10th International Conference on Intelligent Control and Information Processing
Marrakesh, Morocco, December 14-19, 2019

Comparison of Control Methods Based on Imitation Learning for Autonomous Driving

Yinfeng Gao[1,2], Yuqi Liu[1], Qichao Zhang[1], Yu Wang[3]
1: The state Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China,
2: School of Automation and Electrical Engineering, University of Science and Technology Beijing, Beijing 100083, China,

Dongbin Zhao[1], Dawei Ding[2], Zhonghua Pang[4], Yueming Zhang[4]
3: China Automotive Technology&Research Center Co. Ltd, Tianjin 300300, China
4: Key Laboratory of Fieldbus Technology and Automation of Beijing, North China University of Technology, 100093, China

***Abstract*—Recently, some learning-based methods such as reinforcement learning and imitation learning have been used to address the control problem for autonomous driving. Note that reinforcement learning has strong reliance on the simulation environment and requires a handcraft design of the reward function. Considering different factors in autonomous driving, a general evaluation method is still being explored. The purpose of imitation learning is to learn the control policy through human demonstrations. It is meaningful to compare the control performances of current main imitation learning methods based on the provided dataset. In this paper, we compare three typical imitation learning algorithms: Behavior cloning, Dataset Aggregation (DAgger) and Information maximizing Generative Adversarial Imitation Learning (InfoGAIL) in the The Open Racing Car Simulator (TORCS) and Car Learning to Act (CARLA) simulators, respectively. The performance of algorithms is evaluated on lane-keeping task in racing and urban environment. The experiment results show DAgger performs best in simple lane keeping problem, and InfoGAIL has the unique advantage of distinguishing different driving styles from expert demonstrations.**

***Keywords*—autonomous driving, imitation learning, lane keeping, raw visual inputs**

I. INTRODUCTION

Reinforcement learning (RL) is a learning-based decision making method, where the agent learns the optimal policy through interacting with environment [1-3]. It has been used to address many practical problems such as advertising recommendation, video games and so on. However, it is still under discussion whether the vehicle controller [4] designed by RL algorithm can achieve a satisfactory result in real world driving scenarios, because the performance of RL highly depends on the predefined reward function [5-8]. Note that the autonomous vehicle is usually in a non-stationary environment, and many implicit factors such as the safety, comfort level, energy efficiency and so on should be considered, which make it difficult to define a proper reward function for RL [9, 10].

Assuming human drivers take the approximately optimal policy, imitation learning (IL) method can directly learn an similar policy through the demonstrations from human drivers. In general, IL can be divided into two categories: supervised IL and inverse reinforcement learning (IRL) [11]. Note that behavior cloning (BC) [12] is one of the supervised IL approaches, which learns a mapping from state to action through supervised learning over the provided state-action pairs data. It is easy to train a control policy with the development of deep neural network. The disadvantage is that BC requires massive human expert data. Furthermore, it can only perform well in a limited number of scenarios which are from expert demonstrations, and lacks the generalization ability to other scenarios. As another supervised IL method, Dataset aggregation (DAgger) [13] makes an improvement by approaching the real state-action space as far as possible. It guides the agent recovering from bad actions based on expert revised samples. Obviously, it enhances the generalization ability compared with BC. Unfortunately, it is very expensive to obtain the expert revised samples. On the other hand, IRL introduces the idea of RL into IL, which assumes that the optimal expert policy is guided by an unknown reward function. Therefore, to obtain the optimal policy, we should recover the reward function at first. Recently, IRL methods have been widely used in planning and control tasks. Ng *et al*. [11] first use linear programming to address the IRL problem in Gridworld and Mountaincar environments. Abbeel *et al*. [14] propose the Apprenticeship learning, in which the reward function is expressed as a linear combination of state features. Ziebart *et al*. successfully integrate the maximum entropy principle with IRL [15], by modelling IRL problem as a probability model. The maximum entropy IRL method can avoid learned policy having ambiguity in choosing a distribution over decisions. On the basis of maximum entropy IRL, Boularia *et al*. [16] propose the relative entropy IRL, which can build a probability model in the unknown environment. Combined the strong feature extraction ability of deep neural network, deep IRL methods has been applied in the problem with high dimensional inputs such as Atari video games [17], driving trajectory prediction [18] and so

This work is supported by the National Natural Science Foundation of China (NSFC) under Grants No. 61803371, Beijing Science and Technology Plan under Grants Z191100007419002, and Beijing University High-Level Talent Cross-Training Project (Practical Training Plan).

on. Generally speaking, IRL methods have a stronger generalization ability compared with BC. However, it is still expensive for training, because the RL process is included in the IRL to approach the optimal policy.

Recently, some imitation learning methods have been proposed to expand and improve the above-mentioned methods. Finn *et al.* [19] propose the guided cost learning, as a new type of IRL method, it performs far better than relative entropy IRL in real-world robotic control task. Ho *et al.* [20] combine the concept of Generative adversarial nets (GAN) [21] with imitation learning, and present Generative Adversarial Imitation Learning (GAIL), which can learn a policy directly from expert demonstrations without recovering a corresponding reward function. Li *et al.* [22] make an improvement in GAIL and present information maximizing GAIL (InfoGAIL), which is similar to information maximizing GAN (InfoGAN) [23]. They introduce mutual information to measure the relationship between trajectories and latent variables, which can learn different driving styles in TORCS simulator.

It should be mentioned that BC and DAgger are the basic supervised IL methods based on the expert data. And InfoGAIL can be considered as the state-of-the-art method for IRL. The comparison between BC, DAgger and InfoGAIL in autonomous driving should be given in the racing and urban driving scenarios, which can make us understand IL better for autonomous driving. This paper is organized as follows. In Section II, we introduce the preliminaries and three typical imitation learning algorithms: BC, DAgger and InfoGAIL. In Section III, these three algorithms are used to complete the simple lane keeping task in TORCS and CARLA simulators, respectively. We also recover the ability of classification of InfoGAIL in CARLA. Furthermore, we compare their performances and explain the pros and cons of each algorithm. Finally, we give the conclusion in Section IV.

II. Related work

A. Preliminaries

In the problem of IL, we use the tuple $(S,\ A,\ P,\ \rho_0, X_E)$ to define an infinite-horizon and discounted Markov decision process, where S represents the state space, A represents the action space, $P: SxAxS \rightarrow \mathbb{R}$ denotes the transition probability distribution, $\rho_0: S \rightarrow \mathbb{R}$ is the initial state distribution, and $X_E = \{\xi_1, \xi_2, \dots, \xi_3\}$ is the expert demonstrations, which consists of a series of expert trajectories, where each ξ_i includes a sequences of state-action pairs: $\xi_i = \{(s_0, a_0), (s_1, a_1), \dots, (s_n, a_n)\}$. Additionally, we use π_E to represent expert policy, although in many cases we only have expert demonstrations X_E to represent it indirectly.

B. Behavior cloning

BC is the most basic IL method, which aims to learn a mapping from state to action. In BC, a policy is trained only based on the state-action pairs encountered in expert demonstrations, which means it can be solved by using any standard supervised learning algorithm.

Note that the goal of IL is to learn a policy which can mimic expert behaviors. In BC, for each state *s* in the expert demonstrations, we use *C(s,a)* to denote the true expected loss of performing action *a* instead of expert action at state *s.* And the true expected loss of performing policy π at state *s* can be denoted as $C_\pi(s) = E_{a\sim\pi(s)}[C(s,a)]$. For some particular tasks, the true costs $C_\pi(s)$ is usually unknown, so we use surrogate loss function $l(s,\pi)$ instead of $C_\pi(s)$ to denote the difference between policy π and expert policy at state *s.* Our goal is to find a policy π_{sup}, which has the minimum surrogate loss for the distribution of states from expert demonstrations. The objective function is as follows:

$$\pi_{sup} = arg \min_{\pi\in\Pi} E_{s\sim X_E}[l(s,\pi)] \qquad (1)$$

Standard behavior cloning is lack of generalization ability in unknown scenarios and is insufficient for handling complex driving scenarios. In addition, the inevitable compounding errors also increase the gap between learned policy and expert policy. Early work on ALVINN by Pomerleau *et al.* use the BC method to train a shallow neural network to follow the road [24]. Recently, researchers at NVIDIA [25] introduce how to train a deep convolution neural network for the steering control based on the image inputs, where a large number of expert demonstrations (about 72 hours of human driving data) is used to train this model.

C. Dataset Aggregation

To alleviate the existing problems of lacking of generalization ability and compounding errors, some improvements are applied to BC. DAgger is one of the improved algorithms [13, 26, 27]. It is a kind of teaching-apprentice interactive learning method, which uses expert revision to correct actions generated by learned policy. In the first iteration, DAgger uses standard BC algorithm to train a policy π_{sup}. In the next series of iterations, DAgger uses learned policy π_{sup} to collect more trajectories, where each state-action pair is composed of state encountered by π_{sup} and action labelled by expert policy. Then those trajectories are added to the demonstrations X_E. Using supervised learning algorithm, a better policy π_{sup}can be obtained for the next iteration. It means that even if the agent chooses a bad action in unknown scenarios, it can still learn how to fix it with the help of expert policy. The appearance of DAgger improves the robustness and generalization ability of original behaviour cloning.

D. InfoGAIL

Note that GAIL adopts a different approach on imitation learning by integrating GAN with IRL. It models the learning process of the agent as a game process, and can learn a policy without directly estimating the corresponding reward function. Furthermore, InfoGAIL adds the concepts of latent codes and mutual information based on GAIL, thus it can distinguish the trajectories from different experts in an unsupervised way. It also introduces some techniques such as WGAN and replay buffers to stabilize the training process. In the following, we will introduce the related methods with InfoGAIL.

1) GAN

GAN [21] is a kind of generative model based on neural networks. In general, the purpose of generative models is to build a function model which can capture the distribution of input data. The core idea of GAN can be traced back to the Nash equilibrium principle of the game theory. Specifically, the training procedure of GAN can be considered as a minimax two-player game, where a generative model *G* and a discriminative model *D* compete with each other until converging to the Nash equilibrium. For example, in the image generation task, the inputs of GAN are a series of image data, where the generative model *G* aims to build a function model which learns to generate image samples by taking a set of random noise *z* as inputs. And the discriminative model *D* can also be built as a function model which learns to determine whether a sample is from the generative model distribution or the images data distribution. The objective function of GAN is as follows.

$$\min_{G}\max_{D} V(D,G) = E_{x\sim p_{data}(x)}\left[log\left(D(x)\right)\right] + E_{z\sim p_z(z)}\left[log\left(1-D\left(G(z)\right)\right)\right] \quad (2)$$

where *x* represents samples from the data distribution, *G(z)* represents samples from the generative model distribution. In the training procedure, discriminative model *D* tries to discriminate the source of samples, and generative model *G* tries to maximize the probability of *D* making a mistake. Competition between *G* and *D* drives both models to improve their performances. Finally, *G* can capture the data distribution and generate image samples which are very similar to the original image data. Recently, GAN has received a lot of attentions and been applied in high fidelity image synthesis [28] , text to photo-realistic image synthesis [29] and so on.

2) GAIL

The generative model of GAN can generate samples which are similar to original data just by taking a set of random noise *z* as inputs. Converting it into the field of imitation learning, the environment state *s* can be used to replace random noise *z* as inputs of the generative model *G*, and the action *a* can be considered as the generated outputs. Based on this idea, GAIL is proposed [20]. The framework of GAIL consists of two parts: generative model *G* and discriminative model *D*, which is the same with GAN. The inputs of GAIL are expert demonstrations X_E, which are consisted by a series of expert trajectories. In GAIL, the generative model *G* can be considered as a policy model, which can calculate the corresponding action *a* according to the environment state *s*. In the training process of GAIL, generative model *G* keeps interacting with the environment and generates trajectories, while discriminative model *D* tries to distinguish whether the sampled state-action pairs are from expert demonstrations or from the generated trajectories. The competition process between *G* and *D* in GAIL is the same as in GAN. As the competition goes on, eventually the generated state-action distribution will be matched with the state-action distribution in expert demonstrations.

Considering the generative model *G* with weights θ and the discriminative model *D* with weights ω, then the objective function of GAIL can be described as follows.

$$\max_{\omega}\min_{\theta} V(\theta,\omega) = E_{(s,a)\sim X_E}\left[log\left(D_\omega(s,a)\right)\right] + E_{(s,a)\sim X_\theta}\left[log\left(1-D_\omega(s,a)\right)\right] - \lambda H(\pi) \quad (3)$$

where X_θ denotes trajectories generated by generative model *G*, $H(\pi)$ denotes causal entropy of the policy π_θ (*G*). The performance of GAIL in OpenAI gym is comparable with other imitation learning methods, and it can also learn a multi-modal stochastic policy in gym [30] by integrating with the idea of InfoGAN.

3) InfoGAIL

Combined GAIL and InfoGAN, Li *et al*. [22] propose the InfoGAIL algorithm, which can distinguish trajectories from different experts. Note that the standard GAN lacks the ability to distinguish the differences when the original data have unlabelled, latent classifications. The original GAIL algorithm has the same flaw. For example, if an autonomous car is trained using GAIL, and the expert demonstrations are a mixture of trajectories generated by different experts with different preferences, then the learned policy denoted by generative model *G* may only learns from one expert in the worst case. Actually, we cannot say that GAIL algorithm fails, because it does capture a part of expert distribution accurately.

For InfoGAN and InfoGAIL, the latent code *c* is used as a part of the inputs of the generative model *G*, which represents the latent classification of generated samples/trajectories. In other words, the outcome of *G* is not only determined by the random noise *z* or environment state *s*, but also determined by the latent code *c* as well. In order to avoid latent code *c* losing constraint to the generated samples, InfoGAN introduces the concept of mutual information, and uses regularization term of mutual information to improve the original objective function. Mutual information *I(X|Y) = I(Y|X)* represents the relevance between variables *X* and *Y*, which can be denoted as the difference between two entropy terms:

$$I(X|Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) = I(Y|X) \quad (4)$$

Hence the objective function of InfoGAN is as follows.

$$\min_{G}\max_{D} V_I(D,G) = V(D,G) - \lambda I\left(c\middle|G(z,c)\right) \quad (5)$$

where *V(D,G)* denotes the objective function of original GAN, and $\lambda > 0$ is a hyperparameter.

Similarly, it is required to subtract the above regularization term of mutual information in the objective function of InfoGAIL.

$$\max_{\omega}\min_{\theta} V(\theta,\omega) = V(\theta,\omega) - \lambda I(c|\xi) \quad (6)$$

where $V(\theta,\omega)$ denotes the objective function of original GAIL, and ξ denotes the generated trajectorie.

However, it is hard to maximize the regularization term $I(c|\xi)$ as it requires the posterior $P(c|\xi)$. Hence, a variational lower bound $L_I(\pi,Q)$ of the mutual information $I(c|\xi)^3$ is introduced as follows.

$$L_I(\pi,Q) = E_{c\sim p(c),a\sim\pi(\cdot|s,c)}[logQ(c|\xi)] + H(c) \le I(\pi,c) \quad (7)$$

where $Q(c|\xi)$ is an approximation of the true posterior $P(c|\xi)$, which can be easily approximated with Monte Carlo simulation. *H(c)* denotes the casual entropy of latent code *c*, which is usually treated as a constant. Eventually, the objective function of InfoGAIL can be described as follows.

$$\max_{\omega} \min_{\theta} V(\theta,\omega) = V(\theta,\omega) - \lambda L_I(\pi, Q) \quad (8)$$

It should be mentioned that some specific optimization techniques such as WGAN [29], replay buffers and reward augmentation are applied to InfoGAIL in experiments to stabilize the training process. The detail can be found in [22]. And the experiment results prove that InfoGAIL can distinguish different expert trajectories in an unsupervised way and has a longer average driving distance compared with BC and GAIL in TORCS simulator.

E. A macroscopic comparison of three imitation learning methods

As the most basic imitation learning method, BC is easy to deploy and does not require a predefined reward function. But theoretically the policy learned through BC can only perform well over the states which are from or similar with the expert demonstrations. In addition, due to compounding errors, every little deviation from expert grows quadratically in time, which eventually causes generated trajectories have a poor performance compared with expert demonstrations.

Due to the dataset aggregation of expert policy, DAgger is able to correct wrong actions of the learned policy, which enhances the generalization ability of the model. The experiment shows [13], in a 3D racing game, Super Tux Kart, DAgger can gain a significant improvement with only a few iterations of training. But the disadvantage of DAgger is quiet obvious which is the label of every new state visited by the agent should be given by the expert. It means that a large amount of labour and time are required if we do it manually.

Compared with behavior cloning and DAgger, the framework of InfoGAIL is much more complex. Expect from a generative model *G*, which can be seen as a policy network, InfoGAIL has a discriminative model *D* and a posterior approximation model *Q* additionally. Meanwhile, unlike BC and DAgger, the updating of *G* depends on the outcomes of *D* and *Q* rather than the difference between generated actions and expert actions. Furthermore, InfoGAIL uses the trust region policy optimization (TRPO) [31] method to update the generative model *G*, in which the value of reward signals not only relates with the outcome of *D* and *Q*, but also relates with the practical performance of *G*. From this perspective, InfoGAIL can be seen as a hybrid between imitation and reinforcement learning.

In TORCS simulator, [22] uses the average driving distance to compare the performances between InfoGAIL, GAIL, BC and human expert. The result shows that InfoGAIL with several improved optimization techniques performs the best, even surpasses human expert. Besides, InfoGAIL can be used to learn different driving styles from multi-modal expert data, which is very meaningful for the imitation learning with large supervised data from different drives.

In Table I, we give the comparsion of the three methods based on the policy update mode, extra data, interaction with environment, and distinguish latent classifications. InfoGAIL can distinguish the latent driving style, and many optimization tricks are used to solve the specific optimization techniques, which sometimes makes it difficult to deploy. In the following, we try to compare and analyze the performances of these methods in the form of experiments.

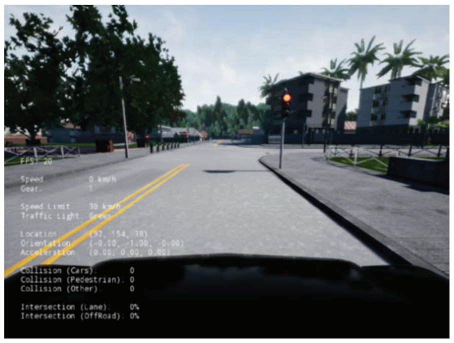
TABLE I. PROPERTIES OF THREE IMITATION LEARNING METHODS

Method	**Policy update mode**	**Using extra data**	**Interaction with environment**	**Distinguish latent classifications**
BC	Supervised learning	No	No	No
DAgger	Supervised learning	Yes	Yes	No
InfoGAIL	Reinforcement learning	No	Yes	Yes

III. EXPERIMENTS

In order to test and compare the performances of three imitation learning algorithms for the specific lane keeping task, and verify the validity of InfoGAIL in a multi-mode driving data. We first apply them in TORCS racing simulator and then move to CARLA simulator. As shown in Fig. 1 and Fig. 2. Compared with TORCS racing simulator, CARLA simulator has a more realistic environment due to the use of Unreal Engine 3, as well as a complete rule for urban driving, which all make it become a more challenging task.

Fig. 1. TORCS simulator



```
FPS: 20

Speed          0 km/h
Gear:          1

Speed Limit    30 km/h
Traffic Light  Green

Location       (92, 154, 38)
Orientation    (-0.00, -1.00, -0.00)
Acceleration   (0.00, 0.00, 0.00)

Collision (Cars):        0
Collision (Pedestrian):  0
Collision (Other):       0

Intersection (Lane):     0%
Intersection (OffRoad):  0%
```

Fig. 2. CARLA simulator

In our experiments, we not only take raw visual images as state inputs, but also take a series of auxiliary information to assist decision-making. The auxiliary information is different in TORCS and CARLA experiments. In TORCS, the vehicle dynamics information is considered as the auxiliary information just like the original InfoGAIL framework does. In CARLA, we take some dashboard data as auxiliary information to better simulate the real-world driving scenarios. The detail settings will be discussed separately later. The policy model receives raw visual images and auxiliary information as its state inputs, and outputs a three-dimensional continuous action including steering, acceleration and braking value. In particular, for InfoGAIL, the discrete latent code is set as an one-hot vector with two possible states. The neural network models are designed as follows.

A. Network structure

The network structures are mainly built as same as [22] in order to get a similar good performance. And we only make a few changes about the state inputs in the CARLA experiment.

For the policy network (Generative network *G*), raw image inputs are firstly converted into high-level information by a pre-trained Deep Residual Network [33], and as the actual inputs of policy network, these exploited visual features pass through two convolutional layers first, then combine with the auxiliary information vector and the latent code *c* (in the case of InfoGAIL), eventually turn into action vectors in output layer.

The inputs of discriminative network *D* are: raw image inputs, auxiliary information and corresponding actions. Firstly, raw input images pass through three convolutional images, then combine with auxiliary information vector and action vector. At the final output layer, *D* marks the input state-action pairs with a score for the WGAN training object. The posterior approximation network *Q* shares the same inputs and network structure with *D*, expect for the final output layer with a softmax function over the discrete latent codes.

B. TORCS experiment

In TORCS experiment, the setting of all parameters are the same as original InfoGAIL, in particular, the setting of auxiliary information at time t consists of the following: 1) velocity of the car, which is denoted by a three dimensional vector, 2) actions at time t-1 and t-2, which are both denoted by three dimensional vectors, 3) damage of the car, which is denoted as a real value.

Our goal is to compare the performance of three imitation learning algorithms in the lane keeping control task. In TORCS experiment, we simply use driving distance to represent the performance of each algorithm. At first, we use a set of well-trained weights to initialize networks, and then compare the driving distance between BC and InfoGAIL. Note that the expert demonstrations consist of 32 trajectories provided by [34], which are all taken in one specific racing track with fixed start point.

TABLE II. AVERAGE DRIVING DISTANCES WITH FIXED DATASET

Method	**Avg. driving distance**
BC	522.737
InfoGAIL(code 0)	529.107
InfoGAIL(code 1)	**537.382**

We test the pre-trained networks provided by [34]. As shown in Table Ⅱ, InfoGAIL does show some advantages than BC in average driving distance. Also note that InfoGAIL can distinguish and learn different driving styles among expert trajectories clearly. We also notice that both BC and InfoGAIL have the issue of driving off the road occasionally.

Furthermore, to bring DAgger algorithm into comparison, some improvements are given on the original TORCS experiment. In this experiment, we first remove the restriction of trajectory step size, and use provided 32 trajectories to train a basic BC network. Then, we follow the idea of DAgger. In each iteration, we gather 30 new trajectories, whose expert actions are labelled by a customized bot, which intends to keep the car driving in the middle of the road. These expert-labelled trajectories will join in the old expert demonstrations and be used to train a new policy network. Meanwhile, in each iteration of DAgger, the mixed expert demonstrations will also be provided to InfoGAIL for policy updating.

TABLE III. AVERAGE DRIVING DISTANCES WITH DIFFERENT DATASETS

Method	**Avg. driving distance**	**The quantity of expert trajectories**
BC	360.31	32
DAgger	402.45	62
	419.74	92
InfoGAIL	333.6	32
	319.45	62
	210.43	92

As shown in Table □, DAgger algorithm performs better as the quantity of expert trajectories increases. On the other hand, InfoGAIL does not increase its performance with more expert trajectories, and even decrease. It is because InfoGAIL takes TRPO method to update its policy, and the reward signal is determined largely by the output of discriminative model *D* and the output of posterior approximation model *Q*. In order to keep the policy updating in the right direction, *D* should be trained neither too good nor too bad, which is hard to balance in practical applications. Also notice that the new added trajectories do not have a clear modal, which may cause InfoGAIL to treat occasional brakes in the trajectories as a driving style to learn, hence the average driving distance reduces.

C. CARLA experiment

In CARLA experiment, we choose a different series of auxiliary information, trying to make it closer to real-world driving scenarios. The auxiliary information at time t consists of the following: 1) forward velocity of the car, which is denoted as a real value, 2) location of the car, denoted as plane coordinates, 3) drive off-road indicating what proportion of the car drives off the road, which is a real value in the range of [0-1], 4) drive other-lane, which is a real value indicating what proportion of the car drives on the wrong side of the road, 5) collision, which is a real value, 6) actions at time t-1, which is a three dimensional vector. Therefore, the auxiliary information is denoted as a nine dimensional vector.

In CARLA, we design two experiments to demonstrate the performance of algorithms. The first experiment is to verify the separation of latent codes that represent different behaviors. We select a scenario of driving in a straight road, expect agent to drive in left lane or right lane by imitating expert demonstration. (We use code 0 and 1 to represent drive in left lane and right lane respectively) In this experiment, expert data are collected equally from both modes, which refers to drive in left lane or right lane. The expert demonstration contains 30 trajectories in total, and the length of each trajectory is fixed at 150 steps.

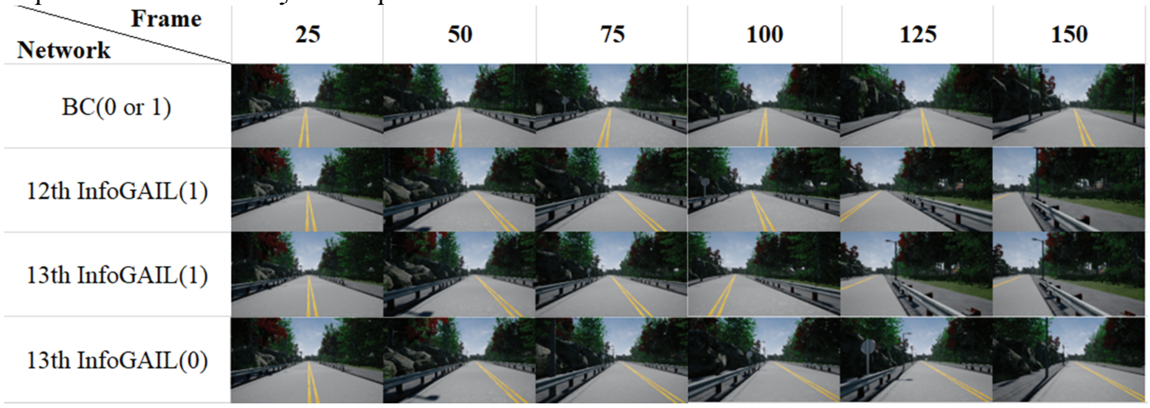


Fig. 3. Visual inputs for policy network

The result of the first experiment is shown in Fig. 3. The frame number refers to current step since the beginning of simulation. The first row is the result of pre-trained BC network, the second row is the result of the 12th iteration of InfoGAIL network with code 1, the third row is the result of the 13th iteration of InfoGAIL network with code 1, and the last row is the result of the 13th iteration of InfoGAIL network with code 0. It should be noticed that, as training proceeds, the agent tends to drive in left lane with latent code 0, and drive in the right lane with latent code 1. According to the experiment result, when we compare the first row with the third row and the last row, it is obvious that InfoGAIL can capture and learn the latent classifications of the expert demonstrations, while BC cannot distinguish the differences in the expert demonstrations. As the third row is compared with the last row, we can also draw a conclusion that after rounds of iteration, InfoGAIL successfully use the separation of latent codes to represent different behavior. Additionally, if we focus on the performance of code 1 only, and compare the second row with the third row, it is clear that the 13th iteration of InfoGAIL can perform the actions of driving in right lane in earlier steps, which refers to a better classification of the expert data. So we can also conclude that the separation of latent codes becomes better and better as the algorithm proceeds.

D. *Comparison on Lane Keeping and Free Driving task*

The second experiment is to compare the availability of different IL methods in lane-keeping and free driving task. The setting of this experiment is more challenging than previous experiments. Firstly, the driving rule is more strict. We confine the drivable area to the right side of the road. And any actions which make the car drive off the road, or drive to the left side of the road are forbidden and will be punished. Secondly, it is hard for the car to take the right steering action at road conjunctions, because lane markings are missing at road conjunctions, which are the important feature for lane keeping task. Also, the driving scenarios are more diversified. We set 6 spots with unique surroundings as the start points of the expert trajectories. In test period we add 4 more start points to verify generation of agent. Furthermore, the distribution of expert demonstrations is more scattered compared with previous experiments, and we only take 3 trajectories for each start point same as expert demonstrations.

As mentioned before, in the second experiment, we have 18 expert trajectories as initial expert demonstrations. At the beginning of the experiment, we use BC algorithm to train an initial policy network to speed up the training process. Then we train the DAgger algorithm. In each iteration of DAgger, we collect 20 new trajectories, and label each state with expert action. Note that the other training processes follow the standard route of DAgger. In the end, we successfully implement 4 iterations of DAgger, and we will show the learning results of the first two policies: DAgger_1 (using 38 trajectories) and DAgger_2 (using 58 trajectories).

To fairly compare the performance of InfoGAIL and DAgger, we also use the demonstrations of DAgger_1 and DAgger_2 to train InfoGAIL. Additionally, a weighted score instead of driving distance is used to reflect the performance of a generated trajectory. The score is added 1.5 at every time step. If the car crashes into other objects, the score will be subtracted by 300. For the situation that the car drives on the sidewalk or drives on the wrong side of the road, the score will be subtracted in terms of how many proportions of the car not in the drivable area. The score function of each time step is shown below, note that c is a boolean refers to whether the car crashes into other objects, p_1 and p_2 refer to the percentage of chassis area which in the sidewalk or in the wrong side of the road respectively. The maximum length of each trajectory is fixed at 440 steps, hence the upper limit score of one trajectory can get is 660.

$$score = 1.5 - c * 300 - 10 * (p1 + p2) \tag{9}$$

TABLE IV. AVERAGE ROLLOUT SCORES

Method	Avg. rollout scores	The quantity of expert trajectories
BC	237.4	18
DAgger	**465.82**	38
	427.52	58
InfoGAIL	308.13	18
	310.84	38
	319.98	58

Table . shows the average rollout scores of different policies with different amount of expert demonstrations. Comparing DAgger with BC, it is clear that only by adding a few expert revised trajectories, DAgger can recover from errors rapidly and get a massive improvement. However, the performance of DAgger is strong related with the collected data, hence DAgger is hard to make further improvements in the latter iterations if no valid data is added. Comparing the outcomes of InfoGAIL and BC, we notice that InfoGAIL does make some improvement, but not as obvious as DAgger. We also notice that InfoGAIL gains a modest improvement as extra expert demonstrations are used.

E. Disscussion

Through the above experiments, it is clear that expert demonstrations play a very important role in the field of supervised IL. Take DAgger algorithm as an example, which can gain a significant improvement just by adding a small amount of high quality (unknown state and expert action pairs) trajectories to demonstrations. InfoGAIL also increases its performance as more trajectories added into expert demonstrations in CARLA experiment, but not as much as DAgger. We speculate that the lack of clear classifications of the expert demonstrations maybe is the reason why InfoGAIL improves little in simple lane keeping task, because when there is only one mode in the distribution of expert demonstration, supervised IL methods have their own advantages compared with InfoGAIL. Because there are more hyper-parameters to be adjusted for InfoGAIL which will increase the difficult to obtain a good performance. Also, as a hybrid algorithm between supervised learning and reinforcement learning, InfoGAIL may not be as sensitive as BC and DAgger towards expert data, and the diversified scenarios and scattered demonstrations may weak the performance of generative model G as well. Furthermore, we succeed in recovering InfoGAIL's unique ability of classification in the urban scenarios of CARLA.

In general, BC algorithm can be rapidly deployed to certain tasks. But its performance is highly related with the distribution of the expert demonstrations. If we can get expert revised trajectories during the training process, DAgger algorithm is a relatively good choice, because its improvement is stable with the addition of new trajectories in most cases. InfoGAIL on the other hand is not easy to deployed when encountering a new task, because it needs to interact with environment and alternately train three networks at least, the appropriate structure and hyperparameters are also hard to set for complex tasks. But InfoGAIL has an unique property of address multi-modal data in expert demonstrations, and the combination with RL methods also makes it have a strong generalization ability for different datasets.

IV. CONCLUSION

In this paper, we compare the performances of three typical imitation learning algorithms: Behavior cloning, DAgger and InfoGAIL for the specific task of lane keeping. The above imitation learning algorithms are first tested in racing scenes and then be deployed in a more challenging urban scenario. The experiment results show that the supervised IL such as BC and DAgger rely heavily on the quantity and quality of expert demonstrations. InfoGAIL algorithm does not show a significant advantage in simple lane keeping task compares with BC and DAgger, but it can accurately separate the latent classifications of expert demonstration and learn different behavior patterns, which is very difficult for supervised IL methods. In our opinion, how to stabilize the training process in complex tasks is one of the key works for InfoGAIL.

ACKNOWLEDGMENT

The authors thank to Dr. Yunzhu Li, one author of InfoGAIL, for the valuable discussion and suggestions.

REFERENCES

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," Nature, vol. 529, pp. 484-489, 2016.

[2] Q. C. Zhang, D. B. Zhao, and D. Wang, "Event-based robust control for uncertain nonlinear systems using adaptive dynamic programming," IEEE Transactions on Neural Networks and Learning Systems, vol. 29, pp. 37-50, 2018.

[3] Q. C. Zhang, D. B. Zhao, and Y. H. Zhu, "Event-triggered H∞ control for continuous-time nonlinear system via concurrent learning," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 47, pp. 1071-1081, 2017.

[4] D. Li, D. B. Zhao, Q. C. Zhang, and Y. R. Chen, "Reinforcement Learning and Deep Learning Based Lateral Control for Autonomous Driving [Application Notes]," IEEE Computational Intelligence Magazine, vol.14, pp. 83-98, 2019.

[5] K. Shao, D. B. Zhao, N. Li, and Y. H. Zhu. "Learning battles in ViZDoom via deep reinforcement learning," In IEEE Conference on Computational Intelligence and Games, pp. 1-4, 2018.

[6] D. Wang, D Liu, Q. C. Zhang and D. B. Zhao, "Data-based adaptive critic designs for nonlinear robust optimal control with uncertain dynamics," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 46, pp. 1544–1555, 2016.

[7] D. Wang and D. Liu, "Learning and guaranteed cost control with event-based adaptive critic implementation," IEEE Transactions on Neural Networks and Learning Systems, vol. 29, pp. 6004–6014, 2018.

[8] Y. L. Yang, Z. S. Guo, H. Y. Xiong, D. W. Ding, Y. X. Yin and D. C. Wunsch, "Data-Driven Robust Control of Discrete-Time Uncertain Linear Systems via Off-Policy Reinforcement Learning," IEEE Transactions on Neural Networks and Learning Systems, 2019.

[9] Y. H. Zhu, and D. B. Zhao. "Driving Control with Deep and Reinforcement Learning in The Open Racing Car Simulator," In International Conference on Neural Information Processing, pp. 326-334, 2018.

[10] D. B. Zhao, Z. P. Xia, and Q. C. Zhang, "Model-free optimal control based intelligent cruise control with hardware-in-the-loop demonstration [research frontier]," IEEE Computational Intelligence Magazine, vol. 12, pp. 56-69, 2017.

[11] D. A. Pomerleau, “Efficient training of artificial neural networks for autonomous navigation,” Neural Computation, vol. 3, pp. 88–97, 1991.

[12] A. Ng, and S. Russell, “Algorithms for inverse reinforcement learning,” Proceedings of the Seventeenth International Conference on Machine Learning, pp. 663-670, 2000.

[13] S. Ross, G. Gordon, and J. A., Bagnell, “A reduction of imitation learning and structured prediction to no-regret online learning,” Proceedings of the Fourteenth International Conference on Artificial Yntelligence and Statistics, pp. 627-635, 2011.

[14] P. Abbeel, and A. Ng, “Apprenticeship learning via inverse reinforcement learning,” Proceedings of the Twenty-First International Conference on Machine learning, p. 1, 2004.

[15] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, “Maximum entropy inverse reinforcement learning,” Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, vol. 8, pp. 1433–1438, 2008.

[16] A. Boularias, J. Kober , and J. Peters, “Relative entropy inverse reinforcement learning,” Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, pp. 182-189, 2011.

[17] M. Bogdanovic, D. Markovikj, M. Denil, and N. Freitas, “Deep apprenticeship learning for playing video games,” Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence. 2015.

[18] Y. Zhang, W. Wang, R. Bonatti, D. Maturana, and S. Scherer, “Integrating kinematics and environment context into deep inverse reinforcement learning for predicting off-road vehicle trajectories,” arXiv preprint arXiv:1810.07225, 2018.

[19] C. Finn, S. Levine, and P. Abbeel, “Guided cost learning: Deep inverse optimal control via policy optimization,” Proceedings of the Thirty-Third International Conference on Machine Learning, vol. 48, pp. 49-58, 2016.

[20] J. Ho and S. Ermon, “Generative adversarial imitation learning,” In Advances in Neural Information Processing Systems, pp. 4565–4573, 2016.

[21] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, “Generative adversarial nets,” In Advances in Neural Information Processing Systems, pp. 2672–2680, 2014.

[22] Y. Li, J. Song, and S. Ermon, “Infogail: Interpretable imitation learning from visual demonstrations,” In Advances in Neural Information Processing Systems, pp. 3812-3822, 2017

[23] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, “Infogan: Interpretable representation learning by information maximizing generative adversarial nets,” In Advances in Neural Information Processing Systems, pp. 2172–2180, 2016.

[24] D. A. Pomerleau, “Alvinn: An autonomous land vehicle in a neural network,” In Advances in Neural Information Processing Systems, pp. 305-313, 1989.

[25] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackl, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, “End to end learning for self-driving cars,” arXiv preprint arXiv:1604.07316 , 2016.

[26] H. Daumé, J. Langford, and D. Marcu, “Search-based structured prediction,” Machine learning, vol. 75, pp. 297-325, 2009.

[27] S. Chernova, and M. Veloso, “Interactive policy learning through confidence-based autonomy,” Journal of Artificial Intelligence Research, vol. 34, pp. 1-25, 2009.

[28] A. Brock, J. Donahue, and K. Simonyan, “Large scale gan training for high fidelity natural image synthesis,” arXiv preprint arXiv:1809.11096, 2018.

[29] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas, “Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks,” Proceedings of the IEEE International Conference on Computer Vision, pp. 5907-5915, 2017.

[30] K. Hausman, Y. Chebotar, S. Schaal, G. Sukhatme, and J. J. Lim, “Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets,” In Advances in Neural Information Processing Systems, pp. 1235-1245, 2017.

[31] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” Proceedings of the Thirty-Second International Conference on Machine learning, pp. 1889–1897, 2015.

[32] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein gan,” arXiv preprint arXiv:1701.07875, 2017.

[33] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition.” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770-778, 2016.

[34] Dataset which contains 6400 state-action pairs provided by experts: https://drive.google.com/open?id=0B1mByo_qyT3PU0JsRm9fOHR1Sj A