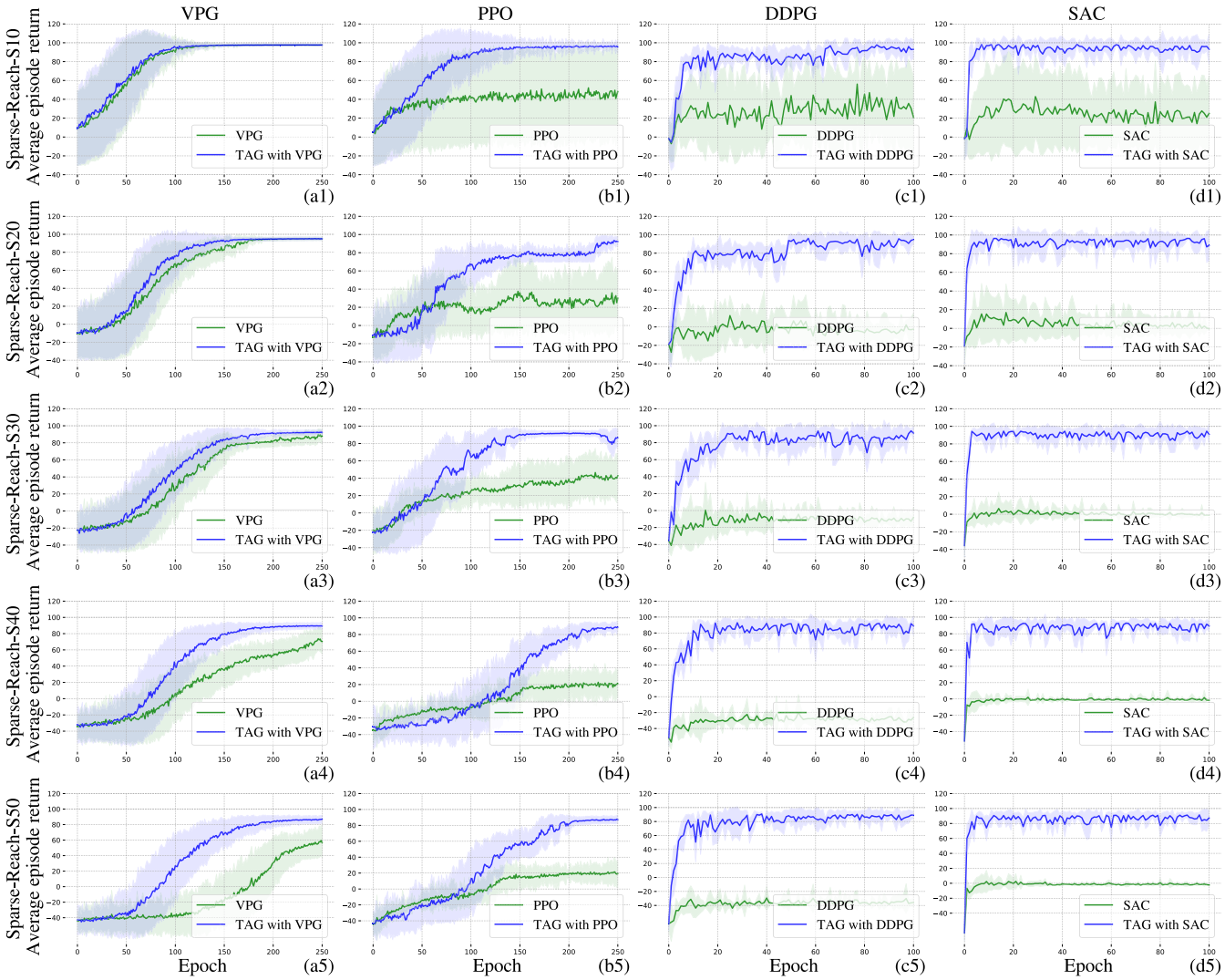

Fig. 5. Average episode return per epoch of the proposed TAG mechanism with four benchmark algorithms versus the benchmark algorithms in five different sizes of sparse-reach environment. The size of the environment is 10×10, 20×20, 30×30, 40×40, and 50×50. (a1–a5) Comparison of VPG and VPG with TAG. (b1–b5) Comparison of PPO and PPO with TAG. (c1–c5) Comparison of DDPG and DDPG with TAG. (d1–d5) Comparison of SAC and SAC with TAG.

TABLE I

AVERAGE EPISODE RETURN COMPARISON OF TAG MECHANISM WITH FOUR BENCHMARK DRL ALGORITHMS VERSUS THE BENCHMARK DRL ALGORITHMS IN FIVE DIFFERENT SIZE SPARSE-REACH ENVIRONMENTS

Task	VPG	TAG-VPG	PPO	TAG-PPO	DDPG	TAG-DDPG	SAC	TAG-SAC
SR-S10	**97.68**±0.95	97.60±1.03	48.18±49.87	**95.68**±7.28	20.89±43.78	**92.97**±9.42	24.99±39.77	**93.10**±16.12
SR-S20	**94.97**±2.06	94.88±1.90	29.99±32.97	**92.21**±4.91	-2.35±17.09	**94.77**±4.56	-0.58±0.11	**88.94**±16.18
SR-S30	88.06±7.77	**92.12**±2.96	42.99±19.54	**86.62**±9.45	-8.09±11.50	**91.32**±7.74	-1.21±0.05	**90.85**±11.45
SR-S40	70.30±15.05	**89.41**±3.82	21.12±16.68	**88.71**±4.06	-26.09±16.02	**88.92**±12.41	-1.67±0.18	**89.49**± 6.92
SR-S50	56.53±19.10	**86.92**±4.99	19.46±14.22	**87.05**±5.05	-35.75±5.63	**89.16**±4.69	-2.25±0.52	**87.38**±10.38

mechanism. Moreover, five different sizes of the sparse-reach environment were used to evaluate the effectiveness of the TAG mechanism. Fig. 5 shows the result of this setting. In Fig. 5, after each epoch of training, we evaluate the policy network π_θ in the test environment and get ten trajectories. Then, the average episode return of these ten trajectories was calculated as the evaluation result, corresponding to the y-axis in the figure.

It can be observed from Fig. 5 that for VPG, it has almost the same learning curve as TAG-VPG when the environment size is 10×10, as shown in Fig. 5(a1). As the environment size increases, the convergence rate of both VPG and TAG-VPG decreases to different degrees, as shown in Fig. 5(a). However, the convergence rate of VPG dropped significantly. The reason for this phenomenon is that, as the size of the environment becomes larger, it becomes more difficult for the agent to reach the target point. More specifically, the positive rewards in the environment are very sparse. As the size of the environment increases, the probability of the agent getting high rewards from the environment becomes smaller, and the favorable gradient directions become less obvious, causing the convergence rate to become slower. However, compared to VPG, the convergence rate of TAG-VPG decreases to a much lesser

extent. This is because, due to the intervention of the teacher model, during the exploration process, the probability of obtaining a high-reward trajectory is higher. As a consequence, the direction of the favorable gradient is still dominant, which makes the algorithm converge faster. Consequently, TAG-VPG is less sensitive to the size of the environment.

For PPO, due to the gradient truncation mechanism, the training speed of PPO is inferior to VPG, as shown in Fig. 5(b). In the experiment, we discovered that for mini-batch data with high-reward trajectories, its gradient would be clipped by the truncation mechanism. On the contrary, for those general batch data, PPO will update normally. The above are the main reasons for the slow training process of PPO. Likewise, same as VPG, as the size of the environment increases, the convergence rate of PPO also decreases. Furthermore, TAG-PPO also suffered from the same problem as PPO that the training speed is slower than its counterpart TAG-VPG. In addition, the gradient clip mechanism also leads to a greater variance of the agent's performance, as shown in Fig. 5(b1)–(b3) and Table I.

For DDPG, it cannot achieve good performance in any size of sparse-reach environment, shown in Fig. 5(c). Intuitively, the primary reason is that in DDPG, the Q value is estimated with single-step rewards, i.e., $r_t + \gamma \max_{\boldsymbol{a}} \hat{Q}(\boldsymbol{s}, \boldsymbol{a})$, which means that the variance is small, but the estimate is biased. Furthermore, in the sparse-reach environment, when the agent does not reach the goal, at each step, it receives a negative reward signal as a penalty. Therefore, most of the reward signals are negative in the replay buffer. This feature makes the Q value estimated inaccurate. Under such circumstances, the agent tends to move with a very short step to avoid negative rewards, which can be shown in Fig. 5(c2) and (c3), where the performance approaches 0. However, in VPG or PPO, the critic is estimated with multistep rewards, e.g., $\sum_{t=0}^{T} \gamma^t r_t$, which is unbiased. Therefore, the performance of VPG and PPO will slowly improve until convergence. As for TAG-DDPG, with the intervention of the teacher model, many high-reward state–action pairs appear in the replay buffer. The performance of the TAG-DDPG improves rapidly in the training process, as shown in Fig. 5(c).

For SAC, it suffers from the same problem as in DDPG. In order to maximize the episode return and avoid penalty, the agent tends to move with a short step length, and the performance ultimately converges to 0, which can be seen in Fig. 5(d). This means that SAC falls into a local optimum solution. However, equipped with the teacher-advice mechanism, TAG-SAC helps the policy network jump out of the local optimal solution and almost reach the optimal solution by better exploration. In addition, the convergence speed of TAG-SAC is even faster than that of TAG-DDPG.

From the above, we can conclude that the TAG mechanism can accelerate the training process of different DRL algorithms in the sparse-reach environment with varied sizes by better exploration. Table I shows the performance of each method in the last epoch of training. It can be observed that the TAG mechanism greatly improves the performance of the DRL algorithms in the sparse-reach environment. Furthermore, algorithms with the TAG mechanism have a smaller variance in most cases.

RQ2: To answer the second question, we compared the TAG-SAC method with other existing algorithms, including the following.

1) *SAC [21]:* A state-of-the-art DRL method based on the maximum entropy RL.
2) *SACfD:* This method combines the SAC and LfD mechanisms that are utilized to employ demonstration data in DQNfD [30] and OERL [31].
3) *GAIL [26] With SAC:* A classic imitation learning method based on GAN, which clones the expert behavior by matching the occupancy measure between the expert policy and the learned policy. For a fair comparison, the generator part is selected as SAC.
4) *VAIL [27] With SAC:* An improved version of GAIL, the IB, is introduced to avoid the discriminator dominating during the training process, causing the gradient passed to the generator that is close to 0. For a fair comparison, the generator part is selected as SAC.

The results of the comparative experiment are shown in Fig. 6. It can be observed that in the sparse-reach environment with a size of 10, TAG-SAC, VAIL, GAIL, and SACfD achieve a certain effect, while TAG-SAC attains the best performance, as shown in Fig. 6(a). However, as the size of the environment becomes larger, the performance of these algorithms decreases to varying degrees, as shown in Table II.

For SAC, in small size environments, such as SR-S10 and SR-S20, the learning curve rises in the early stages of training. However, as the number of interactions increases, the agent is inclined to avoid punishment, causing the learning curve to decline. Furthermore, for larger size environments, the learning curve converges to 0. For SACfD, it suffers from the same problem as SAC. As the training progresses, the learning curve in SACfD has a certain drop, which can be seen in Fig. 6(a)–(c). For VAIL with SAC and GAIL with SAC, because of the introduction of IBs, VAIL achieved better performance than GAIL in these five sparse-reach environments. Due to the limited demonstration data, as the size of the environment increases, the performance of both methods drops drastically, as shown in Fig. 6(c)–(e). In addition, it can be seen from Table II that VAIL with SAC outperforms SACfD in small-size environments, such as SR-S10, SR-S20, and SR-S30. However, for larger size environments, e.g., SR-S30 and SR-S40, SACfD outperforms VAIL with SAC. A reasonable explanation is that VAIL with SAC can only learn from the demonstration data and discards the reward information given by the environment. However, when confronted with insufficient demonstration data, SACfD can also learn from the agent's exploration to achieve better performance.

As for TAG-SAC, it achieves the best performance in sparse-reach environments of different sizes. Due to the powerful generalization ability of the Gaussian process, the teacher model can guide the agent at more states. Thus, as the environment size increases, the performance of TAG-SAC does not decrease significantly, and it can achieve better performance than its counterparts, which can be shown in Table II. Therefore, we can conclude that TAG-SAC outperforms its counterparts that also utilize demonstration data.

RQ3: To answer the third question, we focused on the role of the Gaussian process in our teacher-advice mechanism. In order to verify the effectiveness of the Gaussian process, we conducted an extra experiment, that is, we replaced the Gaussian process with calculating the Euclidean distance between states. More specifically, we judged whether a state $\boldsymbol{s}$ should be guided by calculating the distance between state $\boldsymbol{s}$ and all states in the demonstration data. Then, the new guided

TABLE II

AVERAGE EPISODE RETURN COMPARISON OF THE TAG-SAC WITH OTHER BENCHMARKS ALGORITHMS IN FIVE DIFFERENT SIZE SPARSE-REACH ENVIRONMENTS

Task	SAC	SAC from Demonstration	GAIL with SAC	VAIL with SAC	TAG-SAC
SR-S10	24.99±39.77	62.92±32.13	79.38±17.17	83.39±25.81	**93.10**±16.12
SR-S20	-0.58±0.11	39.12±46.83	38.62±36.55	39.33±42.37	**88.94**±16.18
SR-S30	-1.21±0.05	30.95±40.94	12.50±30.36	31.45±38.79	**90.85**±11.45
SR-S40	-1.67±0.18	25.11±34.29	0.43±17.05	8.46±26.18	**89.49**±6.92
SR-S50	-2.25±0.52	15.08±31.85	1.67±21.18	2.08±24.45	**87.38**±10.38

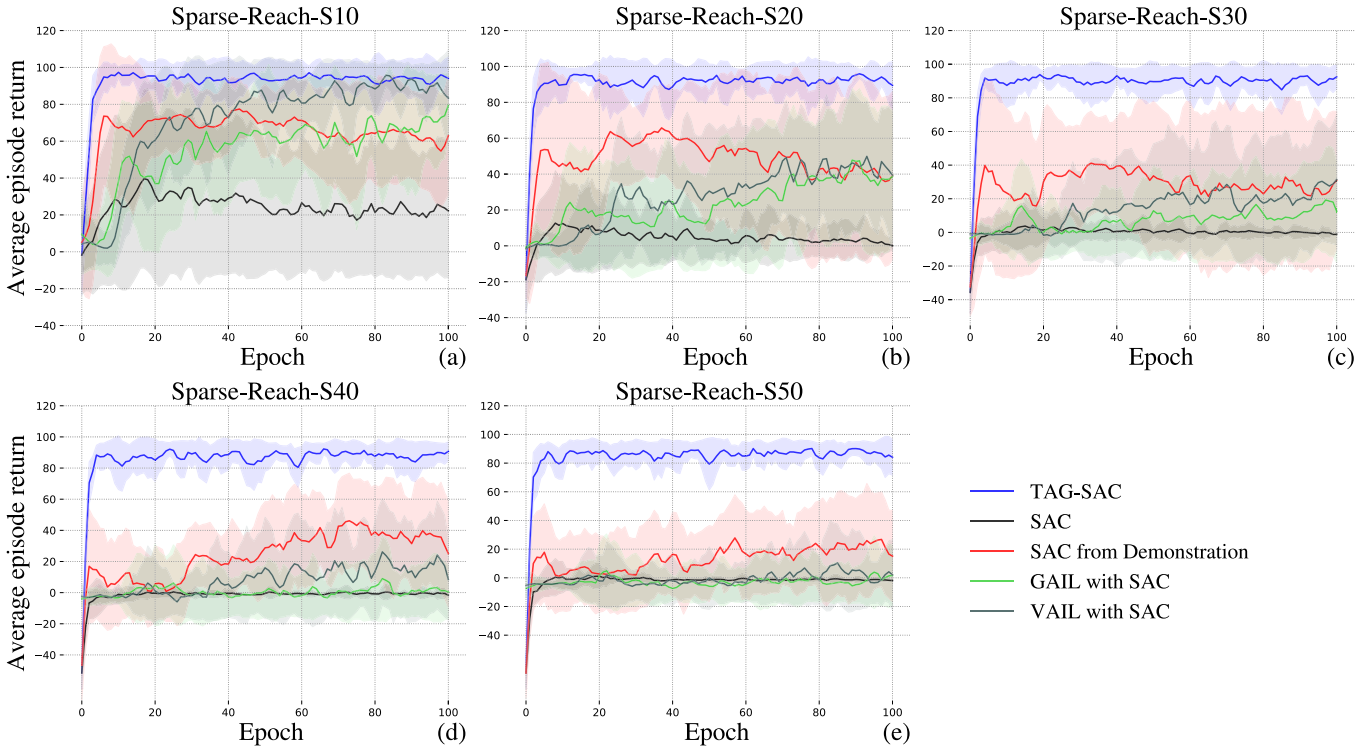

Fig. 6. Average episode return per epoch of the TAG-SAC versus the four benchmark algorithms in five different size sparse-reach environments. For a clearer presentation, each curve was smoothed with a sliding window of length 3. (a–e) Comparison in sparse-reach environments with the size of 10, 20, 30, 40, and 50, respectively.

TABLE III

AVERAGE EPISODE RETURN COMPARISON BETWEEN GP, SAC, TAE-SAC, AND TAG-SAC

*	GP	SAC	TAE-SAC	TAG-SAC
1	**98.76**±0.89	24.99±39.77	69.11±32.66	93.10±16.12
2	**97.22**±2.00	-0.58± 0.11	39.18±43.66	88.94±16.18
3	45.44±49.86	-1.21±0.05	4.74±19.88	**90.85**±11.45
4	43.91±50.93	-1.67±0.18	1.08±14.31	**89.49**± 6.92
5	22.13±46.88	-2.25±0.52	0.47±12.45	**87.38**±10.38

* The labels on the left represent the environment size of 10, 20, 30, 40, and 50, respectively.

policy using the Euclidean distance can be formulated as

$$\pi_\theta^{h_{\text{eu}}}(\boldsymbol{s}) = \begin{cases} \boldsymbol{a}^{i_*}, & \min\limits_i \left\| \boldsymbol{s} - \boldsymbol{s}_i^d \right\|_2 \le T_{\text{eu}} \\ \pi_\theta(\boldsymbol{s}), & \text{else} \end{cases} \tag{25}$$

where $\boldsymbol{s}_i^d$ indicates the ith state in demonstration data, $i_* = \arg\min_i \|\boldsymbol{s} - \boldsymbol{s}_i^d\|_2$, and T_{eu} is a threshold.

We employed the Euclidean distance version of the teacher-advice mechanism in SAC, named TAE-SAC, and compared it with the original SAC and TAG-SAC in the sparse-reach environment. The result is shown in Table III. It can be observed that in a small size environment, the teacher-advice mechanism that calculates the Euclidean distance can also achieve good results. However, as the size of the environment increases, its effect gradually decreases. This is because in large size environments, restricted by limited demonstration data, the teacher model cannot guide the agent adequately. Meanwhile, the teacher-advice mechanism that uses the Gaussian process has little performance degradation as the environment size grows. In addition, we also test the teacher model, named GP, in the environment to observe its performance. The result is shown in Table III. GP has the best performance in small size environments, i.e., SR-S10 and SR-S20. This is because, for small size environments, the demonstration is relatively sufficient for Gaussian process. However, in large size environments, the performance of the GP decreases due to the inability to learn from the environment.

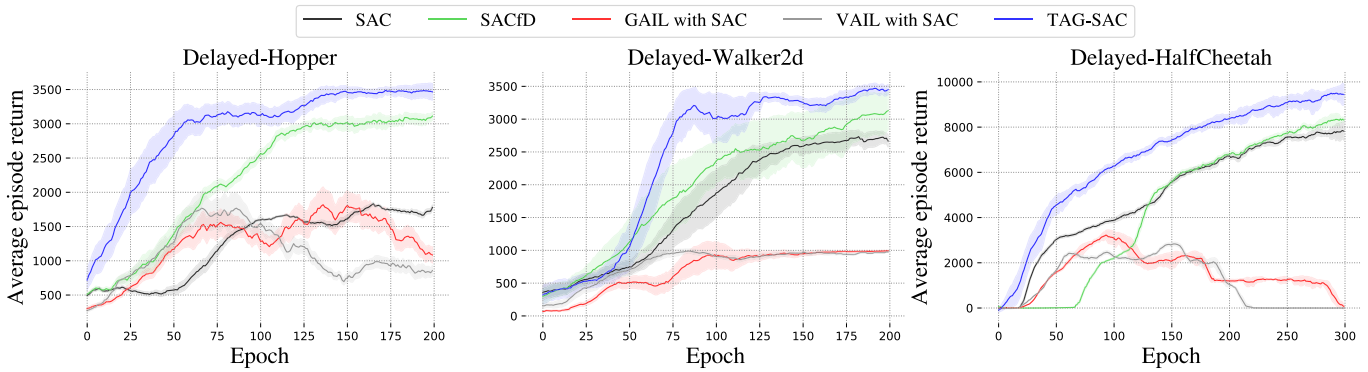

Fig. 7. Average episode return per epoch of the TAG-SAC versus the four algorithms in three delayed-mujoco environments. For a clearer presentation, each curve was smoothed with a sliding window. Each epoch contains 5000 steps.

TABLE IV
AVERAGE EPISODE RETURN COMPARISON OF THE TAG-SAC WITH OTHER BENCHMARKS ALGORITHMS IN THREE DIFFERENT DELAYED-MUJOCO ENVIRONMENTS

Task	SAC	SAC from Demonstration	GAIL with SAC	VAIL with SAC	TAG-SAC
Delayed-Hopper	1778.77±36.17	3112.59±58.40	1085.04±118.95	857.15±75.06	**3465.86**±121.20
Delayed-Walker2d	2666.27±91.66	3130.86±288.60	992.31±8.89	975.07±4.59	**3450.20**±79.49
Delayed-HalfCheetah	7820.57±399.70	8315.40±102.19	59.85±96.15	-1.82±1.59	**9441.17**±456.89

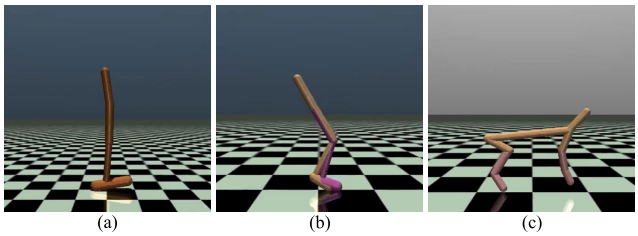

Fig. 8. Delayed-mujoco locomotion environments. (a) Hopper, state-space dimension $|\mathcal{S}|$ is 11 and action-space dimension $|\mathcal{A}|$ is 3. The delayed reward interval h is 25 steps. (b) Walker2d, $|\mathcal{S}| = 17, |\mathcal{A}| = 6$, and $h = 20$. (c) HalfCheetah, $|\mathcal{S}| = 17, |\mathcal{A}| = 6$, and $h = 10$.

Therefore, we can conclude that with the help of powerful data generalization capabilities, the Gaussian process plays a vital role in our teacher-advice mechanism, which helps the teacher model to make full use of the demonstrations. Furthermore, RL algorithms endow the agent with the ability to learn from environments, which makes TAG-SAC achieve the best performance in larger size environments.

C. Delayed-Mujoco Locomotion Experiment

RQ4: To answer the fourth question, we also compared the TAG-SAC with other benchmark algorithms in four delayed-mujoco locomotion control environments, such as delayed-hopper, delayed-walker2d, and delayed-halfcheetah. Different from the dense rewards setting in default mujoco locomotion control environments, the reward signal given by the delayed version is also sparse. In these environments, the agent cannot get a reward signal at every step but obtain a reward after every h step. The experimental results are shown in Fig. 7. The details of the environments are shown in Fig. 8. Since only 1000 state–action pairs are utilized, neither GAIL-SAC nor VAIL-SAC can achieve good performance in these complicated control environments. Moreover, limited by delayed rewards, SAC can only obtain poor results. However, due to the introduction of expert demonstrations, compared with SAC, SACfD achieves better performance in these three environments, especially in the delayed-hopper environment. Finally, the TAG-SAC achieves the fastest convergence speed versus its counterparts in these three control environments shown in Fig. 7 and attains the best performance, which can be shown in Table IV.

Hence, it can be concluded that the proposed method can also achieve better results in complex delayed reward robot locomotion control environments.

VI. CONCLUSION AND DISCUSSION

In this article, we proposed a TAG for RL algorithms, which can make the agent learn skills efficiently with only a small number of expert demonstrations. In TAG, a Gaussian process is utilized to build a teacher model, and a guided policy is introduced to select actions between the teacher model and the original policy. Due to the TAG mechanism, the agent is more inclined to explore high-return regions, making the RL algorithm converge faster and attain better performance. In addition, with the powerful generalization capability of the Gaussian process, the teacher model can be constructed with only a small number of demonstrations. Also, the confidence value output by the teacher model makes the guided policy guide the agent more accurately. Experimental results on sparse reward environments illustrate that the TAG mechanism can make normal DRL algorithms obtain better performance and faster convergence speed. Moreover, the proposed TAG-SAC attains the best performance over other LfD counterparts on several delayed reward and complicated locomotion control environments.

It is worth noting that the TAG mechanism processes the properties of generality, which means that it can be combined with common DRL algorithms and other techniques that also utilize demonstrations. Therefore, the proposed TAG mechanism can be applied in many industrial scenarios. For example, DRL has been used to control robots [6], [7], so our method can accelerate the process of robot skill

acquisition. In addition, there are also cases of DRL in the field of industrial scheduling, i.e., smart grid scheduling [48] and logistics dispatch [49]. As a consequence, the TAG mechanism is also helpful for these methods to obtain intelligent scheduling policies.

Notably, introducing the TAG mechanism in the agent exploration process will bring additional computation to get the advice action and the corresponding confidence. When the demonstration data are small, the extra computation is usually acceptable. However, when the demonstration data are large, it may be necessary to introduce the local Gaussian process [50] to reduce the computational load of the TAG mechanism. On the other hand, in our TAG mechanism, the confidence threshold T_{gp} in the guided policy is a hyperparameter that needs to be adjusted for different environments. If T_{gp} is set too high, the agent may be misled by the wrong actions given by the teacher model. On the contrary, if it is set too low, the agent may not be guided by the teacher model.

Further exploration of applying the TAG mechanism to a real robot control task could be a new direction for future work.

APPENDIX

A. Dataset Size Selection Guideline

To determine the appropriate number n of trajectories in the demonstration according to different environments, this section gives basic guidance.

Generally speaking, the higher the dimensionality of the state space in the environment, the more demonstrations are needed. If n is larger, the generalization ability of the Gaussian process model will be stronger, but it will bring a larger amount of computation. On the contrary, the computational complexity of the model will be smaller, but the generalization ability of the model will be reduced accordingly. A practical scheme for determining the appropriate n according to the environment is given as follows:

1) Use m demonstration trajectories to fit a Gaussian process model $\mathcal{GP}$.
2) Combine this $\mathcal{GP}$ model with a random policy into a hybrid policy, as described in (16). Let this hybrid policy explore in the environment and count the number of times the agent has been guided. If the guided ratio exceeds a certain threshold, such as 20%, the $\mathcal{GP}$ model is considered to have a certain generalization ability, and a suitable n is obtained. Otherwise, the number of trajectories needs to be increased, and we need to go to step 1.

B. TAG for More Complex Environments

In order to enable the proposed method to be used in more complex environments, some further measures are listed as follows.

1) For complex environments, the dimension of the state space is usually higher. To make the teacher model more effective, more demonstration data are usually needed to build a more powerful Gaussian process model.
2) For more complex environments, it may be necessary to introduce the local Gaussian process [50] to enable the more powerful Gaussian process model to perform faster calculations.

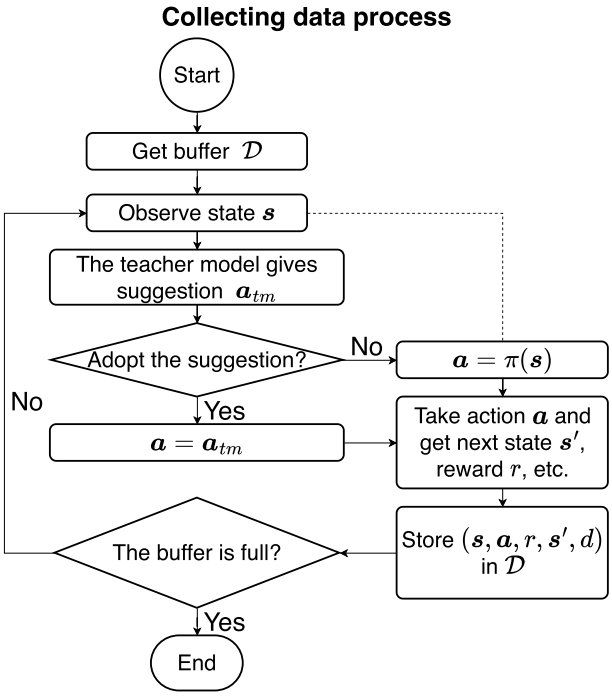

Fig. 9. Flowchart of the process of collecting data.

C. Hyperparameter Selection Guideline

The basic principles for the selection of hyperparameters in the TAG mechanism are given as follows.

T_{gp}: This parameter is mainly related to the sensitivity of state transitions in the environment to action perturbations. If the state transition is more sensitive to action perturbations, then T_{gp} should be smaller. To determine the value of T_{gp}, we can use the demonstrations to build a teacher model and extract a trajectory from the demonstrations. Then, we can try to make the teacher model roll out the trajectory. Then, we can add noise to the input of the teacher model until the teacher model cannot roll out the trajectory. At this time, we can know that the value of T_{gp} should not exceed the given noise level. Generally, T_{gp} should be set relatively small to ensure that the teacher model can guide the agent correctly.

β: This parameter determines the probability that the agent will adopt the teacher model's suggestion when $\sigma^2(\boldsymbol{s}) < T_{\text{gp}}$. The choice of β is related to the quality and quantity of the demonstration data and the reward design in the environment. If the average cumulative reward of trajectories in the demonstration is high, β should become larger so that the agent can be guided by the teacher model with a greater probability. In addition, if the reward of the environment is well-designed and easy to explore, this value should be reduced accordingly to ensure that the agent can explore in the environment with a higher probability.

T_r: When the average cumulative reward obtained by the agent exceeds T_r, the agent will no longer accept the suggestions of the teacher model. Therefore, this parameter determines when the agent chooses to explore in the environment by itself to get higher rewards. To prevent the teacher model from interfering excessively with the behavior of the agent in the later stage of training, T_r can usually be set to $0.8\times$ the average cumulative reward of trajectories in the demonstration data.

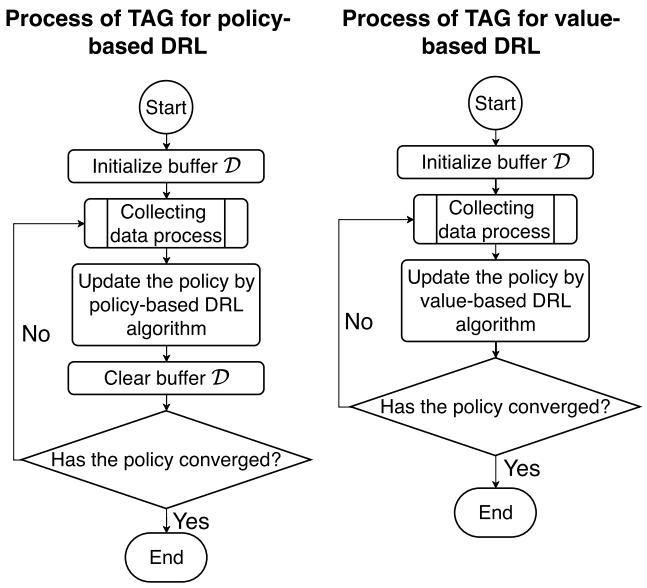

Fig. 10. Flowcharts of the proposed algorithms.

γ: The discount factor is usually set to a number close to 1 to ensure that future rewards are fully considered. However, for the convergence of the RL algorithm, γ cannot be set to 1. Therefore, the discount factor is usually set to 0.99.

D. Flowcharts of the Proposed Algorithms

To further illustrate the structure of the presented algorithms, we first give the subprocess for collecting data in Fig. 9. Then, the flowcharts of the process of TAG for policy-based and value-based DRL algorithms are shown in Fig. 10.

REFERENCES

[1] D. Silver et al., "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

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

[3] W. Shang, Y. Yu, Q. Li, Z. Qin, Y. Meng, and J. Ye, "Environment reconstruction with hidden confounders for reinforcement learning based recommendation," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 566–576.

[4] J. Shi, Y. Yu, Q. Da, S. Chen, and A. Zeng, "Virtual-Taobao: Virtualizing real-world online retail environment for reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 4902–4909.

[5] L. Wang, W. Zhang, X. He, and H. Zha, "Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 2447–2456.

[6] T. Johannink et al., "Residual reinforcement learning for robot control," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 6023–6029.

[7] F. Niroui, K. Zhang, Z. Kashino, and G. Nejat, "Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 610–617, Apr. 2019.

[8] B. R. Kiran et al., "Deep reinforcement learning for autonomous driving: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 4909–4926, Jun. 2022.

[9] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 653–664, Mar. 2017.

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

[11] R. S. Sutton et al., "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, 1999, pp. 1057–1063.

[12] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

[13] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proc. Int. Conf. Learn. Represent.*, San Juan, Puerto Rico, 2016. [Online]. Available: https://arxiv.org/pdf/1506.02438.pdf

[14] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, Jun. 2015, pp. 1889–1897.

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

[16] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Jan. 2015.

[17] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 2094–2100.

[18] Z. Wang et al., "Dueling network architectures for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1995–2003.

[19] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, San Juan, Puerto Rico, 2016. [Online]. Available: https://arxiv.org/pdf/1509.02971.pdf

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

[21] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.

[22] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auton. Syst.*, vol. 57, no. 5, pp. 469–483, May 2009.

[23] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2011, pp. 627–635.

[24] W. Sun, A. Venkatraman, G. J. Gordon, B. Boots, and J. A. Bagnell, "Deeply AggreVaTeD: Differentiable imitation learning for sequential prediction," in *Proc. Int. Conf. Mach. Learn.*, Jul. 2017, pp. 3309–3318.

[25] I. Goodfellow et al., "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014. [Online]. Available: https://papers.nips.cc/paper_files/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html

[26] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 4565–4573.

[27] X. B. Peng, A. Kanazawa, S. Toyer, P. Abbeel, and S. Levine, "Variational discriminator bottleneck: Improving imitation learning, Inverse RL, and GANs by constraining information flow," in *Proc. Int. Conf. Learn. Represent.*, New Orleans, LA, USA, 2019. [Online]. Available: https://openreview.net/pdf?id=HyxPx3R9tm

[28] X. Zhang, Y. Li, Z. Zhang, and Z.-L. Zhang, "f-GAIL: Learning f-divergence for generative adversarial imitation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 12805–12815.

[29] S. K. S. Ghasemipour, R. Zemel, and S. Gu, "A divergence minimization perspective on imitation learning methods," in *Proc. Conf. Robot Learn.*, May 2020, pp. 1259–1277.

[30] T. Hester et al., "Deep Q-learning from demonstrations," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 3223–3230.

[31] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *Proc. ICRA*, May 2018, pp. 6292–6299.

[32] B. Kang, Z. Jie, and J. Feng, "Policy optimization with demonstrations," in *Proc. Int. Conf. Mach. Learn.*, Jul. 2018, pp. 2469–2478.

[33] M. Jing et al., "Reinforcement learning from imperfect demonstrations under soft expert guidance," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 5109–5116.

[34] I. Kostrikov, K. K. Agrawal, D. Dwibedi, S. Levine, and J. Tompson, "Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning," in *Proc. Int. Conf. Learn. Represent.*, New Orleans, LA, USA, 2019. [Online]. Available: https://openreview.net/pdf?id=Hk4fpoA5Km

[35] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *J. Mach. Learn. Res.*, vol. 16, no. 42, pp. 1437–1480, Aug. 2015.

[36] S. Schaal et al., "Learning from demonstration," in *Proc. Adv. Neural Inf. Process. Syst.*, 1997, pp. 1040–1046.

[37] G. Xiang and J. Su, "Task-oriented deep reinforcement learning for robotic skill acquisition and control," *IEEE Trans. Cybern.*, vol. 51, no. 2, pp. 1056–1069, Feb. 2021.

[38] G. Liu et al., "Demonstration actor critic," *Neurocomputing*, vol. 434, pp. 194–202, Apr. 2021.
[39] M. Alshiekh et al., "Safe reinforcement learning via shielding," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 2669–2678.
[40] R. Cheng, G. Orosz, R. Murray, and J. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 3387–3395.
[41] D. Neider, J.-R. Gaglione, I. Gavran, U. Topcu, B. Wu, and Z. Xu, "Advice-guided reinforcement learning in a non-Markovian environment," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 9073–9080.
[42] M. B. Radac and R. E. Precup, "Data-driven model-free tracking reinforcement learning control with VRFT-based adaptive actor-critic," *Appl. Sci.*, vol. 9, no. 9, p. 1807, Apr. 2019.
[43] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.
[44] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
[45] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Vilamoura, Portugal, Oct. 2012, pp. 5026–5033.
[46] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, San Diego, CA, USA, 2015. [Online]. Available: https://arxiv.org/pdf/1412.6980.pdf
[47] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8024–8035.
[48] H.-M. Chung, S. Maharjan, Y. Zhang, and F. Eliassen, "Distributed deep reinforcement learning for intelligent load scheduling in residential smart grids," *IEEE Trans. Ind. Informat.*, vol. 17, no. 4, pp. 2752–2763, Apr. 2021.
[49] K. Li, T. Zhang, R. Wang, Y. Wang, Y. Han, and L. Wang, "Deep reinforcement learning for combinatorial optimization: Covering salesman problems," *IEEE Trans. Cybern.*, vol. 52, no. 12, pp. 13142–13155, Dec. 2022.
[50] R. B. Gramacy and D. W. Apley, "Local Gaussian process approximation for large computer experiments," *J. Comput. Graph. Statist.*, vol. 24, no. 2, pp. 561–578, 2015.

Ke Lin received the B.E. degree in mechanical engineering from the China University of Geosciences, Wuhan, China, in 2017, and the M.E. degree in mechanical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2020. He is currently pursuing the Ph.D. degree in control science and engineering with the Harbin Institute of Technology (Shenzhen), Shenzhen, China.

His research interests include deep reinforcement learning and its applications in autonomous driving.

Duantengchuan Li received the M.E. degree from the School of Computer Science and Technology, Central China Normal University, Wuhan, China, in 2021. He is currently pursuing the Ph.D. degree in software engineering with the School of Computer Science, Wuhan University, Wuhan.

His research interests include deep learning and reinforcement learning.

Yanjie Li (Member, IEEE) received the B.S. degree from Qingdao University (QDU), Qingdao, China, in 2001, and the Ph.D. degree from the University of Science and Technology of China (USTC), Hefei, China, in 2006.

From August 2006 to August 2008, he was a Research Associate with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology (HKUST), Hong Kong. In September 2008, he joined the Harbin Institute of Technology Shenzhen (HITSZ), Shenzhen, China, where he is currently an Associate Professor. His research interests include stochastic optimization and learning, Markov decision process (MDP), partially observable MDP, and reinforcement learning.

Dr. Li was a recipient of the Ho-Pan-Ching-Yi Best Paper Award in 2014.

Shiyu Chen received the B.E. degree in automation from the Anyang Institute of Technology, Anyang, China, in 2014, and the M.E. degree in control science and engineering from the Harbin Institute of Technology Shenzhen (HITSZ), Shenzhen, China, in 2017, where he is currently pursuing the Ph.D. degree in control science and engineering.

His research interests include learning-based control, deep reinforcement learning, and agile trajectory planning.

Qi Liu received the B.E. degree from the Lanzhou University of Technology, Lanzhou, China, in 2017, and the M.E. degree from the Harbin Institute of Technology, Harbin, China, in 2019. He is currently pursuing the Ph.D. degree in control science and engineering with the Harbin Institute of Technology (Shenzhen), Shenzhen, China.

His research interests include deep reinforcement learning and robotics.

Jianqi Gao received the master's degree from Northwestern Polytechnical University, Xi'an, China, in 2017. He is currently pursuing the Ph.D. degree in control science and engineering with the Harbin Institute of Technology (Shenzhen), Shenzhen, China, under the supervision of Prof. Yanjie Li.

His research interests lie in machine learning, reinforcement learning, combinatorial optimization, and multirobot system.

Mr. Gao won many honors, such as the National Scholarship, the First-class Academic Scholarship, and the Excellent Graduate Student.

Yanrui Jin received the Ph.D. degree from the School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2023.

His research interests include signal processing, fault diagnosis, biomedical engineering, and deep learning.

Liang Gong (Member, IEEE) received the B.Sc.(Eng.) degree from the Department of Mechanical Engineering, Hefei University of Technology, Hefei, China, in 2002, and the Ph.D. degree in mechanical engineering from Shanghai Jiao Tong University (SJTU), Shanghai, China, in 2010.

From 2007 to 2008, he worked as a Visiting Ph.D. Student with the Laboratory of Embedded Internet System, Luleå University of Technology, Luleå, Sweden. He is currently an Associate Professor with SJTU. His research interests include computational intelligence and cognitive science, agricultural robotics, wireless sensor networks, and motor drive power electronics.