

## **GACS: Generative Adversarial Imitation Learning Based on Control Sharing**

**Huaiwei SI**

*School of Computer Science and Technology, Dalian University of Technology, Dalian 116024, China*  
*E-mail: sihuaiwei@mail.dlut.edu.cn*

**Guozhen TAN**

*School of Computer Science and Technology, Dalian University of Technology, Dalian 116024, China*  
*E-mail: gztan@dlut.edu.cn*

**Dongyu LI**

*School of Computer Science and Technology, Dalian University of Technology, Dalian 116024, China*  
*E-mail: lifongyu@mail.dlut.edu.cn*

**Yanfei PENG**

*School of Computer Science and Technology, Dalian University of Technology, Dalian 116024, China*  
*E-mail: pengyanfei@mail.dlut.edu.cn*

**Abstract** Generative adversarial imitation learning (GAIL) directly imitates the behavior of experts from human demonstration instead of designing explicit reward signals like reinforcement learning. Meanwhile, GAIL overcomes the defects of traditional imitation learning by using a generative adversary network framework and shows excellent performance in many fields. However, GAIL directly acts on immediate rewards, a feature that is reflected in the value function after a period of accumulation. Thus, when faced with complex practical problems, the learning efficiency of GAIL is often extremely low and the policy may be slow to learn. One way to solve this problem is to directly guide the action (policy) in the agents' learning process, such as the control sharing (CS) method. This paper combines reinforcement learning and imitation learning and proposes a novel GAIL framework called generative adversarial imitation learning based on control sharing policy (GACS). GACS learns model constraints from expert samples and uses adversarial networks to guide learning directly. The actions are produced by adversarial networks and are used to optimize the policy and effectively improve learning efficiency. Experiments in the autonomous driving environment and the real-time strategy game breakout show that GACS has better generalization capabilities, more efficient imitation of the behavior of experts, and can learn better policies relative to other frameworks.

**Keywords** generative adversarial imitation learning; reinforcement learning; control sharing; deep reinforcement learning

---

Received October 24, 2021, accepted December 9, 2022

Supported in Part by the National Natural Science Foundation of China (U1808206)

## 1 Introduction

Decision-making problems<sup>[1]</sup> represent an important type of problem in the field of artificial intelligence<sup>[2]</sup>. Those problems pertain to identifying policy to achieve established goals<sup>[3]</sup>, for example, an intelligent vehicle that must make driving policy to achieve safe, efficient, and human-like driving. Reinforcement learning (RL) and imitation learning are two of the most promising methods to solve decision-making problems<sup>[4]</sup>.

RL is a powerful and popular framework that enables learners to deal with complex continuous control tasks<sup>[5]</sup>. The main idea of RL is to make the agent interact with the environment continuously. The agent can learn the policy by maximizing the cumulative reward expectation obtained from the environment. Although RL has achieved great success in agent learning, traditional RL is based on an unsupervised mode which cannot apply the prior knowledge of experts<sup>[6]</sup>. Current expert systems accumulate extensive prior knowledge, and this feature can provide guidance for agents and accelerate the learning process<sup>[7]</sup>. To fully utilize prior knowledge, researchers proposed the reward shaping<sup>[8]</sup> and control sharing<sup>[9]</sup> methods and applied expert rules to guide the reward and action in the learning process, respectively. Griffith and Subramanian, et al.<sup>[9]</sup> used the Lyapunov function to model the reward in the RL learning process and employed external extra reward to improve RL learning efficiency<sup>[10]</sup>. Knox and Stone applied the control sharing method TAMER<sup>[11]</sup> and proved that control sharing can improve learning efficiency<sup>[12]</sup>. However, the reward shaping and control sharing techniques are limited to the existing expert rules and cannot be generalized effectively.

At the other extreme, imitation learning<sup>[13]</sup> can obtain direct feedback from prior decision data. More specially, imitation learning can solve decision-making problems by imitating the samples demonstrated by experts. Generative adversarial imitation learning (GAIL)<sup>[14]</sup> is a method of imitative learning and uses the framework of a generative adversarial network to overcome the shortcomings of traditional imitative learning, such as the computational burden. GAIL shows excellent performance in large-scale problems and has been widely utilized<sup>[15]</sup>. Compared with RL, imitating learning represented by adversary network can make full use of prior knowledge such as expert rules and generate generalized rules which can flexibly cope with more complex situations<sup>[16]</sup>.

However, GAIL lacks a reward mechanism and cannot directly interact with the environment. Once the environment changes suddenly, a long period of accumulation is necessary to reflect the imitation learning, a situation which leads to slow convergence of the policy learning process<sup>[17]</sup>. When the environment changes, adapting to the new environment requires an extensive period and thus slows down the convergence of policy learning process.

This paper combines RL and imitation learning and proposes generative adversarial imitation learning based on control sharing (GACS) which automatically learns model constraint from expert samples and directly guides action policy in the learning process. Compared with the reward guide algorithm such as GAIL, this algorithm can directly learn the action policy from expert samples. To verify the effectiveness of the algorithm, we apply our algorithm in a driving task environment with varying traffic density and a typical strategy game of Atari. The algorithm eliminates the complexity of manually establishing expert rules and automatically learns model constraints from offline expert samples to guide the RL algorithm to learn

the action policy. The experimental results fully confirm that the GACS algorithm has better learning efficiency and generalization ability, and can learn better policies.

The rest of this paper is arranged as follows. The second section describes research related work and background issues. In the third section, we describe our GACS method and introduce how to guide the model free RL implementation to learn more efficiently. The experiment, results and discussion are given in the fourth section. The conclusion and further work are discussed in the fifth section.

## 2 Related Technologies

This section introduces the problem of RL and the concept of reward shaping, control sharing, and GAIL which are foundations of this paper.

### 2.1 MDP and RL

RL is a general class of algorithms in the field of machine learning that aims at allowing an agent to learn how to make sequential decisions in an environment wherein the only feedback consists of a scalar reward signal<sup>[8]</sup>. An MDP is a commonly adopted framework for RL problems and can be defined by a 4-tuple  $\{S, A, r, P\}$ , where  $S$  is a discrete or continuous state space;  $A$  is a discrete or continuous action space; and  $r : \bar{S} \times A \rightarrow R$  is the reward function, where state  $i \in S$ , after performing an action  $a \in A$ , transforms into state  $j \in S$ .

The decision-making objective is Formula (1), which is the total expected return function, where  $E(\cdot)$  is a mathematic expectation,  $\gamma$  is the discount factor, and  $r_t$  is an immediate reward for performing an action in a state at time  $t$ .

$$R_T = E \left[ \sum_{t=0}^T \gamma^t r_t \right]. \quad (1)$$

RL optimizes the object of the value function or policy to finally realize the control optimization. Assume that  $S_t$  and  $A_t$  states and actions set at time  $t$ .  $\pi_t$  is a policy at time  $t$ ,  $\pi = (\pi_0, \pi_1, \dots)$  is the MDP action policy set, and the action policy set is a mapping  $\pi : S \rightarrow A$ . The decision objective can be maximized with any initial state. The MDP state value function is as follows:

$$V^\pi(s) = E^\pi \left[ \sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s \right], \quad (2)$$

where  $E^\pi(\cdot)$  is the mathematical expectation of policy  $\pi$ .  $V^\pi(s)$  is the total expected return under policy  $\pi$  with a discount on the subsequent state. The value of a state  $S$  is defined under policy  $\pi$ , and  $V()$  is the expected return when starting in  $S$  and according to the policy  $\pi$ . The concept of the value function is introduced to optimize a policy. The action value function is defined as

$$Q^\pi(s, a) = E^\pi \left[ \sum_{t=0}^{\infty} \gamma^t r_t | s_t = s, a_t = a \right] \quad (3)$$

$Q$ -learning is inapplicable to high-dimensional state environments. In DQN<sup>[18]</sup>, the  $Q$  function can be approximated by a neural network. The  $Q$  function is represented by  $Q(s, a; \theta)$ . The DQN has two mechanisms for improving learning performance: Replay memory and target network. The state correlations are weakened by uniformly sampling from the replay memory. The

temporal difference (TD) loss function can be written as  $y - Q(s, a; \theta)^2$ , where  $y = Q(s', a'; \theta') + r$  and  $a' = \arg \max_a Q(s', a; \theta)$ .  $\theta$  represents the current  $Q$  function parameter, and  $\theta'$  represents  $n$  before the times  $Q$  function parameter, which is updated to  $\theta$  with a fixed frequency. The DQN can use high-dimensional information as the  $Q$ -learning inputs, such as when using pictures as the inputs. In deep  $Q$ -learning, the  $Q$  function can approximate by neural network, the  $Q$  function represented by  $Q(s, a; \theta)$  and parameterized by  $\theta$ . When training, actions are select from each time step according to an exploration policy, such as an  $\epsilon$ -greedy policy that selects the currently estimated best action  $\arg \max_{a \in A} Q(s', a | \theta')$  with probability  $1 - \epsilon$ , and takes a random exploratory action with  $\epsilon$  probability. The deep neural network can be trained by iteratively minimizing the following loss function:

$$L(\theta_i) = E \left[ \left( R + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i) \right)^2 \right]. \quad (4)$$

One key technology of DQN is experience replay<sup>[19]</sup>, which is adapted to break the strong correlations between consecutive state inputs during the training. The experience can be defined as a tuple  $e_t = (s_t, a_t, r_t, s_{t+1})$  in computer memory, where  $s_t$  is the state input at time  $t$ ,  $a_t$  is the action taken at time  $t$ ,  $r_t$  is the received reward at  $t$ , and  $s_{t+1}$  is the next state transited from  $s_t$  after taking  $a_t$ . Especially, recent experiences are stored to construct a replay memory  $D = e_1, \dots, e_N$ , where  $N$  is the memory size. Instead of using online data in the original order, training is performed  $\theta$  by sampling experiences from the replay memory to update the network parameters  $\theta$ .

## 2.2 The Reward Shaping and Control Sharing

In the reward shaping of RL, external extra reward guidance is used to improve the learning efficiency, an approach which is indirect. Some studies have extended the reward method to the partially observable MDP problems<sup>[20]</sup> and model-based RL<sup>[21]</sup>. Reward shaping has a considerable long-term impact on the development of RL. Ofir and Rosman proposed a Bayesian reward formation framework which uses prior belief that decays with experience to increase the reward distribution to reward shaping<sup>[22]</sup>. Another direct method is the control sharing method, which directly provides suggestions to the agent and controls the actions which the agent chooses to perform<sup>[23]</sup>. Yu and Wang, et al. applied the form of human intervention to the RL of control sharing and verified that the control sharing method was better than the reward method<sup>[24]</sup>. The method of control sharing entails taking human experience or expert rules as a specific action policy and directly acting on agents to obtain the policy. We define the policy  $\pi(s, a) \in \pi_a$  learned by agents using RL and the policy obtained from human prior knowledge or expert rules as  $\pi_h(s, a) \in \pi_h$ . We define  $\prod(\pi, \pi_h) \in \pi_a$  as the joint behavior decision of RL and expert rules. The control sharing policy proposed in this paper is a typical algorithm based on action policy. Unlike reward shaping, control sharing takes an approach that does not provide agent advice indirectly but makes agents directly choose an action according to expert rules or use action policy learned in the process of RL. We define the joint policy as follows:

$$\prod(\pi, \pi_h) = \pi(s, a)^l * \pi_h(s, a)^{(l-1)}, \quad (5)$$

where  $l$  is the Bernoulli random variable which controls the agent to select the action, and the action policy obtained by RL or by using expert rule. In this study, we use the prior

knowledge of expert rules to guide the RL. However, in practice, we often obtain very simple rules. Ascertaining the rules with wide coverage and high income is also difficult. However, we can collect some excellent samples to guide RL.

### 2.3 Generative Adversarial Network and Imitation Learning

The Generative adversarial networks (GAN) consists of two networks: A generator network-G and a discriminator network-D. The two networks compete and learn from each other with the idea of game theory. The purpose of a generator network is to learn the distribution of real sample data as far as possible until the real sample data cannot be recognized by a discriminator network. Conversely, the purpose of the discriminator network is to correctly distinguish whether the samples are from the realistic samples generated by the generator or from the real samples from the real world. We train both networks simultaneously, and they will become increasingly better over time.

The goal of imitation learning is to learn how to perform tasks directly from expert demonstration samples without any reward signal  $r$ . Generally, imitation learning involves two methods: 1) behavior cloning (BC), which learns policy by supervised learning, and whose learning samples entail state action pairs from expert tracks<sup>[25]</sup>; and 2) apprenticeship learning (AL)<sup>[26]</sup>, which assumes that the expert policy is optimal under some unknown rewards and learns policy by restoring rewards and solving planning problems. Given the composite error and covariate drift, BC has poor generalization properties. By contrast, AL has the advantage of learning the reward function which can be used to score the trajectories, but AL usually runs at a higher cost because the reward function must be solved repeatedly in each episode of RL.

GAIL is an apprenticeship learning method based on generative adversarial network. In the GAIL framework, the agent imitates the behavior of expert policy  $\pi_E$  by matching the generated state action distribution with the expert distribution. The GAIL objective function is expressed as

$$\min_{\pi} \max_{D \in (0,1)^{S \times A}} E_{\pi}[\log D(s, a)] + E_{\pi_E}[\log(1 - D(s, a))] - \lambda H(\pi), \quad (6)$$

where  $\pi$  is the policy that wishes to imitate  $\pi_E$ ,  $D$  is a discriminator classifier which tries to distinguish state-action pairs from the trajectories generated by  $\pi$  and  $\pi_E$ , and  $H(\pi) = E_{\pi}[-\log \pi(s|a)]$  is the  $\gamma$ -discounted causal entropy of the policy  $\pi_{\theta}$ <sup>[14]</sup>. GAIL and other imitation learning methods have been widely applied and studied. Hilleli, et al. applied imitation learning to generate a driving policy and used RL to optimize the policy<sup>[27]</sup>. Codevilla utilized imitation learning in an automatic driving environment and improved its scalability and limitations<sup>[28]</sup>. GAIL introduces a generative adversarial network into imitation learning, uses a generator to generate action, and employs a discriminator to judge whether or not the action comes from expert policy. In the above research, imitation learning methods train the decision model by imitating data in the expert field, such that the algorithm does not rely on the constraints of artificial expert rules to guide the learning of agents.

## 3 Method

In the real world, good samples are easier to obtain than perfect expert rules. For example, in autonomous driving, obtain excellent human driving data is more straightforward than

obtaining excellent expert rules<sup>[29]</sup>. Similarly, in the field of games, formulating perfect game playing strategies is difficult, but we can record the data of excellent players<sup>[30]</sup>.

In this research, we can obtain expert samples and learn a policy. To better guide the exploration and learning of RL and effectively improve the learning efficiency, we proposed GACS based on control sharing policy, as shown in Figure 1. Model constraints are automatically learned from expert samples, and action policies are directly obtained from expert samples. The idea of GACS is similar to the generative adversarial network, which also consists of a generator and discriminator. The generator network in this work is composed of deep reinforcement learning (DRL)<sup>[31]</sup>, and the discriminator network is composed of a neural network. Taking the automatic driving environment as an example, the generator network uses DQN to generate more realistic sample tracks until it deceives the discriminator network, which is equivalent to two classifiers. After continuous training of the neural network, its recognition ability is improved, and the real samples and forced true samples are distinguished. The two compete with each other until an algorithm Nash equilibrium is reached. The definition is as follows:

$$\min_{\psi} \max_{\theta} V(\theta, \varphi) = E_{(s,a) \sim \chi_E} [\log D_{\psi}(s, a)] + E_{(s,a) \sim \chi_{\theta}} [\log(1 - D_{\psi}(s, a))] - \lambda_0 C(\pi_{\theta}) + \lambda_1 r(\pi_{\theta}) + \lambda H(\pi), \quad (7)$$

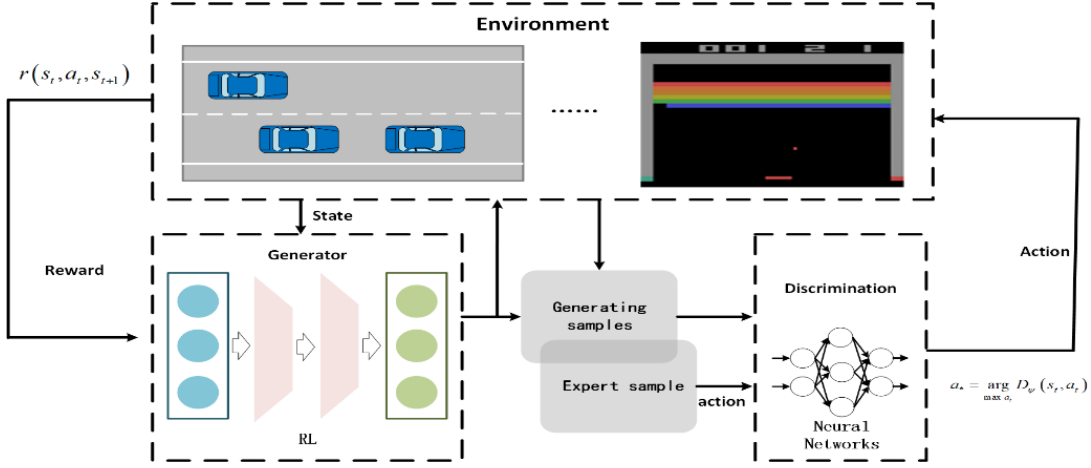
where  $\chi_{\theta} = (s_1, a_1), (s_2, a_2), \dots, (s_T, a_T)$  represents the realistic trajectory generated by the generator,  $\chi_E$  is the true track represented by the expert sample, and  $D_{\psi}$  is the discriminator  $\psi$  and  $\theta$  are the parameters of discriminator and generator respectively.  $\pi_{\theta}$  indicates the policy under the current parameter. In our method, we setting  $\lambda_0 \in (0, 1)$ ,  $D_{\psi}(s, a)$  indicates the probability which is the discriminator used to judge the probability of state action pairs  $(s, a)$  generated by expert policy. According to the probability, we can deduce the expert policy in the current state  $C(\pi_{\theta}) = \arg \max_{\pi_{\theta}} D_{\psi}(s, a)$ , use  $C(\pi_{\theta})$  constraint generator generates policy directly. In the initial stage of training, since the effect of the generator sample is much worse than that of the discriminator sample, we can increase the value of  $\lambda_0$  to guide the generator with higher probability. When the generator sample distribution is close to the expert distribution, the generator is encouraged to explore more by reducing the value of  $\lambda_0$  to improve the generalization ability of the model.  $\lambda_1 > 0$  is a hyperparameters, when the performance of experts is not the best, any policy in discriminator training will also be suboptimal. To solve this problem, we add reinforcement learning reward function  $r(\pi_{\theta}) = r_{\text{safe}} + r_{\text{speed}}$ , for policy optimization. The following experiments show that by using the guidance generator to further explore the environment on the basis of expert samples to achieve better results than expert samples.  $H(\pi_{\theta})$  is the policy casual entropy defined as  $E_{\pi_{\theta}}[-\log \pi_{\theta}(a'|s, a)]$  with hyper-parameter  $\lambda$ .

In GACS, combining control sharing with GAIL, and the control sharing mechanism is used to replace reward and guidance mechanism. GACS uses the policies learned from expert samples to guide the agent's action in the process of confrontation directly. Similar to GAIL, the training process of game between the policy and reward function of GACS can be divided into the following four steps: 1) Use the neural network to train  $D_{\psi}$  which is allocated to policy  $\chi_E$  as much as possible rewards and policy  $\chi_{\theta}$  is as small as possible rewards; 2) Under the

constraints of the expert sample policy, use the DQN to train  $\chi_\theta$ , so that it is steadily updated in the direction of maximizing the gradient of the accumulated reward value, in order to approach (Equation (7)); 3) repeat 1) and 2),  $r(\pi_\theta)$  and  $C(\pi_\theta)$  will be closer to the real reward function policy, and guide  $\chi_\theta$  to approach  $\chi_E$ . 4) In the end, the policy and reward function reach the Nash equilibrium. At this time,  $D_\psi$  can no longer correctly distinguish the source of  $\chi_\theta$  and  $\chi_E$ , and the  $\chi_\theta$  generated sample distribution can perfectly fit the sample distribution of  $\chi_E$ , so no adjustment is needed. The specific process of the algorithm is as follows:

$$Q\left(s_t \middle| \arg \pi_\theta(s_t)\right) \leftarrow Q\left(s_t \middle| \arg \pi_\theta^t(s_t)\right) + \alpha \left[ r(s_t, a, s_{t+1}) + \gamma \max_a Q(s_{t+1} | C(\pi_\theta)) - Q\left(s_t \middle| \arg \pi_\theta(s_t)\right) \right]. \quad (8)$$

$Q(s_t | \arg \pi_\theta(s_t))$  indicates the  $Q$  value of reinforcement learning at the current time. The  $Q$  value is updated by using the current  $Q$  value:  $Q(s_t | \arg \pi_\theta(s_t))$ . Formula (8) uses the reinforcement learning  $Q$ -learning update method. We use the method of strategy to express its action. We build the discriminator and generator in Line 1. In Lines 3 to 5, we use DRL to train the generator and output realistic samples into the generation network. From Lines 6 to 12, the real-world expert samples and the realistic samples of the generator are trained against each other, and the probability of the samples being true samples is expressed as the output. According to the probability of samples, the algorithm decides the probability of the output action to guide the agent's learning.



**Figure 1** Generative adversarial imitation learning with control sharing

## 4 Experimental Evaluation

In the future, we hope our algorithm can be used in real environments, so we try to test the performance of our algorithm in different environments. We chose the Atari game environment where most of the reinforcement learning algorithms are tested, and we tested the algorithms in a simulated highway environment. This shows that our algorithm may be effective in different tasks, and makes some preparation tests for future applications in real environments.

**Algorithm 1** Generative Adversarial Control Sharing

---

```

1: Input: Expert trajectories  $\chi_E$ , initial policy and discriminator parameters  $\theta_0, \psi_0$ . Given
   empty experience buffer  $B$ .
2: for  $i = 0, 1, \dots$  do
3:   for  $j = 0, 1, \dots$  do
4:     Act on environment:  $a_t = \max_a Q^*(s_t, a_t; \theta)$ 
5:     Push  $(s, a, s')$  into  $B$ 
6:   end for
7:   Sample trajectories  $\chi_E$ 
8:   Update the discriminator parameters from  $\psi_i$  to  $\psi_{i+1}$  with the gradient
9:    $E_{\sim \chi_E} [\nabla_{\psi} \log(D_{\psi}(s, a)) + E_{\sim \chi_{\theta}} [\nabla_{\psi} \log(1 - D_{\psi}(s, a))]]$ 
10:  Specifically, take a KL-constrained natural gradient step with:
11:   $E_{\sim \chi_E} [\nabla_{\theta} \log \pi_{\theta}(a|s)Q(s, a)] - \lambda_0 \nabla_{\theta} C(\pi_{\theta}) + \lambda_1 \nabla_{\theta} (\pi_{\theta}) + \lambda \nabla_{\theta} H(\pi_{\theta})$ 
12:  where  $Q(s, a)$  update the parameters of  $\theta$  by Equation (7) with policy:
13:   $C(\pi_{\theta}) = \arg \max_{\pi_{\theta}} D_{\psi}(s, a)$ 
14: end for

```

---

First, we apply the algorithm to a driving task with changing traffic density. In this task, the agent chooses a lane change decision adaptively according to the changing state of the traffic environment, and imitates the driving action of expert samples. Then we extend it to an Atari game (Breakout). Agents must consider the movement of the ball on the screen and the destruction of bricks to choose their own skills and take corresponding strategies. For these two tasks, our algorithm evaluates four baselines: PPO<sup>[32]</sup>, DQN<sup>[33]</sup>, expert rule-based control sharing policy (CS), rewarding shaping policy (RS) and GAIL. On the basis of expert rules, the control sharing policy and reward shaping use artificial expert rules as model constraints to guide the agent’s learning process, a technique which is equivalent to the real-time guidance of experts in agent learning.

#### 4.1 Autonomous Driving Experiment

We use RL method to solve the cooperative problem of intelligent vehicle. Firstly, the intelligent vehicle learning problem is defined as RL problem. Then, we describe the state, action and reward setting of MDP model which using in RL method to train a single intelligent vehicle. In the driving policy learning method based on RL, each intelligent vehicle on the road is regarded as an agent, the whole road and vehicle information are regard as a state.

##### 4.1.1 State Setting

In vehicle RL, the state  $S$  is the main factor that affects the decision of the vehicles. The state of each vehicle consists of a ten tuple of  $(l, v_0, v_i, d_i)$  ( $i = 1, 2, 3, 4$ ). We use the symbol  $l$  to denote the lane in which the intelligent vehicle is located, with  $l = 0$  denoting the intelligent vehicle on the driving lane and  $l = 1$  denoting the intelligent vehicle on the overtaking lane. In this paper, we use index 0 to represent the state of intelligent vehicle, and  $i$  ( $i = 1, 2, 3, 4$ ) to represent the index of neighboring vehicles around the intelligent vehicle. As shown in Figure



2, we use index 0 to represent intelligent vehicle, the index 1 to represent the lead vehicle which is on the driving lane, the index 2 to represent the lag vehicle which is on the driving lane, the index 3 to represent the lead vehicle which is on the overtaking lane and the index 4 to represent the lag vehicle which is on the driving lane. The lead vehicle is the nearest vehicle in front of the intelligent vehicle and is in the same lane. The lag vehicle is the nearest vehicle in rear of the intelligent vehicle and is in the same lane.

The  $v_0$  is the velocity of the intelligent vehicle; and the  $v_i$  is the velocity of neighboring vehicles around the intelligent vehicle and the distance  $d_i$  is the distance between neighboring vehicles around the intelligent vehicle ( $i = 1, 2, 3, 4$ ). As shown in Figure 2. The  $v_1$  is the velocity of the lead vehicle when the lead vehicle is on the driving lane. The  $v_2$  is the velocity of the lag vehicle when the lag vehicle is on the driving lane. The  $v_3$  is the velocity of the lead vehicle when the lead vehicle is on the overtaking lane. The  $v_4$  is the velocity of the lag vehicle when the lag vehicle is on the overtaking lane.  $d_1$  is the distance between the intelligent vehicle and the lead vehicle when the lead vehicle is on the driving lane.  $d_2$  is the distance between the intelligent vehicle and the lag vehicle when the lag vehicle is on the driving lane.  $d_3$  is the distance between the intelligent vehicle and the lead vehicle when the lead vehicle is on the overtaking lane.  $d_4$  is the distance between the intelligent vehicle and the lag vehicle when the lag vehicle is on the overtaking lane.

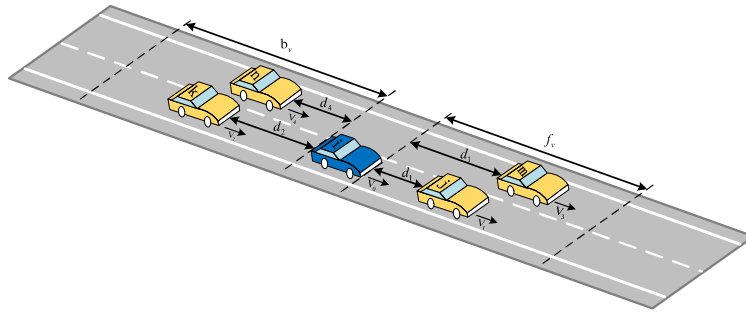
Setting a low dimensional state space can reduce the complexity of the problem and speed up the learning process, so the state can be reduced from ten dimensions  $(l, v_0, v_i, d_i)$  to five dimensions  $(l, t_i)$ ,  $t_i$  is the remnant reaction time (RRT), which indicates the shortest time for a vehicle to collide with surrounding vehicles. Assuming that the lag vehicle takes a braking action at  $T_N$  time to just meet the shortest safety distance (just avoiding a collision), the remaining reaction time refers to the remaining time of the lag vehicle before  $T_N$  time. Therefore, we use the ratio of the remnant reaction distance and the speed of the lag vehicle instead of the relative speed. We define the symbol  $t_1$  for the remnant reaction time of the intelligent vehicle and neighboring vehicles. We explain  $t_i$  in detail here. When the lead vehicle on the driving lane the is the remnant reaction time between the intelligent vehicle and the lead vehicle. When the lag vehicle on the driving lane the  $t_2$  is the remnant reaction time between the intelligent vehicle and the lead vehicle. When the lead vehicle on the overtaking lane the  $t_3$  is the remnant reaction time between the intelligent vehicle and the lead vehicle. When the lag vehicle on the overtaking lane the  $t_4$  is the remnant reaction time between the intelligent vehicle and the lag vehicle. In details,  $t_1 = (d_1 - d_{s1})/v_0$ ,  $t_3 = (d_3 - d_{s3})/v_0$ ,  $t_2 = (d_2 - d_{s2})/v_2$ ,  $t_4 = (d_4 - d_{s4})/v_4$ .

$d_{s_i}$  is the shortest safe distance between two vehicles. For example,  $d_{s1}$  is the shortest safety distance between the intelligent vehicle and the lead vehicle on the driving lane.  $d_{s2}$  is the shortest safety distance between the intelligent vehicle and the lag vehicle on the driving lane. The shortest safety distance is the distance difference between the two vehicles when they decelerate from the current velocity to 0. In more detail, if the distance between two vehicles is greater than the shortest safe distance, there will be no collision; if it is less than the shortest safe distance, there will be a collision.

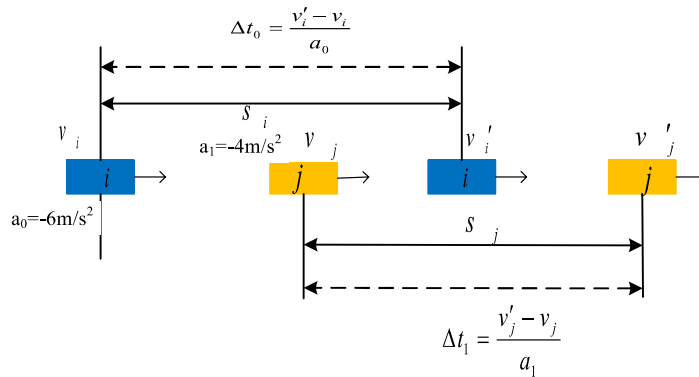
As shown in Figure 3, We use  $s_0$  (the distance of intelligent vehicle when it decelerates from current speed to 0) and  $s_1$  (the distance of lead intelligent vehicle when it decelerates

from current speed to 0) in Figure 3 to calculate  $d_{s1}$ . The derivation process is as follows:  $d_{s1} = s_0 - s_1$ ,  $s_0 = v_0 \Delta t_0 + 1/2 * a_0 \Delta t_0^2$ ,  $\Delta t_0 = \frac{v'_0 - v_0}{a_0}$ . Combining  $d_{s1}$  and  $\Delta t_0$  get  $s_0 = \frac{(v'_0)^2 - v_0^2}{2 * a_0}$ , where  $v_0$  is current velocity of the intelligent vehicle, the shortest safe distance requires the distance difference between two vehicles decelerate from current speed to 0 at the same time, therefore the velocity  $v'_0$  value is 0.  $a_0$  is the deceleration of intelligent vehicle and the value is  $a_0 = 6\text{m/s}^2$ <sup>[34]</sup>.  $\Delta t_0$  is the time required for the velocity to from  $v_0$  to  $v'_0$ . Since  $v'_0$  is 0, it can be eliminated in. It is necessary to substitute a negative sign in the calculation of distance by using deceleration  $a_0$ , therefore the negative sign can cancel in, and finally we get the distance:  $s_0 = v_0^2 / (2 * a_0)$ . In the same way, we can get the distance:  $s_1 = v_1^2 / (2 * a_1)$ .  $v_1$  is the velocity of the lead vehicle. According to [34], the deceleration of the lead vehicle is  $a_1 = 4\text{m/s}^2$  and use a predefined distance 10 m to represent the reaction distance and the lead vehicle's length. Finally, we get the shortest safe distance:  $d_{s1} = v_0^2 / (2a_0) - v_1^2 / (2a_1) + 10$ . Similarly, we can get  $d_{s2} = v_2^2 / (2a_0) - v_0^2 / (2a_1) + 10$ , where  $v_2$  is the velocity of the lag vehicle. The specific derivation process is the same as  $d_{s1}$ , and the other parameters are the same as above. Further, we can get  $d_{s3} = v_0^2 / (2a_0) - v_3^2 / (2a_1) + 10$  and  $d_{s4} = v_0^2 / (2a_0) - v_4^2 / (2a_1) + 10$  use the same derivation process.

When the decision is not to change lanes and there is a vehicle in front, the vehicle following velocity is  $v_p$ , which is the following velocity when the intelligent vehicle in the driving lane or overtaking lane. The following kind of heuristics can be adopted to realize the vehicle-following



**Figure 2** Status representation of No. 0 intelligent vehicle



**Figure 3** Schematic diagram of intelligent vehicle driving parameter calculation

mode 1 and compute the planned speed: If  $(d_1 - d_f > 10 \text{ m})$  and  $(v_1 - v_0 > 3.6) \Rightarrow v_p = (v_0 + 0.25(d_1 - d_f) + 1.5(v_1 - v_0)) \text{ m/s}$ , where  $d_f = (v_0^2/2a_1 + 5) \text{ m}$  is the vehicle following distance with  $a_1 = -6\text{m/s}^2$  and 5 m is the vehicle length<sup>[25]</sup>. The heuristic means that the vehicle should move faster to catch up with the lead vehicle if their distance is more than 10 m and their relative velocity is more than 3.6 m/s. Other situations can be transformed to this kind of heuristics similarly.  $v_t$  is the driving velocity before the next decision. When the intelligent vehicle in the driving lane, the maximum velocity of the  $v_t$  is  $v_t = 30 \text{ m/s}$ . When the intelligent vehicle in the overtaking lane, the maximum velocity of the  $v_t$  is  $v_t = 40 \text{ m/s}$ <sup>[34]</sup>.

#### 4.1.2 Action and Reward Setting

When designing the action policy of autonomous driving, we consider two kinds of actions  $a = (0, 1)$ ,  $a = 0$  indicates that the vehicle is driving in the driving lane, and  $a = 1$  indicates that the vehicle is driving in the overtaking lane. Combined with the actual driving situation, we prioritize safety and speed in autonomous driving to improve the traffic efficiency and reduce the time cost to evaluate the chosen actions. We define the reward function as follows:

$$r = \begin{cases} \min(t_1, t_2), & l = 0 \text{ and } d_1 > 3 \text{ and } d_2 > 3, \\ \min(t_3, t_4), & l = 1 \text{ and } d_3 > 3 \text{ and } d_4 > 3, \\ -40, & \text{if collision,} \end{cases} \quad (9)$$

$$r_{\text{speed}} = \begin{cases} v_p - v_t, & \text{if } v_p > v_t, \\ 0, & \text{else.} \end{cases} \quad (10)$$

Equation (8) means that when a vehicle collides, it will be severely punished, indicating that vehicle safety is the most important priority. When a vehicle is driving in a safe state, the distance between the vehicle and the lead/lag vehicle is greater than a certain safe range  $d_i > 3$ . It means that the longer RRT keep in an emergency, the vehicle will obtain the higher safety. In all other cases, when the distance between the vehicle and the lead/lag vehicle is less than 3 m, the vehicle will also be punished because the distance between the vehicles is too short, which can easily lead to collision accidents.

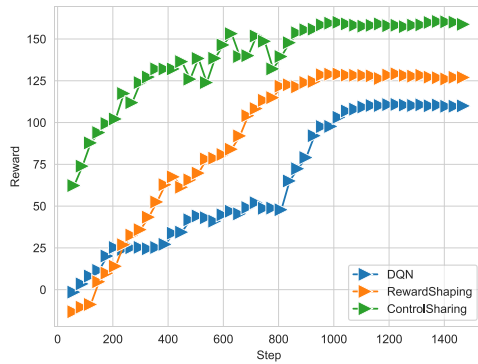
The reward  $s_{\text{speed}}$  value means that when the autonomous vehicle chooses a lane, it can maximize its speed under the premise of ensuring safety. When a vehicle is following at the planned speed  $v_p$ , the vehicle will not be punished if the speed  $v_p$  is controlled to be less than  $v_t$  to ensure safety; when the vehicle chooses to overtake another vehicle for faster driving, the difference between the current actual speed  $v_p$  and  $v_t$  is used as the reward of the vehicle to avoid over-conservative decision making of vehicle selection. Note that our intelligent vehicle learning environment is a simple highway environment. The specific experimental parameters are defined in Table 1. Our hyperparametric selection is based on the literature [14], [24] and [32].

Figure 4 shows the comparison of the cumulative average rewards for the training of DQN, CS, RS, GAIL and GACS, wherein the horizontal axis is the step (each step represents 10 episodes) and the vertical axis represents the average cumulative reward of the 10 episodes in each step. Figure 4(a) indicates that the externally guided RL method (RS and CS) guides

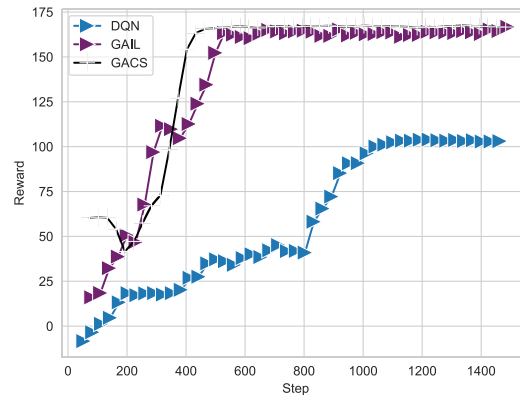
the agent to converge quickly and obtain higher rewards. As can be seen from Figure 4(b), the DQN algorithm gradually converges after about 800 steps. However, due to expert rules, the GAIL and GACS methods have already generated the guidance of adversarial samples. The GAIL converges at about 500 steps, and the GACS converges at about 400 steps. The average cumulative reward of GACS is obviously better than that of DQN, and the learning efficiency of the GACS algorithm in the training process is also obviously better than that of GAIL. Further, as the GACS method adopts the technique of direct action guidance, the training starting point is higher and this direct guidance can converge faster.

**Table 1** Hyperparameters of DQN, GAIL and GACS and road traffic simulation parameters

Hyper parameters	value	Parameters	value
Learning rate	0.1	Starting speed $v_0$	[18/s, 27 ms]
Discount rate	0.95	Overtaking lane speed limit	40 m/s
Exploration rate	0.1	Driving lane speed limit	30 m/s
Exploration Decay	0.9	Maximum acceleration	6 m/s <sup>2</sup>
		Minimum deceleration	-2 m/s <sup>2</sup>



(a) The reward and convergence speed of DQN, Rewarding Shaping and Control Shaping

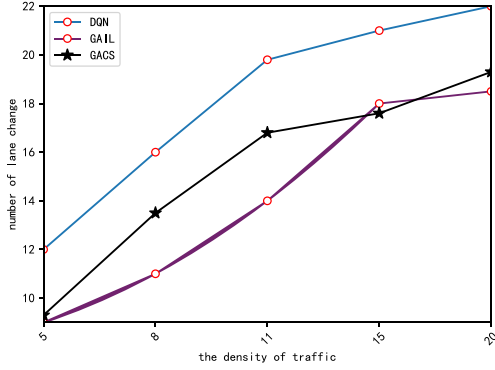


(b) The reward and convergence speed of DQN, GAIL and GACS

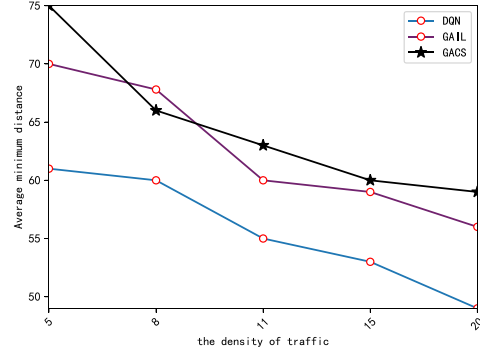
**Figure 4** Performance comparison between the proposed and other algorithms

In addition, we test the overall performance and micro fluidity of traffic density algorithms. Figure 5 shows the comparison of the average speed, lane change times, and minimum forward distance according to DQN, GAIL, and GACS when traffic density increases from 5 to 20 vehicles. With the increase of vehicle density, the GACS method has less lane change times than the DQN method, and although the speed decreases with the increase of traffic density, GACS still outperforms the DQN and GAIL methods. Furthermore, with the increase of vehicle density, the DQN method is closer to other vehicles, and GAIL and GACS are farther and safer than RL. GAIL and GACS have similar speed with the increase of traffic density, but GACS performs better than GAIL in the shortest distance between vehicles and lane changing times, a situation which can maintain a larger safety distance and proves that the GACS algorithm can learn better policy. The experiment fully proves that the result of using control sharing

algorithm to guide intelligent vehicle lane changing is better than employing the DQN algorithm without a model. Finally, we compared the time consumed in the training convergence of DQN, GAIL, and GACS. The externally guided learning methods GAIL and GACS require less time to converge than the unguided RL method. At the same time, given that GACS directly acts on the learning action, the time required for convergence is less than that of the indirect learning method GAIL.



(a) Average velocity at different vehicle densities

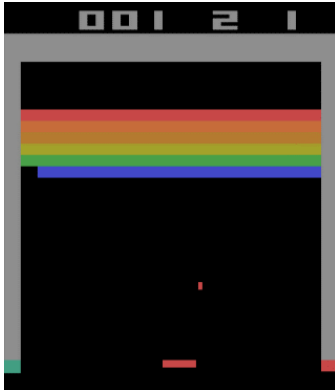


(b) Minimum vehicle distance under different vehicle densities

**Figure 5** Micro traffic indicators

## 4.2 Atari Experiment

We verified the performance of GACS in a Breakout game. As shown in Figure 6, the breakout is a typical policy game in Atari. In the Breakout game, we use the video image as the state input of RL. Within the scope of the screen, the player controls a small paddle that can be moved left and right to control the direction of the ball. When the ball hits the brick above, the brick is broken and the player is rewarded. When all the bricks are broken, the player wins. The Atari game environment is a very good test environment, and many RL research focus on this environment. For example, most of such works do not use image observations, but several recent investigations have incorporated images into real-world<sup>[35–40]</sup>.



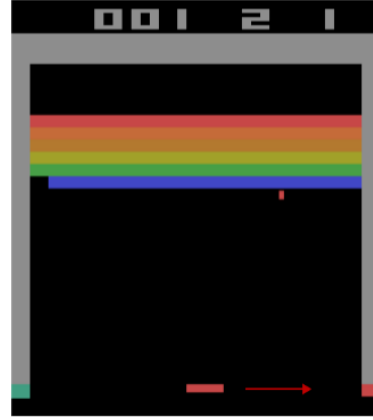
(a) The game starts, and the agent pops up a ball



(b) The ball files along the path



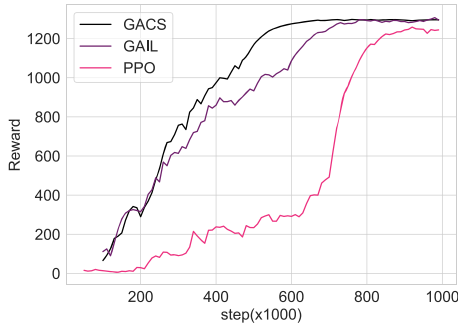
(c) Without using the GACS method, the agent will not move in advance to predict the position of the ball



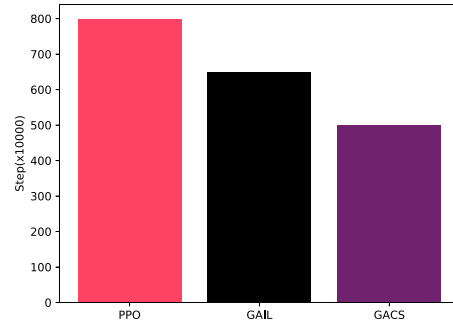
(d) When using the GACS method, the agent experience will move in advance to predict the position of the ball

**Figure 6** Using GACS method to train agents to play atria game

We also ran GACS on the Arcade Learning Environment benchmark (Breakout) and compared it against the implementations of GAIL and PPO. We used the same policy network architecture for all three algorithms. To solve the problem of partial observability during training, we spliced one frame every four consecutive frames. As shown in Figure 6(a), we used every 1000 frames as a step, and each method uses 1000 steps. As no expert rule sample exists in the Atari game, our GAIL method employs the sample trained by PPO as the counter sample to learn. Figure 6(b)(c) shows that the three methods can bounce the ball to the specified position, but Figure 6(d) shows that our method can move the racket to the right in advance, and the other two methods do not predict the action in advance. At the same time, we recorded the convergence sample data of the PPO method as the guidance sample of GACS. Figure 7(a)(b) indicate that the convergence speed of the PPO method is slower than that of GAIL and GACS. Although GAIL can obtain the same reward value as GACS because the latter is a direct guidance method, the convergence speed of GAIL is faster than that of GACS.



(a) The convergence curves of GACS, GAIL and PPO



(b) Training step required for convergence of GACS, GAIL and PPO methods

**Figure 7** The three methods of GACS, GAIL, and PPO train an agent to play the reward curve of the Atria game and the steps required for convergence

## 5 Conclusions

In this paper, we propose a generative adversarial learning method based on control sharing (GACS). Compared with the model-free method, our technique needs less interaction with the environment and fewer parameters to be adjusted. The experimental results show that: 1) By using the existing or manually designed expert rules as the model, the learning efficiency of the control sharing algorithm significantly outperforms that of a single expert rule decision and model-free algorithm (DQN). 2) The model based on expert rules needs manual design, a feature which is usually unrealistic for practical problems. The GACS does not use expert rules and uses only expert samples as guidance. GACS is further better than the generative adversarial imitation learning algorithm. Through the practice and verification in an autonomous driving simulation environment and an Atari game environment, the feasibility of the GACS algorithm to realize the efficient decision-making of an agent is fully confirmed. The GACS algorithm can automatically learn model constraints from expert samples and then guide an agent to explore and learn a model-free algorithm. The proposed framework eliminates the complexity of establishing expert rules manually and can be used in practical application.

## References

- [1] Kou G, Yang P, Peng Y, et al. Evaluation of feature selection methods for text classification with small datasets using multiple criteria decision-making methods. *Applied Soft Computing*, 2020, 86: 105836.
- [2] Li T, Kou G, Peng Y, et al. Classifying with adaptive hyper-spheres: An incremental classifier based on competitive learning. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2017, 50(4): 1218–1229.
- [3] Li T, Kou G, Peng Y, et al. An integrated cluster detection, optimization, and interpretation approach for financial data. *IEEE Transactions on Cybernetics*, 2022, 52(12): 13848–13861.
- [4] Zhu Y K, Wang Z Y, Merel J, et al. Reinforcement and imitation learning for diverse visuomotor skills. *arXiv preprint arXiv: 1802.09564*, 2018.
- [5] Sutton R S, Barto A G. *Introduction to reinforcement learning*. Cambridge: MIT Press, 1998, 135.
- [6] Ramachandran D, Amir E. Bayesian inverse reinforcement learning. *IJCAI*, 2007, 7: 2586–2591.
- [7] Gimelfarb M, Sanner S, Lee C G. Reinforcement learning with multiple experts: A Bayesian model combination approach. *Advances in Neural Information Processing Systems*, 2018: 9528–9538.
- [8] Ng A Y, Harada D, Russell S. Policy invariance under reward transformations: Theory and application to reward shaping. *ICML*, 1999, 99: 278–287.
- [9] Griffith S, Subramanian K, Scholz J, et al. Policy shaping: Integrating human feedback with reinforcement learning. *Advances in Neural Information Processing Systems*, 2013: 2625–2633.
- [10] Dong Y L, Tang X C, Yuan Y. Principled reward shaping for reinforcement learning via Lyapunov stability theory. *Neurocomputing*, 2020, 393: 83–90.
- [11] Knox W B, Stone P. Interactively shaping agents via human reinforcement: The TAMER framework. *Proceedings of the 5th International Conference on Knowledge Capture*, 2009, 9–16.
- [12] Knox W B, Stone P. Reinforcement learning from simultaneous human and mdp reward. *AAMAS*, 2021: 475–482.
- [13] Ho J, Gupta J, Ermon S. Model-free imitation learning with policy optimization. *International Conference on Machine Learning*, 2016: 2760–2769.
- [14] Ho J, Ermon S. Generative adversarial imitation learning. *Advances in Neural Information Processing Systems*, 2016: 4565–4573.
- [15] Zhang T T, Ji H, Sil A. Joint entity and event extraction with generative adversarial imitation learning. *Data Intelligence*, 2019, 1(2): 99–120.
- [16] Yang T P, Hao J Y, Meng Z P, et al. Efficient deep reinforcement learning via adaptive policy transfer. *Twenty-Ninth International Joint Conference on Artificial Intelligence and Seventeenth Pacific Rim*

- International Conference on Artificial Intelligence, 2020.
- [17] Chi N D, Quach K G, Luu K, et al. Learning from longitudinal face demonstration where tractable deep modeling meets inverse reinforcement learning. *International Journal of Computer Vision*, 2019, 127(6-7): 957–971.
  - [18] Mnih V, Kavukcuoglu K, Silver D, et al. Playing Atari with deep reinforcement learning. *arXiv preprint arXiv: 1312.5602*, 2013.
  - [19] Fang M, Li Y, Cohn T. Learning how to active learn: A deep reinforcement learning approach. *arXiv preprint arXiv: 1708.02383*, 2017.
  - [20] Asmuth J, Littman M L, Zinkov R. Potential-based shaping in model-based reinforcement learning. *AAAI*, 2008: 604–609.
  - [21] Eck A, Soh L K, Devlin S, et al. Potential-based reward shaping for finite horizon online pomdp planning. *Autonomous Agents and Multi-Agent Systems*, 2016, 30(3): 403–445.
  - [22] Marom O, Rosman B S. Belief reward shaping in reinforcement learning. *AAAI*, 2018.
  - [23] Fernandez F, Veloso M. Probabilistic policy reuse in a reinforcement learning agent. *Proceedings of the fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, 2006: 720–727.
  - [24] Yu C, Wang D X, Yang T P, et al. Adaptively shaping reinforcement learning agents via human reward. *Pacific Rim International Conference on Artificial Intelligence*, 2018: 85–97.
  - [25] Torabi F, Warnell G, Stone P. Behavioral cloning from observation. *arXiv preprint arXiv: 1805.01954*, 2018.
  - [26] Abbeel P, Ng A Y. Apprenticeship learning via inverse reinforcement learning. *Proceedings of the 21st International Conference on Machine learning*, 2004.
  - [27] Hilleli B, Ran E Y. Toward deep reinforcement learning without a simulator: An autonomous steering example. *AAAI*, 2018.
  - [28] Codevilla F, Santana E, Lopez A M, et al. Exploring the limitations of behavior cloning for autonomous driving. *Proceedings of the IEEE International Conference on Computer Vision*, 2019: 9329–9338.
  - [29] Zhu B, Jiang Y D, Zhao J, et al. Typical-driving-style-oriented personalized adaptive cruise control design based on human driving data. *Transportation Research Part C: Emerging Technologies*, 2019, 100: 274–288.
  - [30] Cai X Q, Ding Y X, Jiang Y, et al. Expert-level atari imitation learning from demonstrations only. *arXiv preprint arXiv: 1909.03773*, 2019.
  - [31] Fujimoto S, Meger D, Precup D. Off-policy deep reinforcement learning without exploration. *International Conference on Machine Learning*, 2019: 2052–2062.
  - [32] Liu B, Cai Q, Yang Z R, et al. Neural trust region/proximal policy optimization attains globally optimal policy. *Advances in Neural Information Processing Systems*, 2019: 10565–10576.
  - [33] Fan J Q, Wang Z R, Xie Y C, et al. A theoretical analysis of deep Q-learning. *Learning for Dynamics and Control*, 2020: 486–489.
  - [34] Li X, Xu X, Zuo L. Reinforcement learning based overtaking decision-making for highway autonomous driving. *2015 Sixth International Conference on Intelligent Control and Information Processing 2015*: 336–342.
  - [35] Finn C, Tan X Y, Duan Y, et al. Deep spatial autoencoders for visuomotor learning. *2016 IEEE International Conference on Robotics and Automation*, 2016: 512–519.
  - [36] Finn C, Levine S. Deep visual foresight for planning robot motion. *2017 IEEE International Conference on Robotics and Automation*, 2017: 2786–2793.
  - [37] Ebert F, Finn C, Lee A X, Levine S. Self-supervised visual planning with temporal skip connections. *arXiv preprint arXiv: 1710.05268*, 2017.
  - [38] Piergiovanni A, Wu A, Ryoo M S. Learning real-world robot policies by dreaming. *arXiv preprint arXiv: 1805.07813*, 2018.
  - [39] Paxton C, Barnoy Y, Katyal K, et al. Visual robot task planning. *2019 International Conference on Robotics and Automation*, 2019: 8832–8838.
  - [40] Rybkin O, Pertsch K, Jaegle A, et al. Unsupervised learning of sensorimotor affordances by stochastic future prediction. *arXiv preprint arXiv: 1806.09655*, 2018.