IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 23 July 2023, accepted 11 August 2023, date of publication 15 August 2023, date of current version 22 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3305453

RESEARCH ARTICLE

Generative Adversarial Inverse Reinforcement Learning With Deep Deterministic Policy Gradient

MING ZHAN[1], JINGJING FAN[2], AND JIANYING GUO[3]
[1]School of Electrical and Control Engineering, North China University of Technology, Beijing 100144, China
[2]Intelligent Transportation Key Laboratory, North China University of Technology, Beijing 100144, China
[3]Tianjin Vocational Institute, Tianjin 300000, China

Corresponding author: Jingjing Fan (jjfan@ncut.edu.cn)

ABSTRACT Although the issue of sparse expert samples at the early stage of training in inverse reinforcement learning (IRL) is successfully resolved by the introduction of generative adversarial network (GAN), the inherent drawbacks of GAN result in ineffective generated samples. Therefore, we propose an algorithm for generative adversarial inverse reinforcement learning that is based on deep deterministic policy gradient (DDPG). We use the deterministic strategy to replace the random noise input of the initial GAN model and reconstruct the generator of the GAN based on the Actor-Critic mechanism in order to improve the quality of GAN-generated samples during adversarial training. Meanwhile, we mix the GAN-generated virtual samples with the original expert samples of IRL as the expert sample set of IRL. Our approach not only solves the problem of sparse expert samples at the early stage of training, but most importantly, it makes the decision-making process of IRL occurring under GAN more efficient. In the subsequent IRL decision-making process, we also analyze the differences between the mixed expert samples and the non-expert trajectory samples generated by the initial strategy to determine the best reward function. The learned reward function is used to drive the RL process positively for policy updating and optimization, on which further non-expert trajectory samples are generated. By comparing the differences between the new non-expert samples and the mixed expert sample set, we hope to iteratively arrive at the reward function and optimal policy. Performance tests in the MuJoCo physical simulation environment and trajectory prediction experiments in Grid World show that our model improves the quality of GAN-generated samples and reduces the computational cost of the network training by approximately 20% for each given environment, applying to decision planning for autonomous driving.

INDEX TERMS Inverse reinforcement learning, generative adversarial networks, deep deterministic policy gradient.

I. INTRODUCTION

Reinforcement Learning (RL) [1] is a potent paradigm that studies how the agent interacts with its surroundings during movement, learning the optimal strategy through continuous trial and error in a given environment. At the same time, we anticipate that the agent's interactions with the environment would always lead to optimal outcomes under different driving strategies. However, the reality of autonomous driving is much more complicated than anticipated. The difficulty of overcoming the sparse reward problem in the learning process means that the application of RL directly to the decision problem in autonomous driving has a limited impact, which is difficult to achieve the ideal state.

In contrast, inverse reinforcement learning (IRL) avoids the sparse reward problem of RL to a certain extent because it does not require reward feedback from the environment [2]. The core idea of IRL is to find the optimal policy distributed near the expert policy using the learned reward function so that it is close to or even consistent with the expert policy. Recent work has addressed many relevant concerns in the area of autonomous driving decision-making. The maximum entropy principle [3], [4], [5], [6] has been validated in IRL applications. They learn potential

The associate editor coordinating the review of this manuscript and approving it for publication was Junho Hong.

reward functions from human demonstrations to complete the autonomous driving task and obtain optimal driving strategies. Alternatively, under the feature-based behavior modeling [7], [8], [9], [10], [11], they designed different reward functions based on different driving styles by using the IRL algorithm. It would enhance the adaptability and accuracy of the prediction model in different scenarios. Additionally, Sadat et al. [12] have conducted joint perception and planning for self-driving vehicles to plan the human driving trajectory with the lowest cost among all possible trajectories by IRL. Meanwhile, some related work accomplishes the updating of reward functions and strategies by avoiding the computation of the optimal value function [13], minimizing the loss of risk-sensitive AGENT on behavioral observations [14], and defining new constraints from the failure demonstration behavior [15]. At the same time, they completed the decision-making task better under the state behavior space with continuous demonstration. And Xue et al. [16] fused inverse reinforcement learning algorithms with optimal control to propose a new model-based approach that is applied to the field of tracking control for learning unknown performance objective functions.

However, in inverse reinforcement learning, we need to utilize the learned reward function to drive reinforcement learning which accomplishes the exploration of the agent's optimal policy trajectory. In the process of exploration, the agent needs to continuously learn by trial and error with the environment. Therefore, it would consume greater computational resources. Also, in practical problems, inverse reinforcement learning causes low learning speed in the early stage of training due to the lack of expert sample data. Therefore, Chen et al. [17] has utilized GAN to generate more expert sample data, which initially solved the sparse expert sample problem of inverse reinforcement learning. Generative Adversarial Network (GAN) [18] makes use of the concept of zero-sum games to complete training through adversarial learning of generator and discriminator models. Recent related work reconfigures the GAN through codec structure as well as implements the combination of GAN and Maximum Entropy modeling strategies. They achieved learning features of discrete targets during adversarial training [19], and at the same time maximized the robustness of the adversarial strategy [20]. Ho and Ermon [21] have suggested Generative Adversarial Imitation Learning (GAIL) to obtain a model-free imitation learning algorithm between imitation learning and GAN. Fu et al. [22] has proposed Adversarial Inverse Reinforcement Learning (AIRL), which has been used in OpenAI Gym simulations of autonomous driving. Both GAIL and AIRL have shown superior performance as decision frameworks in the field of autonomous driving decision planning. Several recent works accomplish motor tasks in IRL by conditioning on reward adjustment [11], [23], [24], [25], driving style [26], and attention mechanisms [27], trained by a generative adversarial framework to learn optimal strategies.

TABLE 1. The comparison of the features of existing methods.

Appoach	Features
IRL	sparse sample of experts more computationally intensive
GAN-IRL	overcome sparse expert samples inefficient sample quality more computationally intensive
GAIL	suitable for autopilot scenarios prone to mode crashes
AIRL	suitable for autopilot scenarios difficult training, mode crash
AC-GAN-IRL (ours)	overcome sparse expert samples generated samples efficiently low computational cost suitable for autopilot scenarios

Albeit the successful fusion of GAN with IRL opens up new possibilities for autonomous driving decision-making methods, the randomness of the generator of GAN itself leads to the inefficiency of generating samples. This causes a significant increase in the training cost of IRL. At the same time, inefficient samples also cause the interaction process between the agent and the environment to become more and more complex, making it difficult for the agent to accomplish the decision-making task in complex scenarios. Therefore, considering the limitations of existing methods under real-world tasks, as demonstrated in Table 1, our goal in this paper is to solve the sample inefficiency problem caused by adding GANs during inverse reinforcement learning. We expect that our approach can be more efficient for decision-making tasks under continuous state space. We seek to leverage Deep Deterministic Policy Gradient (DDPG) [28] as an alternative to the stochastic policy gradient descent of the original GAN training process.

DDPG is a policy learning method that integrates deep learning neural networks in Deep Q-Network (DQN) [29] with Deterministic Policy Gradient (DPG) [30]. Zheng and Liu [31] have proposed an improved multi-agent deep deterministic policy gradient (IMADDPG) algorithm that enables the agent to maximize the collaborative planning performance during the training period. And Nguyen et al. [32] have presented two deterministic policy approaches, distributed deep deterministic policy gradient and shared deep deterministic policy gradient, addressing the multi-agent resource allocation problem.

In this work, we present the AC-GAN-IRL algorithm framework, a DDPG-based approach under GAN to address the sample inefficiency induced by the randomness of the generator. Our goal is to improve the IRL method that incorporates GAN models, enabling our model to solve the

problem of sparse expert samples at the beginning of IRL training. At the same time, our method can generate more efficient samples to optimize the IRL decision-making process and reduce the complexity of the agent's interaction with the environment during the process of exploring the state space. Our model reformulates the generator structure in GAN, using a DDPG deterministic strategy to replace the random noise input in the original GAN model. The new generator is defined based on the Actor-Critic mechanism, which makes the inverse reinforcement learning decision process with the GAN game mechanism more efficient. Our approach enhances the quality of GAN-generated samples and reduces the process of training with invalid samples. This significantly decreases the computational cost of inverse reinforcement learning when joining the network training. Also, the virtual expert samples generated by GAN are mixed with the initial expert samples as the expert sample set for IRL. The reward function, which in turn drives the RL process for policy updating and optimization, is obtained by comparing the differences between non-expert and expert trajectories. And the non-expert trajectory samples are regenerated using the new strategy. Finally, our approach evaluates the difference with the mixed expert sample set again to get the reward function and optimal policy. Meanwhile, in the field of autonomous driving decision planning, we have been exploring the research hotspot of how to realize vehicle safety decision-making through intelligent technology. Our method implements a reconstruction of the generator structure of GAN, which makes the model generate more efficient samples in adversarial training. We accelerate the learning and convergence speed of IRL at the initial stage and reduce the computational cost brought by adding GAN networked training. Thus, our method can better solve large-scale continuous state space problems in the IRL domain, making it applicable to model optimization and safety decisions in more practical scenarios involving autonomous driving and robotics. We summarize our contributions as follows:

1) DDPG-based Virtual Expert Sample Construction under GAN: We reformulate the algorithmic structure of GAN using DDPG. Our approach is based on the Actor-Critic mechanism utilizing a deterministic strategy to solve the problem of inefficient generation of samples caused by the randomness of the generator itself. Since our model uses DDPG to optimally decouple the generators in the original GAN, it improves the sampling efficiency of the entire training process. Meanwhile, our reformulation reduces the complexity of the agent-environment interaction by providing better sample data that approximates the expert trajectory for IRL. Therefore, our model can be applied to address the needs of scenarios in decision-making domains (e.g., autonomous driving or robotics).
2) Trajectory Prediction Based on AC-GAN-IRL Model under Grid World: Our reconstructed generator based on the DDPG algorithm completes the adversarial gaming process, generating more efficient samples. And in the early stage of inverse reinforcement learning, our model utilizes such mixed sample sets to recover a reward model in the scenario of autonomous driving. We obtain the optimal policy based on the recovered reward model, which infers the ultimate goals of vehicle trajectories in the grid world. Additionally, based on the action distribution of the relevant sampling policy, the future vehicle trajectory can be forecast.

II. PRELIMINARIES

This section provides background information and related works, including the relevant basic algorithms for Inverse Reinforcement Learning and Generative Adversarial Network. In particular, a brief description of DDPG.

Recently, several models related to inverse reinforcement learning have been proposed successively to optimize complex decision-making scenarios in artificial intelligence as well as robotics. Shi et al. [33] used the IRL framework to alternately optimize the reward and policy functions for diverse text generation tasks. Recent related work decouples complex tasks through conditionalized models [34], [35], [36]. They accomplish reward function optimization for motor tasks related to inverse reinforcement learning, as well as strategy updating, based on subtask representation learning framework [35], behavioral fusion [34], and decision preferences [36], respectively. Alternatively, by fusing IRL processes through adversarial models [37], [38], [39], they can model the motion task to complete the zero-sum game training process [37], [38], while learning a transferable multi-task reward function to complete sub-tasks in highly complex scenarios [39]. Thus, recent work has adapted to the task requirements in various types of scenarios by conditionalizing the IRL process. In this paper, we aim to decontextualize and optimize the DDPG and GAN algorithmic framework that solves the sample inefficiency problem occurring in the training process of inverse reinforcement learning and improves the computational efficiency of the decision movement process.

A. MDP

We consider a Markov Decision Process as a formal description or modeling of the environment in which an agent operates. A Markov Decision Process is defined as $< S, A, P, R, \gamma >$, where A is denoted as a finite set of actions. At each time, the agent chooses an action among the possible actions $a \in A$. And it moves to the next state $s' \in S$ based on a policy π

$$\pi\left(a|s\right) = P[A_t = a|S_t = s] \tag{1}$$

We hope that there always exists an optimal policy π^* for any MDP to obtain the corresponding optimal state-value function $V_\pi(s)$ and optimal action-value function $Q_\pi\left(s, a\right)$ [2], which corresponds to the Bellman Optimality Equation as

$$V_*\left(s\right) = max_{a\in A}\left(R_s^a + \gamma \sum_{s'\in S} P_{ss'}^a V_\pi\left(s'\right)\right), \tag{2}$$

$$Q_*(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a max_{a' \in A} Q_*\left(s' \mid a'\right). \quad (3)$$

B. IRL

In the autonomous driving scenario, we expect to use the concept of IRL to learn a reward function from the driver's decision behavior in order to help self-driving vehicles make logical planning decisions. At the same time, Inverse reinforcement learning considers the expert strategy as the optimal strategy. And the reward function is learned by randomly generating non-expert trajectories and comparing their differences with the expert trajectories. Then it drives the reinforcement learning process positively to optimize the behavioral strategy.

The demonstration trajectories $D = \{\tau_i\}_{i=1}^n$ for a set of experts are given, with each expert's demonstration trajectory being $\tau_i = \{(s_0, a_0), (s_1, a_1), \ldots, (s_k, a_k)$. We assume that the expert tries to optimize an unknown reward function. And the objective of IRL is to find an optimal policy in terms of performing as well as the expert. Assuming the optimal strategy is π^*, it follows that

$$V_{\pi^*}(s) \geq V_\pi(s), \quad Q_{\pi^*}(s,a) \geq Q_\pi(s,a). \quad (4)$$

At this point, according to Bellman Equation

$$V_\pi(s) = R + \gamma P_{s'|(s,\pi^*(s))} V_\pi(s) \quad (5)$$

to obtain the reward function R. The equation above is specified as [2]

$$P_{s'|(s,\pi^*(s))} V_\pi(s) \geq P_{s'|(s,\pi(s))} V_\pi(s). \quad (6)$$

C. GAN

Inverse reinforcement learning initially faces the problem of lacking expert sample data. Therefore, the introduction of GAN into IRL can initially solve the problem of sparse expert samples. Generative Adversarial Network (GAN), as a deep learning model in unsupervised learning, enables G (Generator) to generate fabricated samples by the mutual game of G and D (Discriminator). Initially, the generator needs to sample a batch of real samples in the real data set. At the simultaneous time, a random noise vector is generated from the Z distribution and input to the generator to generate faked samples, which are jointly input to the discriminator with the real samples for discriminating. The loss function of the training G [18] concerning the noise Z is defined as

$$L_{G=} E_{z \sim P_z(z)} \left[\log\left(1 - D\left(G\left(z\right)\right)\right)\right]. \quad (7)$$

D. DDPG

Equation (7) involves that GAN creates a random noise through the Z distribution during the original training process to generate virtual samples in the generator. Although adding GAN to IRL can generate more expert samples to overcome the problem of sparse expert samples, such a training process optimized by random policy gradients results in inefficient generated samples and greatly increases the training cost of the inverse reinforcement learning process. Therefore, our model uses the deterministic strategy to improve the quality of the virtual expert samples generated by GAN which minimizes the computational cost.

Deep Deterministic Policy Gradient (DDPG) gives the agent the ability to keep learning through trial and error until the action policy and the appropriate state are matched. DDPG utilizes a deterministic policy μ to select the action a_t, where $\mu\theta$ is used as the parameter to generate the deterministic action online μ policy network that defines as

$$a_t = \mu\left(S_t \mid \theta^\mu\right). \quad (8)$$

Convolutional neural networks are used to simulate the policy function μ and Q function (action-value). Additionally, two neural networks, online and target, are created from the policy network and Q network respectively. Meanwhile, the performance measure of strategy μ [30] is defined as

$$J_\beta(\mu) = \int \rho^\beta(s) Q^\mu(s, \mu(s))\, ds = E_{s \sim \rho^\beta}\left[Q^\mu(s, \mu(s))\right]. \quad (9)$$

We reconstructed the algorithmic structure of GAN using the Actor-Critic idea. Our approach combines DDPG with GAN to enhance the quality of GAN-generated samples, which better solves the problems arising from adding GAN to IRL. It also lessens the complexity of agent-environment interaction throughout the learning process.

III. PROPOSED APPROACH

The expert strategy is regarded as the best one by IRL, which recovers the reward function via the expert's demonstration trajectory and motivates RL to raise the agent's strategy level in turn. Also, IRL needs to include the training process where RL continuously interacts with the environment in trial and error, which will result in increased processing costs for complex environments. Although some researchers [17] have addressed the issue of sparse expert samples by incorporating GAN during the initial period of training, the inherent flaws of the GAN algorithm itself can result in inefficient sample generation and increase the high complexity of the agent's interaction with the environment. Therefore, we propose a generative adversarial inverse reinforcement learning algorithm based on DDPG in response to this issue.

We reconstruct the structure of the generator through the Actor-Critic framework of DDPG so that it completes the adversarial game with the discriminator. In the Actor model of the generator, we define a policy network with two fully connected layers and use the ReLu function for activation. Finally, we use the linear layer structure of the neural network to output the action. Similarly, the Critic model defines the value function network for estimating the value of a given observed state and behavioral action. It receives observed states and actions as inputs and performs feature extraction through two fully connected layers. Finally, we output the estimated values of the model. Also, we define the discriminator network structure to determine a given observed state and action whether it is an expert's demonstrated behavior or not. It can generate a reward signal based on the observed

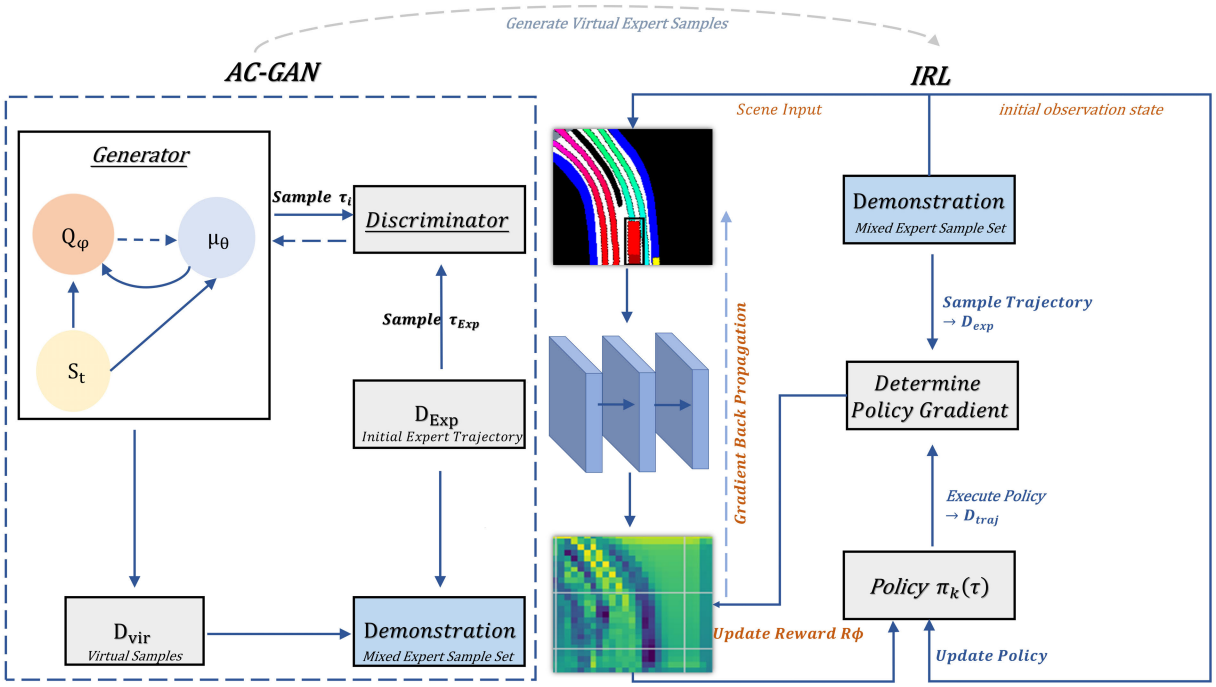

FIGURE 1. Overview: our AC-GAN-IRL model consists of three modules: (1) a generator based on an Actor-Critic framework, (2) a GAN model combined with DDPG to generate virtual expert samples, and (3) a MaxEnt IRL process for learning reward function and optimal policy.

states and actions of the environment, containing two fully connected layers. And we use the LeakyReLU activation function for iterative training with the generator model reconstructed in the DDPG framework.

Using the Actor-Critic framework based on an off-policy model-free feature, in Figure 1, our approach addresses the sample inefficiency caused by the stochasticity of the GAN generator. By recoupling the generator structure in the GAN, we utilize the DDPG deterministic strategy to complete the adversarial training process with the discriminator, which lessens the complexity of the agent learning process interacting with the environment. The virtual expert trajectory samples generated by GAN are mixed with the original expert samples, which would be used as the expert sample set for inverse reinforcement learning. Furthermore, we generate a series of non-expert trajectories based on a stochastically generated strategy $\pi_k(\tau)$ as the initial strategy. It compares with the mixed expert trajectory samples to complete the learning of the reward function $R\phi$. In turn, forward-driven reinforcement learning is performed to update the policy $\pi_{k+1}(\tau)$. Then it generates new non-expert trajectories with expert trajectories to update the reward function $R\phi$ iteratively. It is stopped when no policy produces a cumulative reward expectation that is higher than the one generated by the expert policy.

A. DDPG-BASED VIRTUAL EXPERT SAMPLE CONSTRUCTION UNDER GAN

To address the intrinsic defect of GAN that leads to inefficient sample generation, we suggest adding DDPG to the generator along with the discriminator for accomplishing the adversarial training and enhancing the quality of generated samples. Ultimately, our method generates more virtual samples that approximate the expert trajectory to participate in the inverse reinforcement learning training. With the help of the two models, the generator and the discriminator, we hope to create more virtual expert samples that resemble real samples. This process leads to a Nash equilibrium [18], [21], which can be characterized as the procedure of resolving a binary-valued function with a minimax game solution process that defines as

$$\begin{aligned} &min_G max_D V(D,G) \\ &= max_D min_G E_{x\sim P_{data}(x)}[logD(x)] \\ &\quad + E_{z\sim P_z(z)}\left[\log\left(1-D\left(G\left(z\right)\right)\right)\right]. \end{aligned} \tag{10}$$

We bring in the DDPG Actor-Critic architecture to decouple the generator into two interrelated modules: the policy module (μ_θ) and the Critic module (Q_φ). Within the Actor-Critic architecture, the agent uses the deterministic policy μ_θ to complete its interaction with the environment. We expect to obtain the optimal policy module μ_θ in the reconstructed generator, which produces an efficient set of trajectory samples τ_i. We store the state transfer process (s_t, a_t, r_t, s_{t+1}) in the replay buffer R. Sampling random transitions from R, as the mini-batch training data of strategy μ_θ and Critic Q_φ. And the target value y_i is given by

$$y_i = r_i + \gamma Q'\left(s_{i+1}, \mu'\left(s_{i+1}|\vartheta^{\mu'}\right)|\vartheta^{Q'}\right). \tag{11}$$

We update the Critic parameter Q_φ based on gradient descent to minimize the loss

$$L = \frac{1}{N}\sum_i \left(y_i - Q\left(s_i, a_i|\varphi\right)\right)^2. \tag{12}$$

And we also utilize policy gradient for the parameter update of the policy μ_θ

$$\nabla_\theta J \approx \frac{1}{N}\sum_i \nabla_a Q\left(s, a|\varphi\right)|_{s=s_i, a=\mu(s_i)} \nabla_\theta \mu(s|\theta)|_{s_i}, \tag{13}$$

and we also add learning rate τ using a soft update to renew the target network μ' and Q'

$$\vartheta^{\mu'} \leftarrow \tau\vartheta^{\mu} + (1-\tau)\,\vartheta^{\mu'}, \tag{14}$$

$$\vartheta^{Q'} \leftarrow \tau\vartheta^{Q} + (1-\tau)\,\vartheta^{Q'}. \tag{15}$$

Our model incorporates the Actor-Critic framework of DDPG into the GAN generator module to participate in the adversarial training process. The trajectory samples $\tau_i = \{(s_0, a_0), (s_1, a_1), \ldots, (s_k, a_k)$ sampled with the deterministic strategy μ_θ and the real trajectory samples τ_{Exp} of expert sampling D_{Exp} are used as the input of the discriminator for discriminating the authenticity of the trajectory samples. We aim to enable the generator model to produce sample data that increasingly approximate the expert's sampling trajectory, providing more efficient samples for the subsequent IRL decision-making process. We solve for the optimal value of the GAN's objective function $min_G max_D V(D, G)$. Then $V(D, G)$ is converted into a calculus form for the expression of the expectation

$$\begin{aligned} &V(D, G) \\ &= \int_{-\tau_{Exp}}^{\tau_{Exp}} P_{data}\left(\tau_{Exp}\right)\left[logD\left(\tau_{Exp}\right)\right] d\tau_{Exp} \\ &\quad + \int_{-\tau_i}^{\tau_i} P_g(\tau_i)\left[log\left(1 - D\left(\tau_i\right)\right)\right] d\tau_i. \end{aligned} \tag{16}$$

We fix the generator to perform a bias derivative on $V(D, G)$ concerning the discriminator to obtain the optimal solution.

$$D^*\left(\tau_{Exp}, \tau_i\right) = \frac{P_{data}(\tau_{Exp})}{P_{data}\left(\tau_{Exp}\right) + P_g(\tau_i)}. \tag{17}$$

When $P_{data}\left(\tau_{Exp}\right) = P_g(\tau_i)$, we expect the trajectory sample distribution τ_i generated by the deterministic strategy sampling to be consistent with the true sample distribution τ_{Exp}. The output of the discriminator is the probability that the trajectory samples sampled from the deterministic strategy μ_θ are expert, which is used to feedback to the generator to calculate the loss function of GAN. Then we take backpropagation to obtain the gradients of the corresponding nodes of the network parameters and perform the gradient update continuously. The parameters of the entire AC-GAN model are updated according to the respective gradients by selecting the corresponding optimizer, which will optimize the GAN model containing the DDPG algorithm. Our approach successfully raises the standard of the virtual samples produced by GAN.

We blend the virtual expert trajectory samples in Figure 2 generated by DDPG-based training under GAN with the initial expert examples, which serve as the expert sample set for the IRL process.

Also, the pseudo-code of Algorithm 1 (AC-Generator For Virtual Sample) provides a more thorough explanation of the training procedures.

B. PROBABILITY-BASED MODELING IRL

Two modeling processes, the maximal margin-based model and the probability-based model, are categorized into IRL. For any policy in the maximal margin-based model, there may be more than one reward function that makes the resulting trajectory optimal. Therefore, probabilistic model-based modeling of IRL from the maximum entropy principle eliminates the matching ambiguity of such a many-to-one relationship.

We seek a reward function that maximizes the entropy of the obtained strategy if there is a prospective probability distribution that generates an expert trajectory D. Also, the feature expectation of the trajectory generated by the strategy matching the feature expectation of the expert is the minimum deviation distribution of the trajectory. Without making any subjective assumptions about the unknown scenario, we constrain the model via the maximum entropy approach. It is possible to guarantee feature expectation matching with no more restriction to a specific trajectory. At this point, [40] a prerequisite is given by

$$\sum_{\tau_i} P(\tau_i) f = f_{Exp}. \tag{18}$$

Here, f is used to denote the feature expectation and f_{Exp} indicates the expert feature expectation. We model a maximum entropy inverse reinforcement learning model to solve the optimization problem, enabling the model decision process to eliminate matching ambiguity. We solve the optimization problem with maximum entropy, which can be tackled by the Lagrange multiplier method as follows

$$\begin{aligned} min\, L = &\sum_{\tau_i} P(\tau|\phi)\, logP(\tau|\phi) \\ &- \sum_{j=1}^{n} \lambda_j \left(P(\tau|\phi) f_j - f_{Exp}\right) \\ &- \lambda_0 \left(\sum_{\tau_i} P(\tau|\phi) - 1\right). \end{aligned} \tag{19}$$

Then the probability $P(\tau|\phi)$ is differentiated and made its derivative 0 to obtain the probability of the trajectory satisfying the maximum entropy

$$P(\tau\,|\,\phi) = \frac{exp\left(\sum_{j=1}^{n} \lambda_j f_j\right)}{exp\,(1-\lambda_0)}, \tag{20}$$

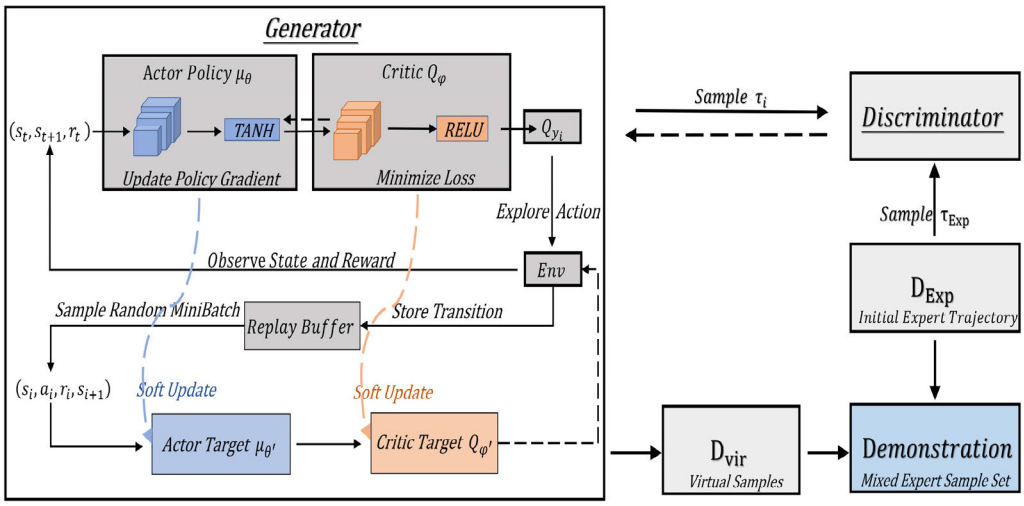

FIGURE 2. Schema for our generator based on an Actor-Critic framework.

Algorithm 1 AC-Generator For Virtual Sample

The critic network $Q(s_t, a \mid \varphi)$ and actor $\mu(s_t \mid \theta)$ are stochastically initialized
Set the network weights as φ and θ
Initialize target network parameters (θ', φ')
Initialize replay buffer R
for $m = 1, M$ **do**
  A random process N is initiated to explore the action
  Receive the initial observation state s_1
  for $t = 1, T$ **do**
    Execute action a_t and observe reward r_t, getting a new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Update critic Q_φ by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\varphi))^2$
    Update the actor policy μ_θ using the sampled policy gradient
$$\nabla_\theta J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\varphi)\mid_{s=s_i, a=\mu(s_i)} \nabla_\theta \mu(s|\theta)\mid_{s_i}$$
    Update target network parameters (θ', φ') to slowly track φ and θ respectively
  end for
  Collect trajectories $\tau_i = \{(s_0, a_0), (s_1, a_1), \ldots, (s_k, a_k)\}$ by executing the actor policy μ_θ
end for

where λ_j $(j = 1, \ldots, n)$ is the auxiliary variable. And Z is the normalization factor with ϕ being the parameter of the reward function.

We utilize the Maximum Likelihood method to find the reward function ϕ with parameters [2], [40]

$$max_\phi \frac{1}{N}\sum_{i=1}^{N} logP(\tau|\phi) = max_\phi \frac{1}{N}\sum_{i=1}^{N} r_\phi(\tau_i) - logZ, \tag{21}$$

in which $Z = \int P(\tau|\phi)\, exp\left(r_\phi(\tau)\right) d\tau$. We simplify it to obtain

$$\nabla_\phi L = \frac{1}{N}\sum_{i=1}^{N} \nabla_\phi r_\phi(\tau_i) - \frac{1}{Z}\int P(\tau|\phi)\, exp\left(r_\phi(\tau)\right)\nabla_\phi r_\phi(\tau_i)\, d\tau, \tag{22}$$

$$\nabla_\phi L \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\phi r_\phi(\tau_i) - \frac{1}{M}\sum_{j=1}^{M} \nabla_\phi r_\phi(\tau_j), \tag{23}$$

which represents the difference between the reward gradient summation of the expert sample trajectory and the reward gradient summation currently obtained.

During the process of updating the parameters of the reward function ϕ, on each occasion, we need to find the optimal policy under the current reward function to make a policy gradient which requires a significant amount of computational power. Therefore, when instructing RL to solve the policy, we employ lazy policy optimization that terminates the computation after a certain number of iterations.

To compensate for the generated loss, an important sample is used to add weights ω. Accordingly, rather than processing the distribution of the whole trajectory, we only need to find one optimal trajectory corresponding to the optimal policy in the gradient calculation. The gradient function for resolving the reward function parameter ϕ is

$$\begin{aligned}\nabla_\phi L \approx & \frac{1}{N}\sum_{i=1}^{N}\nabla_\phi r_\phi(\tau_i) \\ & -\frac{1}{\sum_j \omega_j}\sum_{j=1}^{M}\omega_j \nabla_\phi r_\phi(\tau_j),\ \omega_j \\ = & \frac{p(\tau)\, exp\left(r_\phi(\tau_j)\right)}{\pi(\tau_j)}. \end{aligned} \tag{24}$$

C. DDPG-BASED GENERATIVE ADVERSARIAL INVERSE REINFORCEMENT LEARNING

To address the problem of inefficient samples generated resulting from the addition of GAN to the IRL process, we propose a DDPG-based generative adversarial inverse reinforcement learning algorithm. Our method combines the Actor-Critic architecture in DDPG with the original generator in GAN and then completes adversarial training with the discriminator D_θ in GAN, aiming to solve the inefficiency of generated samples caused by the randomness of the original GAN generator and improve the quality of generated samples. Simultaneously, we execute the deterministic policy module μ_θ of the AC-GAN generator in Algorithm 1 to generate the virtual expert sample set D_{vir} in the iterative process. We mix the virtual expert sample set D_{vir} with the original expert sample D_{Exp},making the mixed sample set D to fulfill the decision-making task of the IRL process.

We aim to model IRL based on the maximum entropy mechanism. The optimal parameters of the reward function ϕ are solved based on comparing the differences between the non-expert trajectory samples generated by a stochastic policy and the mixed expert trajectory samples, as well as finding the optimal policy $\pi(\tau)$. Also, Algorithm 2 AC-GAN-IRL pseudo-code has a more detailed description of our approach.

IV. EXPERIMENTAL ANALYSIS

A. DIFFERENT ALGORITHM PERFORMANCE COMPARISON FOR BENCHMARK TASKS

Considering the fact that the actual model of autonomous driving is relatively huge and requires high computational efficiency of the network in a realistic environment, our model is based on the OpenAI Gym interface and uses the third-party physics engine MuJoCo to create interactive environments for validating the performance of our approach. In our experiments, based on our approach with the existing framework, three different types of models are built to enable the agent to accomplish the related motion tasks in various environments. Also, we evaluate the performance efficiency of the algorithms for comparison. The total times of interaction when reaching the value of expert demonstration reward is used as the evaluation metric$_1$, which indicates the basis of whether the performance efficiency of our model is superior to the existing methods. Meanwhile, we use the episode reward obtained by the model during each round of iterative training in different environments as the model performance evaluation metric$_2$, which indicates the basis of whether the model can perform as well as the experts. Three different simulation environment settings—InvertedPendulum-v2, Hopper-v2, and HalfCheetah-v2—are sequentially developed in accordance with the motion environment's growing complexity. In Figure 3, we explore how the algorithm performance of AC-GAN-IRL, GAN-IRL, and IRL changes in the process of agent-environment interaction. In different MuJoCo simulation environments, the three types of models are trained in an increasing number of iterations depending on the complexity of the simulation environment. We expect that within a limited number of iterations, our method can obtain plot reward values as good as those performed by the expert demonstration. At the same time, we can complete the decision-making process more efficiently, i.e., completing the iterative training faster than the other two existing methods (GAN-IRL and IRL). In each environment, different learning rates are set for training to complete a comparison of the effects of actor loss and critic loss in the AC-GAN-IRL model.

In order to compare the performance differences between our approach and the existing models, we provide the corresponding sample set of expert trajectories in each environment. Accordingly, we train the corresponding algorithm models in different environments with the same random seeds. As in Figure 3 (left) where the horizontal coordinates represent the number of iterations, and the vertical coordinates represent the obtained episode rewards. In each scenario, we perform the corresponding iterative training according to the complexity of the environment and display the experimental data as scatter plots or line plots. The different colored curves represent each episode reward for the same random seeds, in which the blue curve represents our AC-GAN-IRL model with DDPG, the red curve represents the IRL model combined with GAN only, as well as the green curve represents the IRL model just. In each model, we keep the training logs saved at every 200 steps and the number of iterations for each round of training is set to 20 uniformly. At the same time, the discount factor γ is 0.9 and the number of expert trajectories we provide to each simulation environment is set to 32. In the three types of simulation environments, we initially set our model to achieve the desired value by completing 10,000 iterations of training. However, we find that the experimental models in different environments with such hyperparameter settings have difficulty in reaching the reward value of the expert demonstration during the iteration process. Therefore, we consider that in the comparison experiments between our method and the existing methods, the number of iterations for model training needs to be differentiated according to the complexity of the three

Algorithm 2 AC-GAN-IRL

Obtain initial expert trajectories D_{Exp}
Initialize Discriminator D_θ
for $n\epsilon\{1, \ldots, n_{max}\}$ **do**
  for $t = 1, T$ **do**
    Collect trajectories $\tau_i = \{(s_0, a_0), (s_1, a_1), \ldots, (s_k, a_k)\}$ by executing actor policy μ_θ
    Sample minibatch of k examples $\tau_{Exp} = \{(s_0, a_0), (s_1, a_1), \ldots, (s_k, a_k)\}$ from D_{Exp}
    Train D_θ to classify expert data τ_{Exp} from samples τ_i
    Update the Discriminator D_θ by ascending its stochastic gradient

$$L_{D=}E_{\tau_{Exp}\sim P_{data}(\tau_{Exp})}\left[logD\left(\tau_{Exp}\right)\right] + E_{\tau_i\sim P_g(\tau_i)}\left[\log\left(1 - D\left(\tau_i\right)\right)\right]$$

  end for
  Update the Generator by executing Algorithm 1
end for
Initialize $\pi_k(\tau)$ as a stochastic initial policy
Collect virtual samples D_{vir} by executing Algorithm 1
Append initial expert trajectories D_{Exp} and D_{vir} as a mixed expert sample set D
for $m = 1, M$ **do**
  Generate samples D_{traj} by executing $\pi_k(\tau)$
  Collect samples D_{exp} from D
  Append samples: $D_{samp} \leftarrow D_{exp} \cup D_{traj}$
  Use D_{samp} to update reward function $R\phi$ with a policy gradient

$$\nabla_\phi L \approx \frac{1}{N}\sum_{i=1}^{N}\nabla_\phi r_\phi(\tau_i) - \frac{1}{\sum_j \omega_j}\sum_{j=1}^{M}\omega_j \nabla_\phi r_\phi\left(\tau_j\right), \omega_j = \frac{p(\tau)\, exp\left(r_\phi\left(\tau_j\right)\right)}{\pi\left(\tau_j\right)}$$

  Update $\pi_k(\tau)$ using D_{traj} and the reward function $R\phi$ to obtain $\pi_{k+1}(\tau)$
end for
return optimized rewards function parameters ϕ and trajectory distribution $\pi_k(\tau)$

different MuJoCo simulation environments. We perform iterative trials and tuning of the experimental model in different environments with the aim of obtaining an efficient iterative training interval in which our method always reaches the expert demonstration reward faster than other existing methods. In the InvertedPendulum-v2 simulation environment, we assign the reward value of the expert demonstration to approximately 1000 and made the three different algorithm models trained for nearly 10,000 iterations, as shown in Table 2. Meanwhile, in the Hopper-v2 simulation environment, the reward value of our expert demonstration is set to about 600 and we make three different types of models to perform nearly 20,000 iterations of training in Table 3. Finally, the reward value for the expert demonstration is set to around 850 in the HalfCheetah-v2 simulation environment. We made the three different models training for nearly 30,000 iterations with the results shown in Table 4. Meanwhile, in Table 6, we have summarized the experimental data in Table 2-4 as well as in Figure 3 statistically. We expect to illustrate the implementation of the evaluation metrics for the three types of models to achieve the expert demonstration rewards in different environments, in Table 6.

And we train each model to learn a given expert trajectory demonstration in different environments, in order to obtain rewards almost equal to that of the expert trajectory. Additionally, we modified our AC-GAN-IRL model with various learning rates in Figure 3 (middle and right), exploring the variations of actor loss and critic loss across the generator AC framework. In the beginning, our model set the learning rate of the hyperparameter lr_1 to 0.001 in three different simulation environments, intending to set a larger value to achieve fast convergence. However, whether in the simple InvertedPendulum-v2 with Hopper-v2 environment or the difficult HalfCheetah-v2 simulation environment, we can find that the convergence effect of our Actor-Critic loss curves appears to be unstable with large oscillations. And in the subsequent adjustment process, we found that our model needed more time to complete trial-and-error learning when lr_3 was set to a smaller value of 0.0001, resulting in an overall slow convergence rate. Therefore, through constant trial and error of the model hyperparameters, we compare the convergence and stability performance of our method in different experimental settings according to the consequent changes in different learning rates in Table 5. If our model is able to achieve both fast convergence as well as stability under the coordination of a certain learning rate hyperparameter, we may try to use it as the optimal learning rate parameter value.

It has been discovered that 0.0003 as the learning rate lr_2 is a superior choice for rapid convergence of the model making

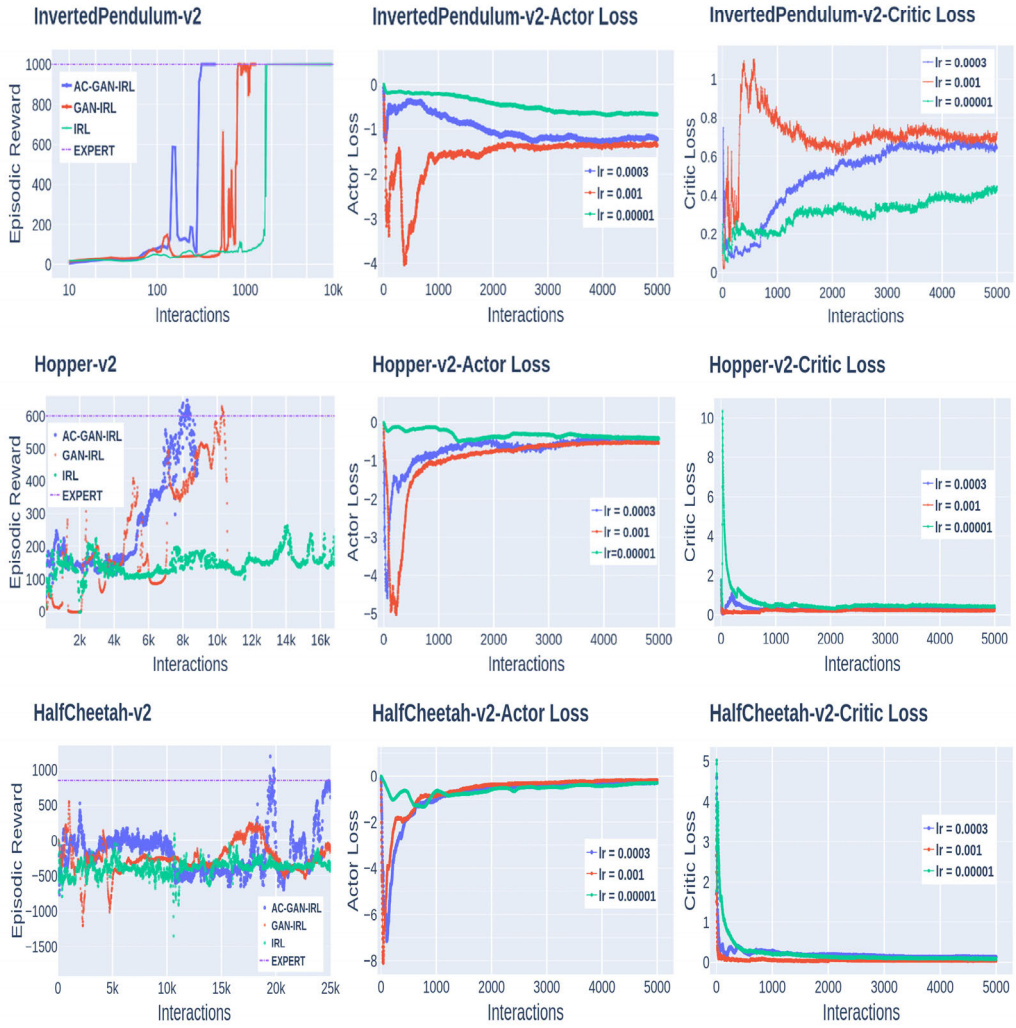

FIGURE 3. Different algorithm performance for benchmark tasks: We set the same seeds to randomize different algorithm models in various environments. In tens of thousands of iterations, we can show that our model (left) can complete the process of interacting with the environment more quickly and a certain amount of calculation cost is reduced in this work. Additionally, our model investigates the AC losses at various learning rates (center and right), illustrating the effects on model convergence.

the loss to be minimized. With the reconciliation effect of the learning rate on the model, the optimal strategy is fitted faster, making the learning process more stable. In Figure 3 (middle and right), we use the blue curve to depict the fluctuation of the Actor-Critic's loss curve when using lr_2 as the optimal parameter value for the learning rate. Also, at the best learning rate lr_2, in Table 5, we show the loss results of the training process of our method.

The comparison of experiments in different environments shows that our method addresses the sample inefficiency induced by the stochasticity of the generator. We reconfigure the generator model of GAN by utilizing the Actor-Critic framework in DDPG to replace the original generator model. During the adversarial training with the discriminator, our approach improves the quality of the samples generated by GAN to provide more high-quality samples that approximate the expert trajectory for IRL, making the process of reward function update and policy optimization more efficient. Therefore, our method AC-GAN-IRL enables the agent to complete the process of interacting with complex environments faster in all three different types of simulation environments. Compared with the other two existing methods (GAN-IRL and IRL), the iteration times of our model are much faster, which significantly lowers the computing cost of joining the network training. As shown in Table 6, our approach reduces the computational cost by approximately 20% for each type of given environment, using the iteration times as the execution of $metric_1$. Meanwhile, in the three different types of experimental environments, our method can reach the expert demonstration reward in a shorter time, which indicates that our method is efficient enough to eventually generate trajectory sample data as good as the expert performance. The episode rewards obtained by our method are constantly approaching the expert demonstration reward,

TABLE 2. Model performance comparison in InvertedPendulum-v2 environment.

Model	Evaluation Metrics	
	Interaction Time	Episode Reward
EXPERT	-	$(10.01 \pm 0.02) \times 10^2$
AC-GAN-IRL(ours)	$\mathbf{(4.88 \pm 1.52) \times 10^2}$	$\mathbf{(10.02 \pm 0.08) \times 10^2}$
GAN-IRL	$(10.69 \pm 2.10) \times 10^2$	$(9.69 \pm 1.08) \times 10^2$
IRL	$(23.64 \pm 3.80) \times 10^2$	$(9.23 \pm 0.58) \times 10^2$

TABLE 3. The comparison of model performance in Hopper-v2 environment.

Model	Evaluation Metrics	
	Interaction Time	Episode Reward
EXPERT	-	$(6.01 \pm 0.18) \times 10^2$
AC-GAN-IRL(ours)	$\mathbf{(8.071 \pm 0.348) \times 10^3}$	$\mathbf{(6.27 \pm 0.26) \times 10^2}$
GAN-IRL	$(10.25 \pm 0.290) \times 10^3$	$(5.78 \pm 0.58) \times 10^2$
IRL	$(16.91 \pm 1.328) \times 10^3$	$(2.69 \pm 0.34) \times 10^2$

TABLE 4. Results of comparing model performance in HalfCheetah-v2 environment.

Model	Evaluation Metrics	
	Interaction Time	Episode Reward
EXPERT	-	$(8.50 \pm 0.12) \times 10^2$
AC-GAN-IRL(ours)	$\mathbf{(19.83 \pm 0.330) \times 10^3}$	$\mathbf{(8.51 \pm 1.79) \times 10^2}$
GAN-IRL	$(24.93 \pm 0.241) \times 10^3$	$(2.38 \pm 2.34) \times 10^2$
IRL	$(25.87 \pm 1.416) \times 10^3$	$(0.913 \pm 0.42) \times 10^2$

shown in Table 6. However, the other two types of existing models sometimes fail to achieve expert performance after tens of thousands of training sessions.

Meanwhile, considering the safety factor in vehicle driving while reducing the cost overhead is very important for the decision domain of autonomous driving, which makes the idea that our model reduces the interaction process with the environment more desirable. Thus, in future practical engineering, our method can be applied to the field of autonomous driving or even robotics and other potential development directions in the field of artificial intelligence. Whether it is to learn the implicit driving behavior laws from human driver's trajectory data or to build more accurate decision-making models to improve the safety of the models by combining with environment sensing, our method can better act on practical problems in the IRL field.

B. TRAJECTORY PREDICTIONS IN GRID WORLD

Every grid in the grid world corresponds to a state in the environment, where the agent can perform four types of Action operations: up, down, left, and right. Meanwhile, the agent's action in each state is reflected as moving one frame on the grid. In the case that the agent takes an action that will take it off the grid, the agent's state remains unchanged with a penalty of -1. Also, the agent receives the corresponding reward (0 or 1) at the current State in the process of moving to the end of the grid world, in which we also set the discount factor γ to represent the effect on the current node value.

To explore the process of recovering the reward function for our AC-GAN-IRL model, we set up a basic experiment based on the grid world. In Figure 4 (left), we set up the true reward map in the grid world, as well as the initial value function map (middle-right). In the early stage of the experiment, we selected a set of expert trajectories as the initial sample set. Our method utilizes GAN incorporating the DDPG framework to generate virtual expert samples that provide more expert trajectory data in IRL. In a 30×30 grid world, the initial number of expert trajectories is set to 10, with the length of the expert trajectory samples set to 50 cells. Simultaneously, the discount factor γ is 0.9 and the learning rate is set to 0.01. We visualize the recovery effect of the reward function in the grid world of Figure 4 (middle-left) via the training of 1000 iterations of agent-environment interaction. Our model completes the iterative process of the value

TABLE 5. Model convergence or stability with learning rate adjustment and loss results.

Env	lr_1	lr_2	lr_3	Actor Loss	Critic Loss
InvertedPendulum-v2	√	√ √	√	-1.352	0.698
Hopper-v2		√ √	√	-0.539	0.229
HalfCheetah-v2	√	√ √		-0.167	0.038

TABLE 6. Results of comparing model performance in different environments.

Env	Metrics	**AC-GAN-IRL(ours)**	GAN-IRL	IRL	**Expert Demonstration**
InvertedPendulum-v2	Interaction Time /($\times 10^2$)	$\mathbf{4.88 \pm 1.52}$	10.69 ± 2.10	23.64 ± 3.80	-
	Episode Reward /($\times 10^2$)	$\mathbf{10.02 \pm 0.08}$	9.69 ± 1.08	9.23 ± 0.58	$\mathbf{10.01 \pm 0.02}$
Hopper-v2	Interaction Time /($\times 10^3$)	$\mathbf{8.071 \pm 0.348}$	10.25 ± 0.290	16.91 ± 1.328	-
	Episode Reward /($\times 10^2$)	$\mathbf{6.27 \pm 0.26}$	5.78 ± 0.58	2.69 ± 0.34	$\mathbf{6.01 \pm 0.18}$
HalfCheetah-v2	Interaction Time /($\times 10^3$)	$\mathbf{19.83 \pm 0.330}$	24.93 ± 0.241	25.87 ± 1.416	-
	Episode Reward /($\times 10^2$)	$\mathbf{8.51 \pm 1.79}$	2.38 ± 2.34	0.913 ± 0.42	$\mathbf{8.50 \pm 0.12}$

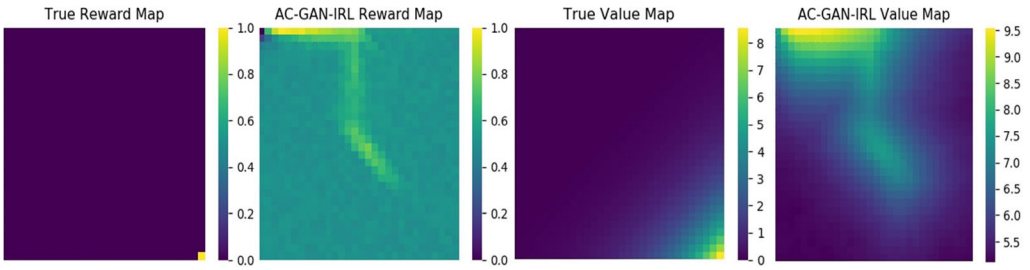

FIGURE 4. Recovery of reward functions based on the grid world: We obtain the true reward map (left) and value function map (middle-right) in the grid world. Our model produces more high-quality samples, which makes it better for IRL to recover the reward function (middle-left). According to the AC-GAN-IRL reward map recovered by our model, we sum the target state reward of any action. At the same time, through the max operation on the target state reward values to update the value function, we conduct the value function iteration process to accomplish the update of the AC-GAN-IRL value map(right).

function (right) by selecting the appropriate set of actions based on the learned reward update strategy.

Also, to explore the application of our approach in the field of autonomous driving decision-making, we expect to recover the reward function of the vehicle trajectory. It enables to obtain the optimal policy in the grid world where the final target location of the vehicle trajectory can be inferred. Meanwhile, we perform policy sampling based on the corresponding action distribution to generate reasonable vehicle prediction trajectories. We test our model on the publicly available autonomous driving dataset nuScenes [41].

With our model AC-GAN-IRL, we can generate more virtual expert trajectory samples based on the historical trajectory data of the vehicles, which provides more expert data for the subsequent learning of the reward function. In addition, for the trajectory prediction experiments in the grid world, we created an extractor for scene features based on convolutional neural networks. We use the feature extractor

TABLE 7. Results of our model in comparison with existing approaches.

Model	Benchmark Test			Evaluation Metrics	
	Grid World	Decision-making Scenarios	Performance Comparison		
LSTM-IRL[42]	-	√	-	Time Step	Lane Change Probability
BNIRL[13]	√	-	√	States Number	Policy Loss
Augmented-AIRL[43]	-	√	-	Time Step	Total Reward
Risk-Sensitive-IRL[14]	√	-	-	Time Step	Price Multiplier
AC-GAN-IRL(ours)	√	√	√	**Interaction Time**	**Episode Reward**

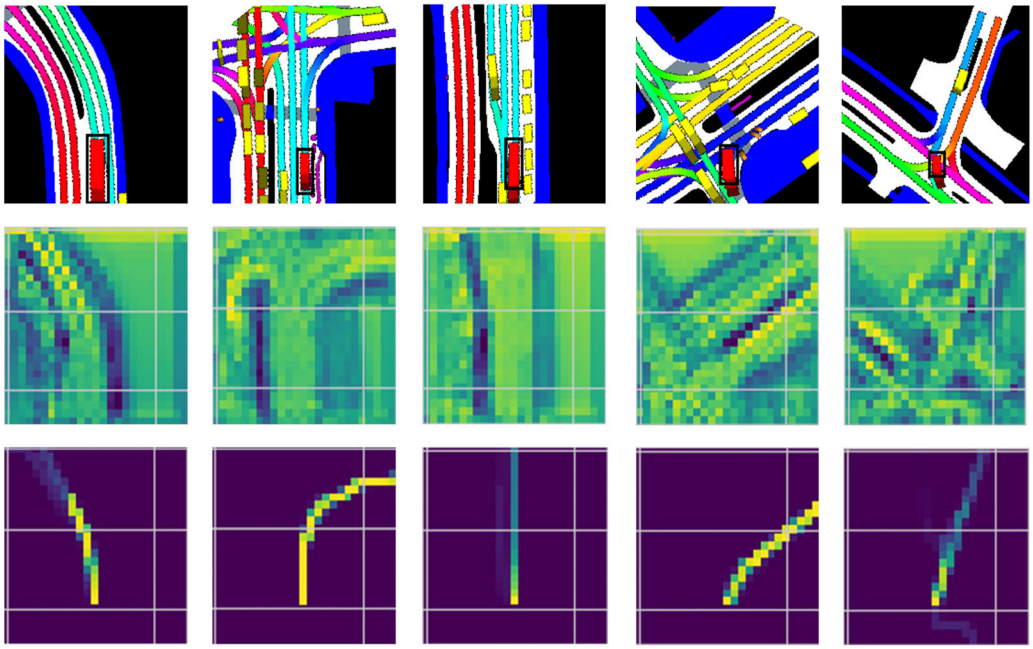

FIGURE 5. Trajectory predictions in the grid world. From top to bottom: Scene input, reward, vehicle trajectory prediction. With the historical trajectories of the vehicles, our model generates more samples that approximate the expert trajectories. It also contributes more expert data for recovering the reward function. We have tested our approach on a public dataset of autonomous driving to achieve superior prediction of vehicle trajectories (bottom) while visualizing the results of our experiments based on a 25 × 25 grid world.

to extract feature data from historical vehicle trajectories and current scene information in nuScenes data. With iterative training of the neural network, we input historical trajectory data (including virtual samples generated by our model) into the reward model along with current environmental feature information (including vehicle and environmental states). At the beginning of training, our model AC-GAN-IRL set the value of num of epochs to 200. We expect our method to recover our reward function better in the grid world after completing 200 epochs of training. However, in the process of recovering the reward function, it was found that the loss of the model did not decrease accordingly after 100 rounds of training. Therefore, considering the cost of time, we find that our method does not need to go through the training iteration process beyond 100 epochs to quickly solve the optimal reward function. With the learning rate of the model set to 0.001, we set the num of epochs to 100. After completing 100 epochs of training, we recover the reward function of

the vehicle trajectory and save the loss log of the model at every 100 steps. In order to determine the terminal target and sampling trajectory in the grid world, we continuously update the policy for the final optimal action distribution based on the learned rewards. The results of vehicle trajectory prediction based on the grid world are shown in Figure 5 (bottom). We recovered the reward function (middle) and implemented a reasonable prediction of the future trajectory of the vehicle depending on the optimal policy for sampling in the grid world. In addition, we provide a more detailed analysis of related work in Table 7, including the experimental design of the existing work with our model under the basic task, and their evaluation metrics defined for the model experiments. It can be found that we have better designed several experimental tasks to demonstrate the performance advantages of our approach. At the same time, according to the two evaluation metrics of Interaction Time and Episode Reward, we realized the comparison between our method and the existing models, as shown in Table 6.

V. CONCLUSION

We propose an AC-GAN-IRL method based on the DDPG deterministic policy ideology. Our approach reconstructs the generator in GAN through the Actor-Critic framework to overcome the inefficiency of generating samples due to the inherent stochasticity defect of GAN. In the process of completing adversarial training with the discriminator, our method generates samples of virtual expert trajectories that more closely approximate the rewards of expert demonstrations. This will provide more efficient sample data for the IRL decision-making process occurring under the GAN, dramatically improving the sample sampling efficiency of the IRL decision-making process. That process provides more expert samples for the initial phase of inverse reinforcement learning. At the same time, we complete the recovery of the reward function for policy updating and optimization. The experimental results demonstrate that our method can accelerate the convergence speed of the training process and reduce the complexity of agent-environment interaction. Remarkably, the computational cost of joining the GAN training is greatly diminished by about 20%. Also in the grid world, we show that our method recovers the reward function quite well and achieves a reasonable prediction of vehicle trajectories under the publicly available autonomous driving datasets. In future research work, we will continue to explore the continuous state space issues, making our model algorithm applicable to more practical scenarios. At the same time, we expect to optimize our method even further through more possibilities, so that our model can accomplish the decision-making task more accurately under different working conditions and highly complex traffic scenarios in the field of autonomous driving decision planning.

REFERENCES

[1] M. Wiering and M. Van Otterlo, "Reinforcement learning: State-of-the-art," in *Adaptation, Learning, and Optimization*, vol. 12. Berlin, Germany: Springer, 2012, doi: 10.1007/978-3-642-27645-3.

[2] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," in *Proc. 17th Int. Conf. Mach. Learn.* San Francisco, CA, USA: Morgan Kaufmann, Jun. 2000, pp. 663–670.

[3] M. Wulfmeier, D. Rao, D. Z. Wang, P. Ondruska, and I. Posner, "Large-scale cost function learning for path planning using deep inverse reinforcement learning," *Int. J. Robot. Res.*, vol. 36, no. 10, pp. 1073–1087, Sep. 2017, doi: 10.1177/0278364917722396.

[4] C. You, J. Lu, D. Filev, and P. Tsiotras, "Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning," *Robot. Auto. Syst.*, vol. 114, pp. 1–18, Apr. 2019, doi: 10.1016/j.robot.2019.01.003.

[5] K. Lee, D. Isele, E. A. Theodorou, and S. Bae, "Spatiotemporal costmap inference for MPC via deep inverse reinforcement learning," 2022, *arXiv:2201.06539*.

[6] Z. Wu, L. Sun, W. Zhan, C. Yang, and M. Tomizuka, "Efficient sampling-based maximum entropy inverse reinforcement learning with application to autonomous driving," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 5355–5362, Oct. 2020, doi: 10.1109/LRA.2020.3005126.

[7] M. Kuderer, S. Gulati, and W. Burgard, "Learning driving styles for autonomous vehicles from demonstration," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Seattle, WA, USA, May 2015, pp. 2641–2646, doi: 10.1109/ICRA.2015.7139555.

[8] Z. Huang, J. Wu, and C. Lv, "Driving behavior modeling using naturalistic human driving data with inverse reinforcement learning," 2020, *arXiv:2010.03118*.

[9] T. Fernando, S. Denman, S. Sridharan, and C. Fookes, "Deep inverse reinforcement learning for behavior prediction in autonomous driving: Accurate forecasts of vehicle motion," *IEEE Signal Process. Mag.*, vol. 38, no. 1, pp. 87–96, Jan. 2021, doi: 10.1109/MSP.2020.2988287.

[10] Y. Sun, Y. Chu, T. Xu, J. Li, and X. Ji, "Inverse reinforcement learning based: Segmented lane-change trajectory planning with consideration of interactive driving intention," *IEEE Trans. Veh. Technol.*, vol. 71, no. 11, pp. 11395–11407, Nov. 2022, doi: 10.1109/TVT.2022.3193220.

[11] Y. Zhou, R. Fu, and C. Wang, "Learning the car-following behavior of drivers using maximum entropy deep inverse reinforcement learning," *J. Adv. Transp.*, vol. 2020, Nov. 2020, Art. no. 4752651, doi: 10.1155/2020/4752651.

[12] A. Sadat, S. Casas, M. Ren, X. Wu, P. Dhawan, and R. Urtasun, "Perceive, predict, and plan: Safe motion planning through interpretable semantic representations," 2020, *arXiv:2008.05930*.

[13] B. Michini, M. Cutler, and J. P. How, "Scalable reward learning from demonstration," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 303–308, doi: 10.1109/ICRA.2013.6630592.

[14] L. J. Ratliff and E. Mazumdar, "Inverse risk-sensitive reinforcement learning," 2017, *arXiv:1703.09842*.

[15] K. Shiarlis, J. Messias, and S. Whiteson, "Inverse reinforcement learning from failure," in *Proc. Adapt. Agents Multi-Agents Syst. Int. Found. Auton. Agents Multiagent Syst.*, May 2016, pp. 1060–1068.

[16] W. Xue, P. Kolaric, J. Fan, B. Lian, T. Chai, and F. L. Lewis, "Inverse reinforcement learning in tracking control based on inverse optimal control," *IEEE Trans. Cybern.*, vol. 52, no. 10, pp. 10570–10581, Oct. 2022, doi: 10.1109/TCYB.2021.3062856.

[17] C. Jianping, C. Qiqiang, F. Qiming, G. Zhen, W. Hongjie, and L. You, "Maximum entropy inverse reinforcement learning based on generative adversarial networks," *Comput. Eng. Appl.*, vol. 55, no. 22, pp. 119–126, 2019.

[18] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, Oct. 2020, doi: 10.1145/3422622.

[19] X. Ye, Y. Bai, R. Song, K. Xu, and J. An, "An inhomogeneous background imaging method based on generative adversarial network," *IEEE Trans. Microw. Theory Techn.*, vol. 68, no. 11, pp. 4684–4693, Nov. 2020, doi: 10.1109/TMTT.2020.3015495.

[20] M. Shen and J. P. How, "Active perception in adversarial scenarios using maximum entropy deep reinforcement learning," 2019, *arXiv:1902.05644*.

[21] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proc. Adv. Neural Inf. Process. Syst.* Red Hook, NY, USA: Curran Associates, 2016, pp. 1–12.

[22] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adversarial inverse reinforcement learning," 2017, *arXiv:1710.11248*.

[23] Y.-F. Zhang, F.-M. Luo, and Y. Yu, "Improve generated adversarial imitation learning with reward variance regularization," *Mach. Learn.*, vol. 111, no. 3, pp. 977–995, Mar. 2022, doi: 10.1007/s10994-021-06083-7.

[24] A. Kinose and T. Taniguchi, "Integration of imitation learning using GAIL and reinforcement learning using task-achievement rewards via probabilistic graphical model," *Adv. Robot.*, vol. 34, no. 16, pp. 1055–1067, Aug. 2020, doi: 10.1080/01691864.2020.1778521.
[25] S. Choi, J. Kim, and H. Yeo, "TrajGAIL: Generating urban vehicle trajectories using generative adversarial imitation learning," *Transp. Res. C, Emerg. Technol.*, vol. 128, Jul. 2021, Art. no. 103091, doi: 10.1016/j.trc.2021.103091.
[26] J. Liu, L. N. Boyle, and A. G. Banerjee, "An inverse reinforcement learning approach for customizing automated lane change systems," *IEEE Trans. Veh. Technol.*, vol. 71, no. 9, pp. 9261–9271, Sep. 2022, doi: 10.1109/TVT.2022.3179332.
[27] J. Sun, L. Yu, P. Dong, B. Lu, and B. Zhou, "Adversarial inverse reinforcement learning with self-attention dynamics model," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 1880–1886, Apr. 2021, doi: 10.1109/LRA.2021.3061397.
[28] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.
[29] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.
[30] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 387–395.
[31] S. Zheng and H. Liu, "Improved multi-agent deep deterministic policy gradient for path planning-based crowd simulation," *IEEE Access*, vol. 7, pp. 147755–147770, 2019, doi: 10.1109/ACCESS.2019.2946659.
[32] K. K. Nguyen, T. Q. Duong, N. A. Vien, N.-A. Le-Khac, and L. D. Nguyen, "Distributed deep deterministic policy gradient for power allocation control in D2D-based V2V communications," *IEEE Access*, vol. 7, pp. 164533–164543, 2019, doi: 10.1109/ACCESS.2019.2952411.
[33] Z. Shi, X. Chen, X. Qiu, and X. Huang, "Toward diverse text generation with inverse reinforcement learning," 2018, *arXiv:1804.11258*.
[34] H. Shi, J. Li, S. Chen, and K.-S. Hwang, "A behavior fusion method based on inverse reinforcement learning," *Inf. Sci.*, vol. 609, pp. 429–444, Sep. 2022, doi: 10.1016/j.ins.2022.07.100.
[35] S.-W. Yoo and S.-W. Seo, "Graph-based subtask representation learning via imitation learning," in *Proc. Int. Conf. Electron., Inf., Commun. (ICEIC)*, Feb. 2022, pp. 1–4, doi: 10.1109/ICEIC54506.2022.9748273.
[36] X. Zhang, Y. Li, X. Zhou, and J. Luo, "CGAIL: Conditional generative adversarial imitation learning—An application in taxi drivers' strategy learning," *IEEE Trans. Big Data*, vol. 8, no. 5, pp. 1288–1300, Oct. 2022, doi: 10.1109/TBDATA.2020.3039810.
[37] D.-T. Pham, T.-N. Tran, S. Alam, and V. N. Duong, "A generative adversarial imitation learning approach for realistic aircraft taxi-speed modeling," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 3, pp. 2509–2522, Mar. 2022, doi: 10.1109/TITS.2021.3119073.
[38] M. Yan and J. Yang, "Interference coordination for autonomous HetNets based on adversarial learning," in *Proc. 13th Int. Conf. Commun. Softw. Netw. (ICCSN)*, Chongqing, China, Jun. 2021, pp. 393–398, doi: 10.1109/ICCSN52437.2021.9463652.
[39] S.-W. Yoo and S.-W. Seo, "Learning multi-task transferable rewards via variational inverse reinforcement learning," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 434–440, doi: 10.1109/ICRA46639.2022.9811697.
[40] Z. Zhou, M. Bloem, and N. Bambos, "Infinite time horizon maximum causal entropy inverse reinforcement learning," *IEEE Trans. Autom. Control*, vol. 63, no. 9, pp. 2787–2802, Sep. 2018, doi: 10.1109/TAC.2017.2775960.
[41] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "NuScenes: A multimodal dataset for autonomous driving," 2019, *arXiv:1903.11027*.
[42] X. Liao, Z. Wang, X. Zhao, Z. Zhao, K. Han, P. Tiwari, M. J. Barth, and G. Wu, "Online prediction of lane change with a hierarchical learning-based approach," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 948–954, doi: 10.1109/ICRA46639.2022.9812269.
[43] P. Wang, D. Liu, J. Chen, H. Li, and C.-Y. Chan, "Decision making for autonomous driving via augmented adversarial inverse reinforcement learning," 2019, *arXiv:1911.08044*.

MING ZHAN was born in Jingdezhen, Jiangxi, China, in 1999. She received the B.S. degree in software engineering from the Zhuhai College of Jilin University, Zhuhai, China, in 2021. She is currently pursuing the M.S. degree in digital information with the North China University of Technology, Beijing, China. Her current research interests include machine learning, inverse reinforcement learning, and decision planning for autonomous driving.

JINGJING FAN received the Ph.D. degree in mechanical engineering from Tsinghua University, Beijing, China, in 2011. He is currently a Senior Engineer with the School of Electrical and Control Engineering, North China University of Technology. He has published more than 20 journal articles and applied for more than 30 patents. His current research interests include intelligent networked vehicle control technology and complete vehicle control technology for hybrid electric drive vehicles.

JIANYING GUO received the degree from the Tianjin University of Technology and Education, in 2005. He is currently a Professor with the School of Automotive, Tianjin Vocational Institute, Tianjin, China. He is a famous Teacher of leading talents teaching at the Ten Thousand People Plan. His current research interest includes intelligent networked vehicle technology.