

Article

# Modeling Car-Following Behaviors and Driving Styles with Generative Adversarial Imitation Learning

Yang Zhou <sup>1,2</sup>, Rui Fu <sup>1,\*</sup>, Chang Wang <sup>1</sup> and Ruibin Zhang <sup>1</sup>

<sup>1</sup> School of Automobile, Chang'an University, Xi'an 710064, China; 2016022005@chd.edu.cn (Y.Z.); wangchang@chd.edu.cn (C.W.); 2016022016@chd.edu.cn (R.Z.)

<sup>2</sup> School of Vehicle Engineering, Xi'an Aeronautical University, Xi'an 710077, China

\* Correspondence: furui@chd.edu.cn

Received: 6 August 2020; Accepted: 2 September 2020; Published: 4 September 2020



**Abstract:** Building a human-like car-following model that can accurately simulate drivers' car-following behaviors is helpful to the development of driving assistance systems and autonomous driving. Recent studies have shown the advantages of applying reinforcement learning methods in car-following modeling. However, a problem has remained where it is difficult to manually determine the reward function. This paper proposes a novel car-following model based on generative adversarial imitation learning. The proposed model can learn the strategy from drivers' demonstrations without specifying the reward. Gated recurrent units was incorporated in the actor-critic network to enable the model to use historical information. Drivers' car-following data collected by a test vehicle equipped with a millimeter-wave radar and controller area network acquisition card was used. The participants were divided into two driving styles by K-means with time-headway and time-headway when braking used as input features. Adopting five-fold cross-validation for model evaluation, the results show that the proposed model can reproduce drivers' car-following trajectories and driving styles more accurately than the intelligent driver model and the recurrent neural network-based model, with the lowest average spacing error (19.40%) and speed validation error (5.57%), as well as the lowest Kullback-Leibler divergences of the two indicators used for driving style clustering.

**Keywords:** human-like car-following model; driving styles; generative adversarial imitation learning; gated recurrent units

---

## 1. Introduction

With the process of urbanization and the rapid growth in the number of vehicles, car following has become the most common driving behavior in daily driving. It is of great significance to build a model that can accurately simulate drivers' car-following behaviors. Car-following models have been widely applied in microscopic traffic simulation, the tests of driving assistant systems, and other fields [1–3]. In the future of autonomous driving, designing an autonomous vehicle that can achieve human-like driving can also improve the riding comfort and trust of passengers [4,5].

Research on car-following modeling has lasted for more than half a century. The modeling methods can be divided into two types: theoretical-driven methods and data-driven methods [6]. The former generally puts forward various hypotheses on drivers' reactions based on the observation of actual car-following behaviors and establishes mathematical relationships to express drivers' car-following strategies. In the expression of theoretical-driven models, there are some parameters representing the preferences of different drivers. Since the theoretical-driven models contain rather few parameters, their expression ability is limited, thus the accuracy and generalization ability cannot

be ensured [7]. Therefore, in recent years, a large number of studies have used data-driven methods to build car-following models [7–10].

The data-driven methods can directly use data to learn drivers' control strategies and have fewer assumptions on driving behavior. Data-driven models such as neural networks have stronger expression ability and can be easily updated. Studies have shown that the data-driven car-following models have better generalization ability than the theoretical-driven models [7,11]. In the related research of data-driven models, directly applying learning methods to learn the map from states to actions is referred to as behavior cloning (BC) [12]. Although the BC models have been proven to be effective in some studies, they may suffer from the problem of cascading errors, which is very common in sequential decision-making problems [13]. The error existing in the single time-step prediction will accumulate and gradually increase in the sequential decision-making process, which may cause the model to reach unseen states, making the model have even worse predictions. To avoid this problem, some researchers have begun to use reinforcement learning (RL) methods [11,14].

RL assumes that human behavior maximizes the cumulative long-term rewards [15]. When the reward is determined, the strategy learned by using RL algorithms can have the ability of long-term planning and can avoid cascading errors [16]. However, the designing of the reward function in RL is very hard, manually designing it is time-consuming, and an inappropriate reward function may even lead to unexpected behaviors [17,18]. To overcome this issue, inverse reinforcement learning (IRL) has been proposed [19]. Instead of modeling the strategy directly, the IRL method first learns the reward function and then obtains the strategy by using RL.

The traditional IRL method often uses a linear representation of reward function, which may not reflect drivers' nonlinear intrinsic preferences [20]. Besides, it requires a lot of computation power as it needs to solve the RL subprocess during training [21]. In this study, a recently proposed algorithm called generative adversarial imitation learning (GAIL) was applied to model drivers' car-following strategies. The main contributions of this study are as follows: (1) a novel way to model drivers' car-following behaviors was proposed. The proposed model uses a nonlinear function that uses neural networks to automatically learn drivers' rewards and strategies, and the training of the model does not need to solve the RL subprocess, which can save a lot of computation power. Besides, a kind of recurrent neural network (RNN) called gated recurrent units (GRU) is used in the proposed model to fit drivers' car-following policy, which is able to take advantage of the historical information for time-sequence prediction. (2) The car-following data of drivers collected in real-world situations is used to train and verify the proposed model and two other representative car-following models. Different methods were compared in terms of replicating drivers' car-following trajectories and driving styles.

The rest of the paper is organized as follows: Section 2 briefly reviews the related work and backgrounds of this study. Section 3 presents the architecture of the proposed model and the algorithm. Section 4 describes the experiments and data used in this study. Section 5 presents the details of model training and validation. Section 6 presents the results and comparisons of different methods. The final section presents the discussion.

## 2. Related Work

### 2.1. Theoretical-Driven Car-Following Models

The development of theoretical-driven car-following models can be dated back to the 1960s. Pipes [22] and Chandler, et al. [23] proposed the earlier General Motors models. Since then, several different models have been proposed, such as the Gipps model [24], the optimal velocity model [25], and the intelligent driver model (IDM [26]). The detailed development of theoretical-driven car-following models can be found in [27,28].

Among the mentioned models, IDM is one of the most widely used car-following models, which has the formulations as presented in Equations (1) and (2). In this paper, IDM was calibrated and compared with the proposed model.

$$a_t = a_{\max} \left[ 1 - \left( \frac{v_t}{v_f} \right)^\beta - \left( \frac{h_t^{des}}{h_t} \right)^\beta \right] \quad (1)$$

$$h_t^{des} = h_{jam} + \max\left(0, v_t \cdot TH_{des} - \frac{v_t \cdot \Delta v_t}{2 \cdot \sqrt{a_{\max} \cdot a_{com}}}\right) \quad (2)$$

where  $a_{\max}$  and  $a_{com}$  are the maximum desired acceleration and comfortable deceleration,  $v_f$  is the maximum desired speed,  $TH_{des}$  is the desired time headway,  $h_{jam}$  is the minimum jam spacing, and  $\beta$  is the acceleration parameter.  $v_t$ ,  $h_t$ ,  $h_t^{des}$ ,  $\Delta v_t$ , and  $a_t$  refer to the speed, spacing, desired spacing, relative speed, and acceleration at time  $t$ .

### 2.2. Behavior Cloning Car-Following Models

The earlier study using behavior cloning (BC) methods to model drivers' car-following behaviors was conducted by Jia et al. [29]. The authors of the study proposed a neural network with two hidden layers that took inputs including speed, relative speed, spacing, and desired speed, and gave the acceleration as the output. Chong et al. [30] proposed a similar BC model but with only one hidden layer. The type of neural networks adopted in these studies is often called fully connected neural networks (FCN), which normally takes only a single time-step of relevant information as the input. For a sequential decision modeling problem like car-following, RNNs, which can use historical information, have gained popularity in recent years. For example, in the study conducted by Zhou et al. [10], an RNN model was proposed that was found to have better adaptability in oscillating traffic circumstances compared to the FCN approaches. Li et al. [6] applied a more advanced type of RNN called long short-term memory (LSTM) for modeling car-following. GRUs, which is a simpler version of LSTM, were adopted in the study conducted by Wang et al. [7] for modeling drivers' car-following behaviors. Except for neural networks, there are studies applying other learning methods like Gaussian mixture models and hidden Markov models for modeling car-following behavior [31,32]. In this study, the RNN model proposed by Zhou et al. [10] was selected as a comparative method.

### 2.3. Reinforcement Learning

In the setting of RL, the Markov-decision process (MDP) is used to model a sequential decision-making problem. An MDP can be defined as a tuple  $\{S, A, T, R, \gamma\}$ , where  $S$  denotes the state space,  $A$  denotes the action space,  $T$  denotes the transition matrix,  $R$  denotes the reward function, and  $\gamma$  denotes the discount factor. RL assumes an agent exists in the predefined environment. At every time-step, the agent observes a state and takes an action following its policy, gets to the next state according to the transition matrix, and it then receives a reward.

To solve an MDP using RL, the reward function must be specified, then the policy can be obtained by applying value-based or policy-based algorithms [15]. As for the study applying RL for modeling car-following, Zhu et al. [11] adopted a deep deterministic policy gradient model in which the reward function was determined as speed discrepancies between the simulated trajectories and the test data.

In this study, the MDP for modeling car-following is defined as follows: at a certain time-step  $t$ , the state includes the speed of the following vehicle denoted as  $v_t$ , the spacing between the follower and the leading vehicle denoted as  $h_t$  and the relative speed between the follower and the leader denoted as  $\Delta v_t$ . The action is the longitudinal acceleration of the follower denoted as  $a_t$ . A kinematic model is adopted as the state transition matrix (Equation (3)). The reward does not need to be defined

in this study because the proposed model can learn it from the data. The discount factor  $\gamma$  was set as 0.99 in this study.

$$\begin{aligned} v(t+1) &= v(t) + a(t) \cdot dt \\ \Delta v(t+1) &= v_{lead}(t+1) - v(t+1) \\ h(t+1) &= h(t) + \frac{\Delta v(t) + \Delta v(t+1)}{2} \cdot dt \end{aligned} \quad (3)$$

where  $dt$  is the simulation time interval which was set as 0.1 s in the present study, and  $v_{lead}$  denotes the speed of the leader that was externally inputted from the collected data.

#### 2.4. Inverse Reinforcement Learning

The goal of IRL is to learn the reward function given a set of expert demonstrations. Many approaches have been proposed to solve this problem [33]. The framework of maximum entropy IRL (Max-Ent IRL) proposed in [34] has been accepted by many studies in recent years. In the formulation of Max-Ent IRL, the probability of a trajectory is proportional to the sum of the exponential rewards gained along the trajectory (Equation (4)). Therefore, trajectories with higher rewards become exponentially more likely to happen, while trajectories with lower rewards can still happen but with exponentially fewer probabilities. As proved in [35], the principle of maximum entropy can handle the suboptimality of human behavior.

$$p(\tau) \sim \exp\left(\sum_{s,a \in \tau} r(s,a)\right) \quad (4)$$

where  $p(\tau)$  denotes the probability of a certain trajectory, and  $r(s,a)$  denotes the return of state-action pairs.

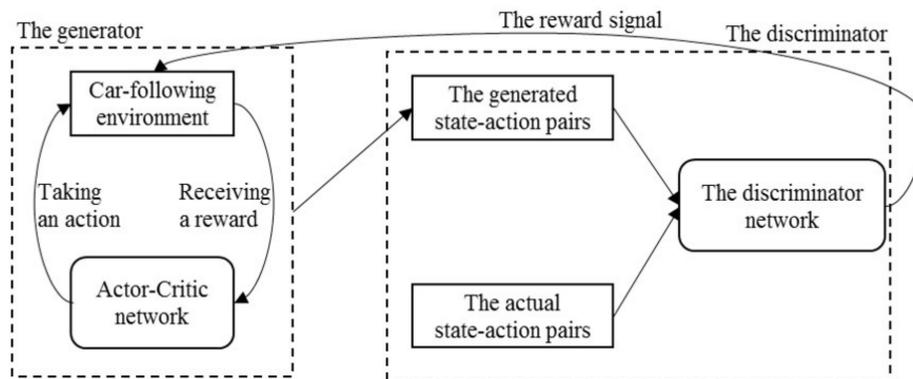
With the above formulation of trajectory distribution, the reward function represented by a linear representation or by a neural network can be learned by maximizing the log-likelihood of demonstrators' trajectories. However, the Max-Ent-IRL typically requires expensive computational power as it needs to solve the forward problem of finding an optimal policy with respect to the current reward [21], making it difficult to apply to MDPs with high dimensional state space.

#### 2.5. Generative Adversarial Imitation Learning

Since the basic ideas of IRL and generative adversarial nets (GANs) proposed by Goodfellow et al. [36] have a lot in common [37], generative adversarial imitation learning (GAIL) combines IRL with GANs, and extends Max-Ent-IRL by using the framework of GANs [38]. GAIL retains the advantages of Max-Ent-IRL. At the same time, GAIL uses a surrogate reward and directly imitates the demonstrator's behavior through policy optimization. GAIL has the advantage of low computation cost and can be better applied to complex problems. In the study conducted by Kuefler et al. [16], GAIL was applied to model drivers' behavior in highway driving using the next-generation simulation (NGSIM) datasets. Comparing with this study, the present study used a different architecture of GAIL, and different driving styles were modeled using the datasets after clustering.

### 3. The Proposed Model

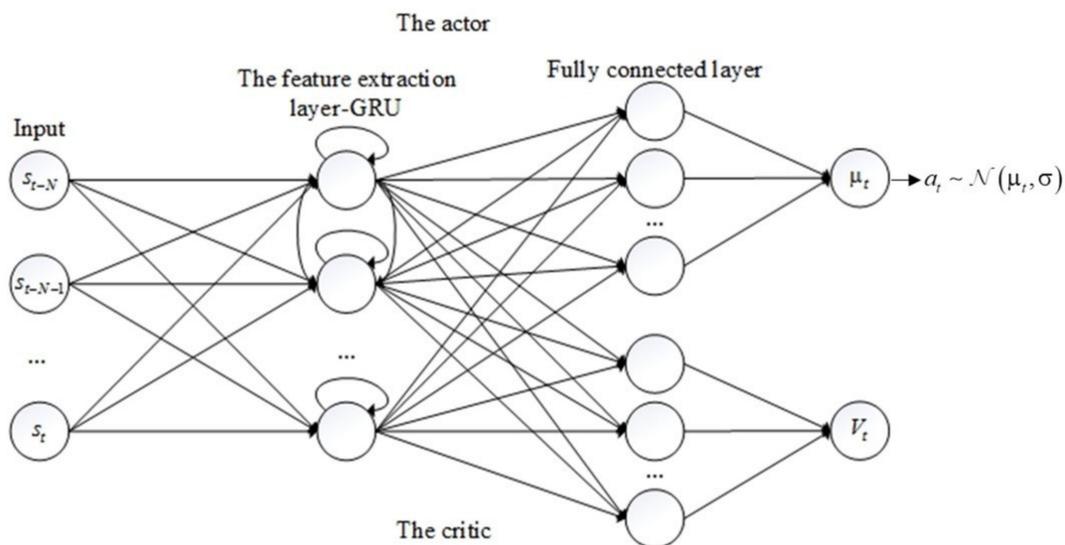
The structure of the proposed model denoted as GAIL-GRU [16,39] is shown in Figure 1 which is composed of two parts: (1) a generator that consists of an actor-critic structure and the car-following environment, which generates the faked state-action pairs, and (2) a discriminator that functions as classifying the generated state-action pairs and the actual state-action pairs obtained from the collected data. Also, the discriminator outputs the reward signal for the car-following environment. In this section, the two components of the model and the algorithm are introduced respectively.



**Figure 1.** The structure of the proposed model generative adversarial imitation learning-gated recurrent unit (GAIL-GRU).

### 3.1. The Generator

An actor-critic structure, which is the core component of the generator, was adopted as an agent to interact with the car-following environment. As can be seen in Figure 2, the actor in the upper part and critic in the lower part share the same feature extraction layer for reducing the parameters of the model. As mentioned in Section 2, the RNN that can use historical information has been widely applied in car-following modeling. Therefore, this study adopted a GRU in the feature extraction layer in which the number of neurons was set to 60. After the feature extraction layer, FCN was used to convert the output of GRU to a single value. Exponential linear units (ELUs) are used as the activation function between each layer as ELUs have been proven to overcome the vanishing gradient problem [40]. The inputs of the actor-critic is a time series state  $\{s_{t-N}, s_{t-N+1}, \dots, s_t\}_t$  in which the sequence length  $N$  was set to 10. The actor outputs the policy of the agent which is represented by a normal distribution  $\mathcal{N}(\mu_t, \sigma)$ , and the critic outputs the value  $V_t$  which represents the expected long-term reward of the input state.



**Figure 2.** The architecture of the actor-critic structure.

The proximal policy optimization (PPO) algorithm proposed in [41] was used to update the parameters of the actor-critic networks (Equations (5)–(9)). The PPO algorithm is a policy-based RL algorithm, which uses actor-critic as its basic architecture. The PPO algorithm can effectively solve the problems of high variance and difficult convergence in the training of a policy-based RL algorithm

by limiting the range of policy updates in every single step, avoiding the excessive change of policy, and making the policy improve steadily.

$$L(\theta) = \hat{E}_t \left[ L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 E[\pi_\theta](s_t) \right] \quad (5)$$

$$L_t^{CLIP}(\theta) = \hat{E}_t \left[ \min \left( \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t, \text{clip} \left( \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}, 1 - \varepsilon, 1 + \varepsilon \right) \hat{A}_t \right) \right] \quad (6)$$

$$L_t^{VF}(\theta) = \left( \sum_{t' > t} \gamma^{t'-t} r_{t'} - V_\theta(s_t) \right)^2 \quad (7)$$

$$\text{clip}(x, a, b) = \begin{cases} ax & \text{if } x \leq a \\ x & \text{if } a < x < b \\ bx & \text{if } x \geq b \end{cases} \quad (8)$$

$$\hat{A}_t = \sum_{t' > t} \gamma^{t'-t} r_{t'} - V_\theta(s_t) \quad (9)$$

where  $L(\theta)$  is the loss function of the actor-critic networks;  $L_t^{CLIP}(\theta)$  denotes the loss for the actor and  $L_t^{VF}(\theta)$  denotes the loss for the critic;  $V_\theta(s_t)$  denotes the value of state  $s_t$ ;  $\hat{A}_t$  denotes the advantage value of state  $s_t$ ;  $\theta$  is the parameters of the actor-critic networks;  $\hat{E}_t$  denotes the expectation operation;  $c_1, c_2, \varepsilon$  are all parameters which all set to 0.5, 0.01, and 0.2, respectively;  $\pi_{\theta_{old}}(a_t|s_t)$  is the last output of the actor; and  $E[\pi_\theta](s_t)$  denotes the entropy of policy  $\pi_\theta$ .

### 3.2. The Discriminator

The discriminator is an FCN with two hidden layers as depicted in Figure 3. Each hidden layer has 64 neurons. ELUs were used as the activation function except for the last layer, which used the sigmoid function. The input of the discriminator is state-action pairs, and the output is the probability of the input belonging to each category. The generated state-action pairs and the actual state-action pairs were marked as 1 and 0, respectively. The cross-entropy loss function commonly used in the binary classification problem, as shown in Equation (10), was used to update the parameters of the discriminator.

$$L(\omega) = \frac{1}{N} \sum_i -[y^i \cdot \log(D_\omega(s^i, a^i)) + (1 - y^i) \cdot \log(1 - D_\omega(s^i, a^i))] \quad (10)$$

where  $L(\omega)$  is the loss of the discriminator networks,  $N$  is the number of the training samples,  $y^i$  is the label of sample  $i$ ,  $(s^i, a^i)$  is the  $i$ th state-action pair, and  $D_\omega(s^i, a^i)$  denotes the output of the discriminator parameterized by  $\omega$ .

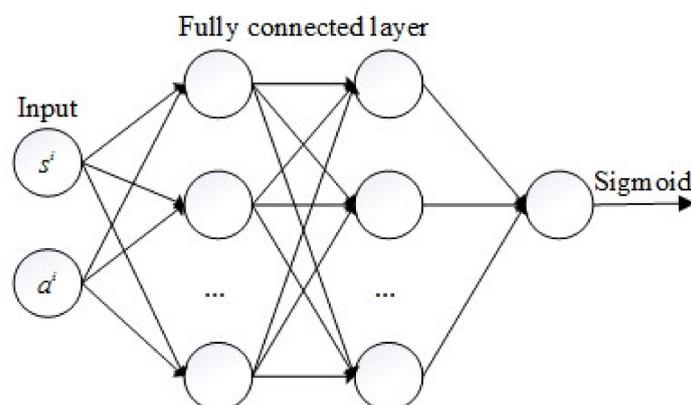


Figure 3. The architecture of the discriminator.

The discriminator also provides the reward signals for the car-following environment. The state-action reward  $r(s_t, a_t)$  that denotes the return of taking action  $a_t$  in the state  $s_t$  is determined by Equation (11). It can be inferred that when the generated state-action samples get closer to the actual samples, the output of the discriminator for the generated samples becomes closer to 0, and the reward becomes higher. Therefore, the reward signal provided by the discriminator is actually encouraging the agent to act more similarly to the demonstrator.

$$r(s_t, a_t) = -\log(D_\omega(s_t, a_t)) \quad (11)$$

### 3.3. The Proposed Algorithm

The proposed algorithm is showed presented in Algorithm 1 below. The machine learning package PyTorch (Facebook, Menlo Park, CA, USA) was used for implementing the proposed model. The Adam optimizer [42] was used for training the model. In every training episode, the discriminator was trained for more times than that of the actor-critic networks to facilitate the convergence of the entire model [43].

---

#### Algorithm 1: Algorithm for modeling drivers' car-following behaviors

---

**Input:** The collected state-action pairs of drivers denoted as  $D$

**Output:**

1: The strategy of drivers denoted as  $\pi$

2: **Algorithm begins:**

3: Randomly initialize the parameters of the discriminator and the actor-critic networks as  $\theta_0$  and  $\omega_0$ .

4: For  $i = 0$  to  $N$ , repeat the following steps

5: Using the policy  $\pi_i$  output by the actor-critic parameterized by  $\omega_i$  to interact with the car-following environment, record the interaction history as the generated state-action pairs  $D_{G_i}$

6: Update the parameters of the discriminator for three times from  $\theta_i$  to  $\theta_{i+1}$  with the generated samples  $D_{G_i}$  and the actual samples  $D$  using the Adam rule and the loss  $L(\omega)$ .

7: Use the updated discriminator parameterized by  $\theta_{i+1}$  to provide rewards (Equation (11)) for the environment, update the parameters of the actor-critic networks from  $\omega_i$  to  $\omega_{i+1}$  using the Adam rule and the loss  $L(\theta)$ .

8: **Algorithm end**

---

## 4. Data Description

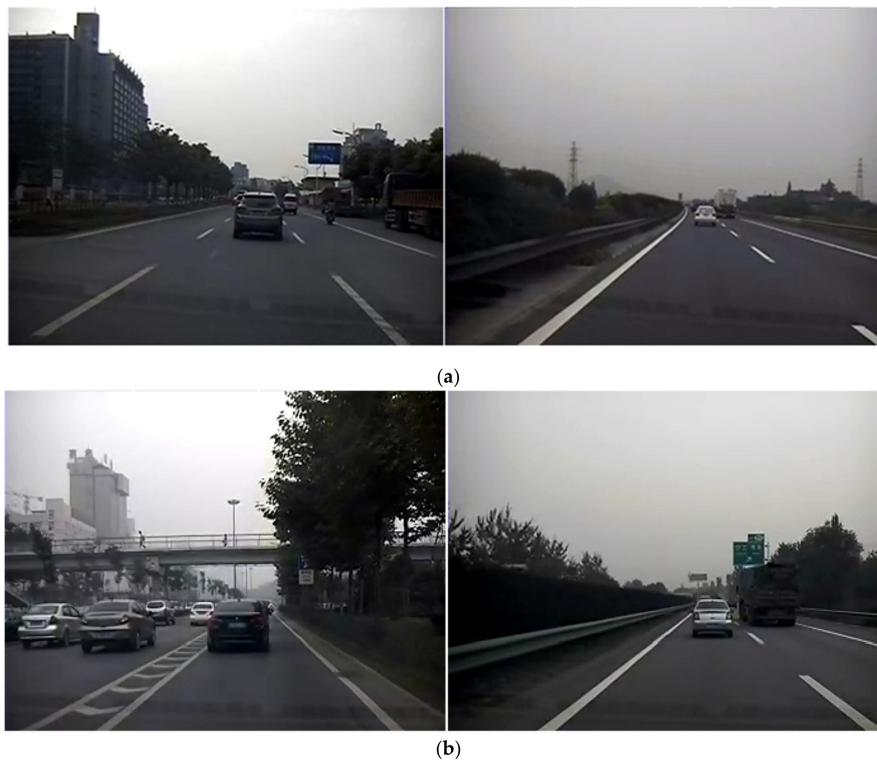
### 4.1. The Experiments

Data from two field tests conducted in Huzhou City in Zhejiang Province and Xi'an city in Shaanxi Province in China were used in this study. The driving route included urban roads and expressways, as shown in Figure 4. The urban section is part of the Huzhou section of national highway 104 (where the speed limit is 70 km/h) and part of the east third ring road and south third ring road of Xi'an (where the speed limit is 100 km/h), and the highway section is the Huzhou-Changxing section of the G25 highway (where the speed limit is 120 km/h) and the Xiangwang-Lantian section of G40 highway (where the speed limit is 120 km/h).

A total of 42 drivers participated in the experiment. All of the subjects were male, the average age was 40.3 years old and the standard deviation was 5.1. The participants' driving experiences ranged from 2 to 30 years with the average being 15.2 years and the standard deviation was 7.5. During the test, all of the subjects were asked to drive the test vehicle according to their personal driving styles, and they were only told the starting and ending positions. It took 90–110 min for the subjects to complete the entire test.

The test data were collected by a test vehicle equipped with a variety of sensors as exhibited in Figure 5. Among them, the millimeter-wave radar can collect the motion states of the surrounding vehicles during driving including the spacing and relative speed with the targets. The detection angle

is  $\pm 10^\circ$  and 64 targets around the test vehicle can be tracked at the same time. The speed of the test vehicle was collected from the CAN acquisition card. All of the recorded data were sampled at 10 Hz. During the tests, the real-time pictures of drivers' faces, as well as the front, back, and sides of the vehicle, were captured by the video monitoring system of the vehicle, which facilitates the verification of uncertain car-following periods.



**Figure 4.** Driving scenarios in (a) Huzhou city and (b) Xi'an city.



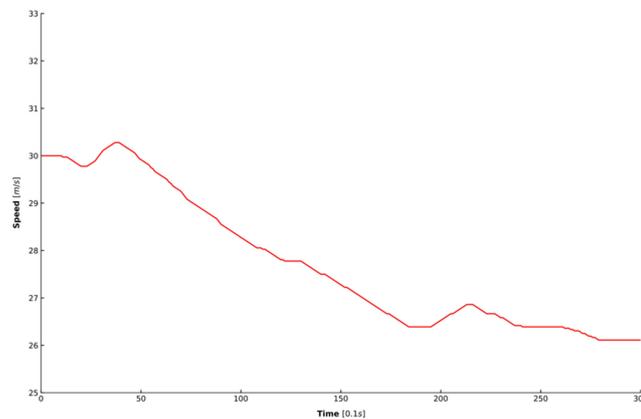
**Figure 5.** The test vehicle and related equipment.

#### 4.2. Car-Following Periods Extraction

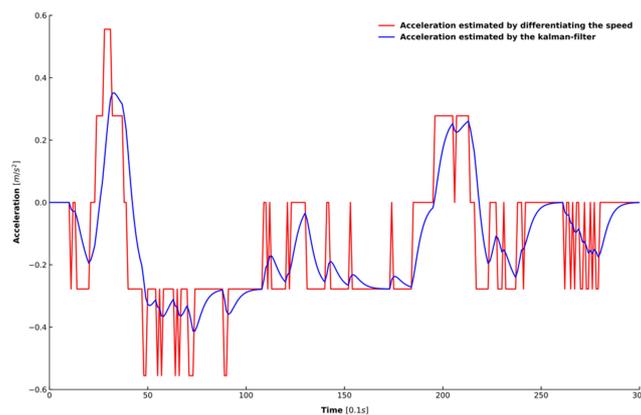
In this study, the rules for determining a car-following period implemented in a previous study were adopted [44]. The details include the following: (1) the test vehicle should follow the same leading car throughout the entire period; (2) the distance between the test vehicle and leading vehicle should be less than 120 m; (3) the test vehicle and leading vehicle should be in the same lane; and (4) the

length of the car-following periods should be greater than 15 s. A program was designed to filter the data segments that meet the above criteria, and then the filtered segments were checked by manually playing back the video recorded by the front camera of the test vehicle. The checking procedure includes the confirmation of the validation of every period and the determination of the starting point and the endpoint of each car-following period. Finally, a total of nearly one thousand car-following periods were extracted from the collected data. The extracted car-following data included the speed of the ego vehicle, the spacing from the leading vehicle, and the relative speed between the ego vehicle and leading vehicle which were all measured by the instrumented vehicle.

Two steps were taken to process the extracted data. First, all of the data were processed by a moving average filter to eliminate the noise in the data [45], where the length of the sliding window was set as 1 s. Second, because acceleration is also needed to train the model, this study adopted a Kalman filter for estimating acceleration because simply differentiating the speed cannot achieve an accurate estimation of the acceleration [46]. Figure 6 shows the speed in one car-following period randomly selected from the dataset. Figure 7 depicts the results of the acceleration estimated from the raw signal using the two different approaches mentioned above. It can be observed that the estimation results of the Kalman filter in blue is smoother and more accurate than that of differentiating the speed in red.



**Figure 6.** The speed in one car-following period.



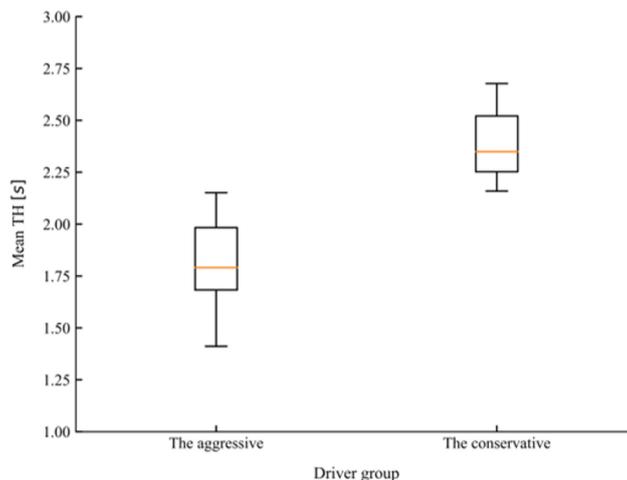
**Figure 7.** The comparison of different methods for estimating acceleration from the raw signal.

#### 4.3. Driving Style Clustering Based on K-Means

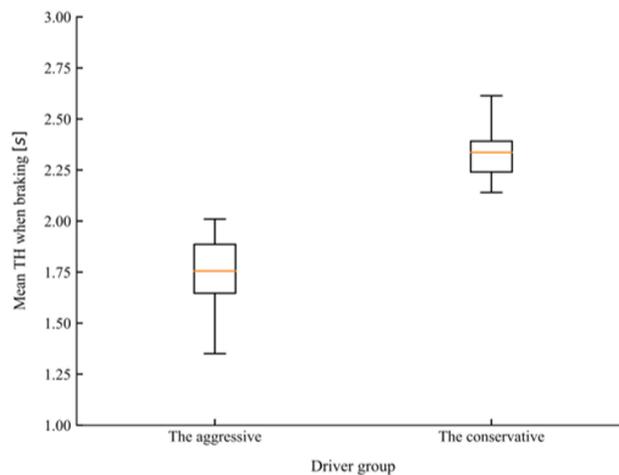
Because the proposed model is based on RL, it assumes that the agent that should be modeled has certain kinds of preferences or rewards. Every individual driver may have different preferences during car-following. However, in the literature of driving-behavior research, it is common to cluster drivers into different driving styles such as aggressive or conservative. Drivers belonging to one group may have similar preferences. So, in this study, instead of using the car-following data of each participant

to train the proposed model, K-means was initially applied to cluster the participants into different driving styles. The data from the different groups were then used for model training.

There are many indicators that can be used to characterize the driving style of drivers in car-following. Except for basic kinematic indicators, such as spacing, speed, or relative speed, time-based variables such as time headway (TH) and time-to-collision (TTC) have also been accepted for characterizing driving styles [47,48]. In the study conducted by Liu et al. [49], mean TH and mean TH when braking were adopted as features to classify drivers into different driving styles. In the present study, different combinations of the above-mentioned features were selected as the inputs for the K-means algorithm, the silhouette coefficient was used to evaluate the quality of the clustering results [50]. Before carrying out clustering, the car-following dataset was first normalized to eliminate the influence of dimension. Finally, mean TH and mean TH when braking were determined as feature vectors for clustering as the combination achieved the highest silhouette coefficient among all other choices. According to the value of the silhouette coefficient, the number of clusters was also determined to be two. Figures 8 and 9 present the comparison of mean TH and mean TH when braking of the two groups of drivers after clustering. T-tests showed that the conservative group, which consisted of 16 drivers, maintained a significantly higher mean TH ( $t = 6.748, p < 0.001$ ) and mean TH when braking ( $t = 7.655, p < 0.001$ ) than the aggressive group, which consisted of 26 drivers.



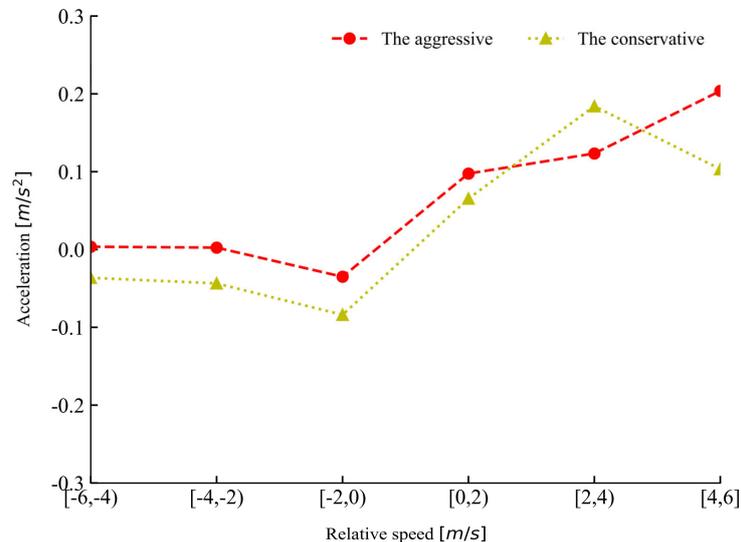
**Figure 8.** The comparison of mean time headway (TH) for the two driving styles.



**Figure 9.** The comparison of mean time headway (TH) when braking for the two driving styles.

Figure 10 shows the changes for mean acceleration for the two groups of drivers in different ranges of relative speed during car-following. The aggressive drivers kept higher acceleration compared to

the conservative drivers in all the ranges except in the range of 2–4 m/s. Table 1 presents the descriptive statistics of car-following data from the two groups of drivers. The conservative drivers maintained a lower mean speed and acceleration, with a larger mean spacing compared to the aggressive drivers. It can be seen that the two groups of drivers exhibited distinct driving styles.



**Figure 10.** Rule of changes for acceleration in different relative speed ranges for the two groups of drivers.

**Table 1.** Descriptive statistics of the two groups of drivers in car-following.

Type	Spacing (m)			Speed (m/s)		
	Mean	Min	Max	Mean	Min	Max
Aggressive	34.15	2.27	120.00	18.57	2.50	33.70
Conservative	42.79	3.91	119.90	17.47	1.49	34.84
Type	Acceleration (m/s <sup>2</sup> )			Relative Speed (m/s)		
	Mean	Min	Max	Mean	Min	Max
Aggressive	-0.01	-3.14	1.64	-0.42	-12.45	5.80
Conservative	-0.04	-1.80	1.40	-0.52	-8.85	5.02

## 5. Model Training and Evaluation

### 5.1. Simulation Setup

The initial state  $s_0$  which included the speed, spacing, and relative speed of the following vehicle, was externally input from the empirical data to enable the RL agent to interact with the car-following environment. Following this, the RL agent chooses an action based on its policy  $a_t \sim \pi(a_t|s_t)$ , and the future states of the following vehicle were updated by the state-transition matrix  $s_{t+1} = T(s_{t+1}|a_t, s_t)$ . The simulation for every car-following period terminated when it reached its maximum time-step.

### 5.2. Evaluation Matrix

#### 5.2.1. Root Mean Square Percentage Error

As suggested in [51], this study used the root mean square percentage error (RMSPE) of spacing and speed between simulation results and actual data to evaluate the performance of car-following models. The evaluation matrices are as follows:

$$RMSPE(v) = \sqrt{\frac{\sum_{i=1}^N (v_i^{sim} - v_i^{obs})^2}{\sum_{i=1}^N (v_i^{obs})^2}} \quad (12)$$

$$RMSPE(h) = \sqrt{\frac{\sum_{i=1}^N (h_i^{sim} - h_i^{obs})^2}{\sum_{i=1}^N (h_i^{obs})^2}} \quad (13)$$

where  $RMSPE(v)$  and  $RMSPE(h)$  denote the RMSPE of speed and spacing, respectively;  $v_i^{obs}, h_i^{obs}$  denote the empirical speed and spacing in  $i$ th time; and  $v_i^{sim}, h_i^{sim}$  denote the simulated speed and spacing in  $i$ th time.

### 5.2.2. Kullback-Leibler Divergence

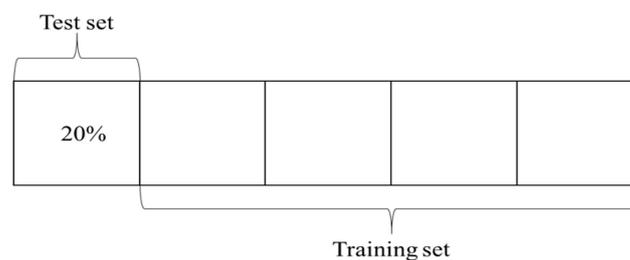
Kullback-Leibler (KL) divergence is a measure of the difference between two distributions that can be calculated by Equation (14). When the value of the KL divergence is smaller, the two distributions become more similar. The car-following models should produce a similar distribution of representative features with the drivers' demonstrations. Because this study used mean TH and mean TH when braking to divide the drivers into two different driving styles, the similarity between the distribution of the simulated mean TH and mean TH when braking and the empirical distributions were measured by calculating the KL divergence to evaluate the performance of car-following models in reproducing drivers' driving styles.

$$KL(P, Q) = \sum P(x) \log \frac{P(x)}{Q(x)} \quad (14)$$

where  $P, Q$  denotes two distributions.

### 5.3. Cross-Validation

K-Fold cross-validation was adopted in this study to evaluate the generalization ability of car-following models. Specifically,  $k$  was determined to be 5 in this study due to the scale of the dataset. The dataset of aggressive and conservative drivers was equally divided into 5 groups. As illustrated in Figure 11, every fold was taken as the test set, and the remaining four folds were taken as the training set. The procedure was repeated five times, and then the average performance of RMSPE was calculated for the training sets and the test sets.



**Figure 11.** Five-fold cross-validation.

### 5.4. Models Investigated

As mentioned in Section 2, the representative theoretical-driven model IDM and recent behavior cloning model RNN were chosen to be compared with the proposed model.

### 5.4.1. IDM

The parameters of IDM were calibrated with the objective of minimizing the RMSPE of spacing as Equation (15). A large penalty of the crash was also added in the objective function, as no collision was observed in the collected data. In this paper, the population size and the maximum iteration times of the genetic algorithm (GA) are set to 100. In order to reduce the influence of the randomness of GA on the calibration of IDM, the GA algorithm is repeated 12 times, and the final result is the combination of parameters with the minimum error.

$$c(IDM) = RMSPE(h) + \lambda \cdot n_{crash} \quad (15)$$

where  $RMSPE(h)$  denotes the RMSPE of spacing,  $n_{crash}$  denotes the number of crashes when applying the calibrated IDM model for simulation, and  $\lambda$  is the parameter for the crash penalty.

### 5.4.2. RNN Based Model

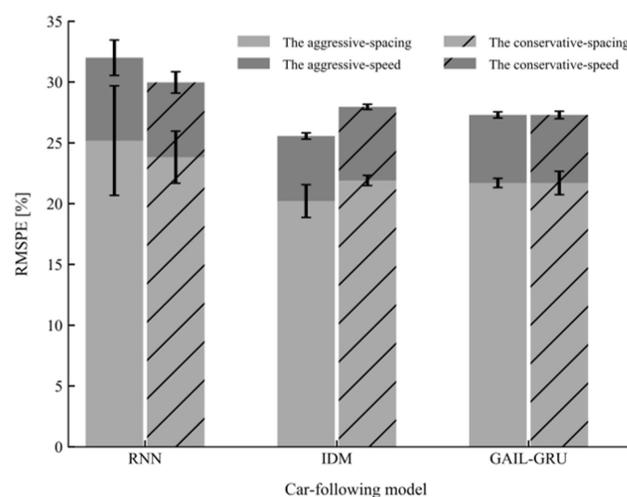
In accordance with another study [10], an RNN with 60 neurons for its hidden layer was built. The model takes a time sequence of 1 s as the input. The gradient descent algorithm was used to train the model with the cost function defined as follows.

$$c(RNN) = RMSPE(v) + RMSPE(h) \quad (16)$$

where  $RMSPE(v)$  and  $RMSPE(h)$  denotes the RMSPE of speed and spacing respectively.

## 6. Results

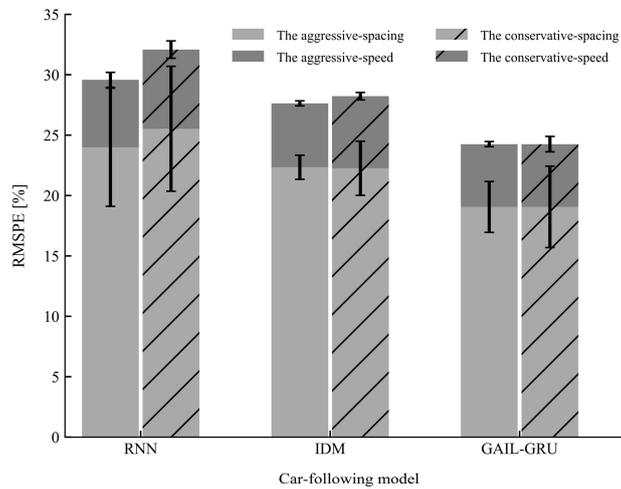
The performances of the proposed model GAIL-GRU, the theoretical-driven model IDM, and the recent behavior-cloning model RNN for replicating the car-following trajectories of the drivers with different driving styles on the training sets and test sets are presented in Figures 12 and 13, respectively. As can be observed, GAIL-GRU outperformed the other two models as it achieved the lowest training error and test error for speed and spacing in both datasets. The RNN model has the highest training error and test error among the three models. For IDM, even though it exhibited a similar performance with GAIL-GRU on the training sets, it failed to generalize to the same level as GAIL-GRU on the test sets.



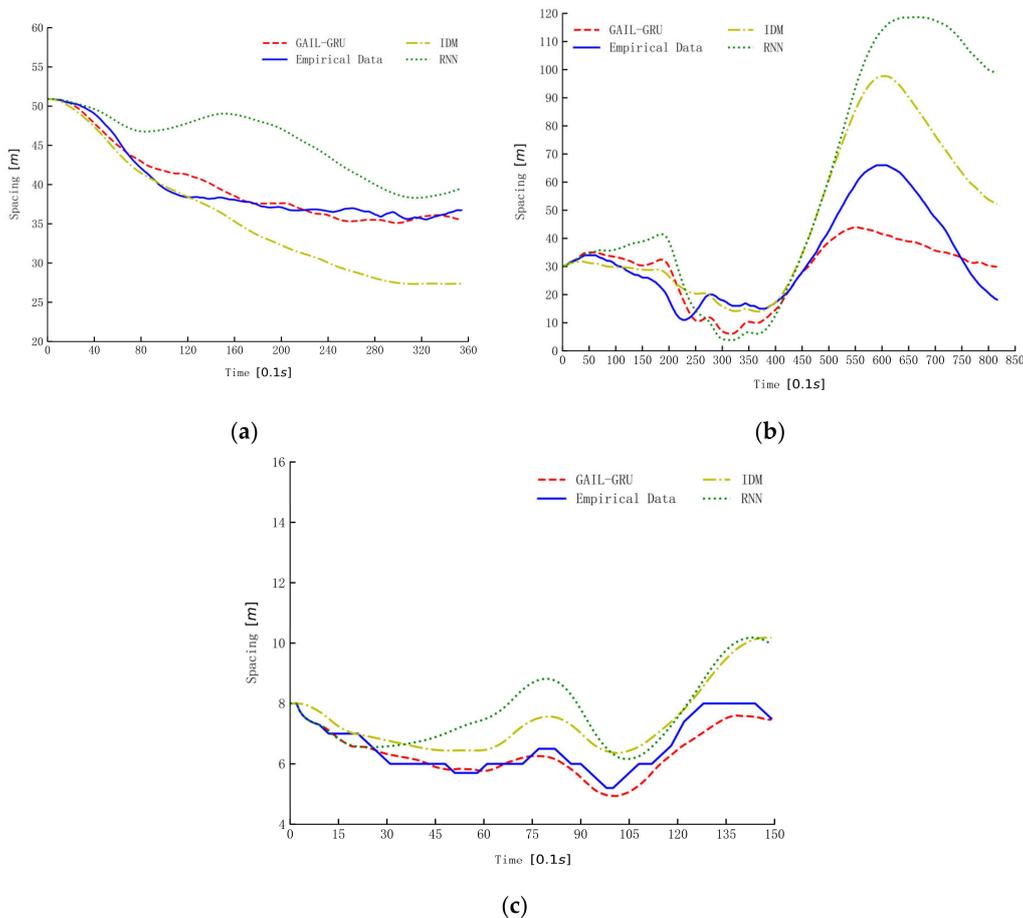
**Figure 12.** Mean and standard deviation of the root mean square percentage error (RMSPE) on the training sets for the three models.

The three models trained by the same data from the aggressive dataset were used for simulation to illustrate and compare the characters of the three models in reproducing drivers' car-following

trajectories. The simulation results of spacing for three car-following periods randomly selected from the dataset are presented in Figure 14. As can be seen, the proposed GAIL-GRU tracks the empirical spacing more closely than the IDM and RNN methods. The simulation results of RNN and GAIL-GRU, which all make use of historical information, are similar in the first 10 s or so, but the deviation from the empirical data becomes increasingly larger for RNN as the simulation continues.

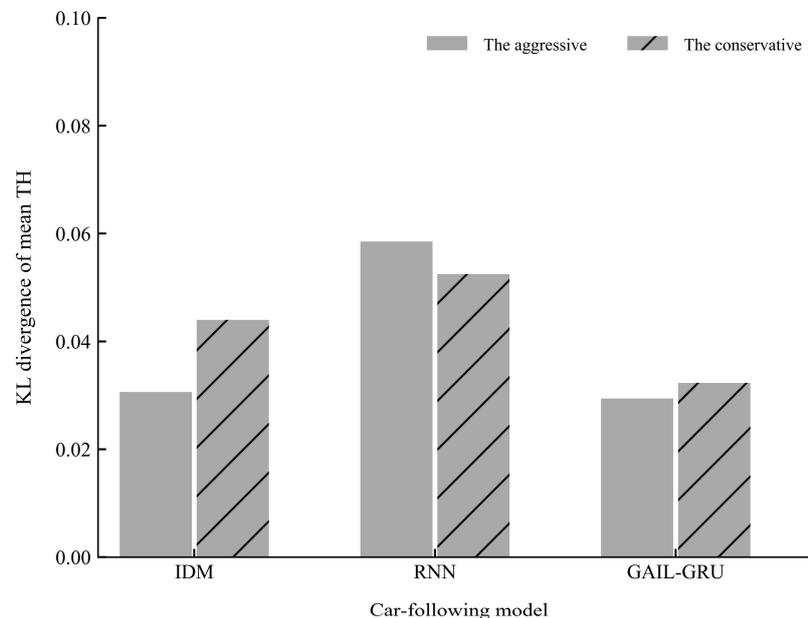


**Figure 13.** Mean and standard deviation of the root mean square percentage error (RMSPE) on the test sets for the three models.

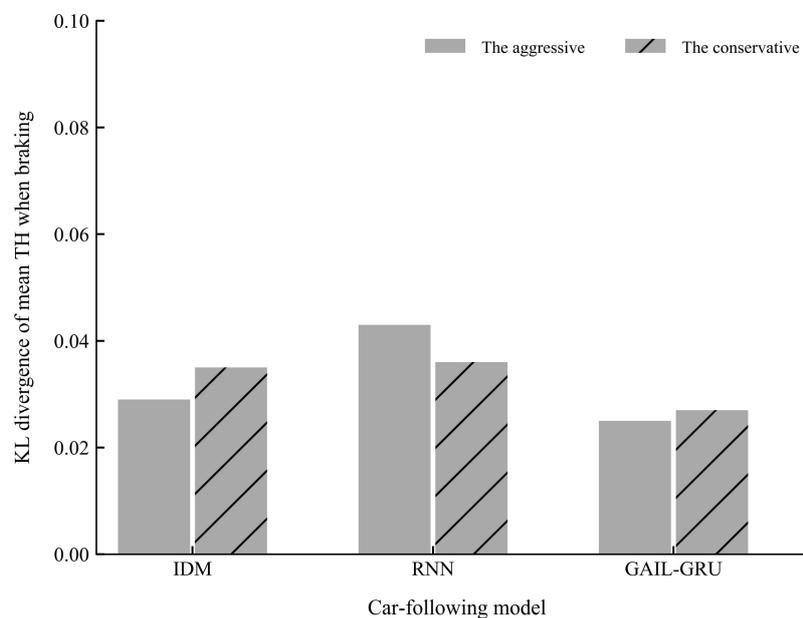


**Figure 14.** The simulation of spacing by different models. (a–c) are the simulation results of spacing for the three car-following periods randomly selected from the datasets.

Figures 15 and 16 present the comparison of the KL divergence for mean TH and mean TH when braking, respectively. As can be seen, the GAIL-GRU model can produce the closest distribution to the empirical data of the two indicators which were used to cluster the drivers into two different driving styles.



**Figure 15.** The Kullback-Leibler (KL) divergence of mean time headway (TH) for different models.



**Figure 16.** The Kullback-Leibler (KL) divergence of mean time headway (TH) when braking for different models.

## 7. Discussion

The present study demonstrated that the proposed GAIL-GRU model can replicate human-like car-following behaviors and driving styles more accurately than the theoretical-driven model IDM and the behavior-cloning model RNN. Compared with IDM, the proposed model used the actor neural network which consists of two hidden layers including a GRU network and a fully connected network, to fit drivers' car-following policies, while IDM only contains six parameters to be calibrated.

Significantly more parameters were contained in the proposed model, accounting for a better fitting ability and generalization ability compared to the IDM.

Compared with RNN, the proposed model also used a kind of RNN to represent drivers' strategies, and the length of the input was also set to be 1 s. However, the RNN model has a higher error for replicating car-following trajectories. From the simulation results of spacing for three car-following periods, it can be seen that the RNN experienced the problem of cascading errors, while the proposed model did not have this problem. The reason for this is that the RNN only learns the state-action relationships during training. When the input for RNN is not included in the training data, the output of RNN may deviate from drivers' actual control actions. The proposed model is based on RL, the strategy output by the proposed model has the objective of maximizing the accumulative rewards, which award behavior similar to the demonstrator. Therefore, the strategy obtained by the proposed model can be extended to unseen states.

Comparing RNN with IDM, the error of the BC model is higher than the theoretical-driven model, which seems to be inconsistent with previous studies [7,11]. However, it must be noted that the RMSPE of RNN has large variances. In the five-fold cross-validation tests, RNN achieved a comparatively low error with IDM in some of the validation, while obtaining a much higher error in others. It seems that the performance of RNN is largely dependent on the division of training and test data. This issue has also been mentioned in the study conducted by Wang et al. [7], which pointed out that the selection of training data has a great impact on the performance of purely data-driven BC car-following models. The five-fold cross-validation methods adopted in this study can eliminate the influence of dataset division on the performance of car-following models, which can help to make a more reliable evaluation of the actual performance of the model in real applications.

The proposed model borrows the idea of GANs. In the fields of computer vision, GANs have been proven to be able to generate life-like images to the original image in recent years. In this study, the proposed model is demonstrated to be able to generate car-following strategies and driving styles closest to the drivers. It can be seen that the training process of the proposed model is very similar to that of GANs. The essence of the training process is the process of mutual confrontation between the generator and the discriminator. The training of the discriminator makes the discriminator distinguish the category of the samples more accurately. However, the training of the generator produces samples more similar to drivers' actual behaviors, so as to cheat the discriminator. After training, the samples generated by the generator can be perfectly close to the distribution of the real samples, while the discriminator cannot correctly tell the difference between the generated samples from the actual samples. The proposed model can output the strategy that is identical to the drivers' actual behaviors.

This study also had some important limitations. First, the car-following behaviors modeled in this study could only represent a small group of drivers. A broader sample is needed in future studies. Secondly, this study did not consider the influence of vehicles in adjacent lanes on drivers' car-following behaviors. These factors will be fully considered in future research to obtain a more accurate description of drivers' behaviors.

**Author Contributions:** Conceptualization, Y.Z. and R.F.; methodology and conducting experiments, Y.Z. and C.W.; data analysis, R.Z.; funding acquisition, C.W.; writing Y.Z. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was jointly supported by the National Key R&D Program of China under Grant 2019YFB1600500, the Changjiang Scholars and Innovative Research Team in University under Grant IRT\_17R95, the National Natural Science Foundation of China under Grants (51775053, 51908054), the Natural Science Foundation of Shaanxi Province under Grants (2020JQ-908,2020JQ-910), and the Key Laboratory of Transport Industry of Management, Control and Cycle Repair Technology for Traffic Network Facilities in Ecological Security Barrier Area (Inner Mongolia Transport Development Research Center) under Grant 2019KFJJ-004.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Brackstone, M.; McDonald, M. Car-following: A historical review. *Transp. Res. Part F Traffic Psychol. Behav.* **1999**, *2*, 181–196. [\[CrossRef\]](#)
2. Peng, H. Evaluation of driver assistance systems—a human centered approach. In Proceedings of the 6th International Symposium on Advanced Vehicle Control, Hiroshima, Japan, 9–13 September 2002.
3. Simonelli, F.; Bifulco, G.N.; De Martinis, V.; Punzo, V. Human-Like Adaptive Cruise Control Systems through a Learning Machine Approach. In *Applications of Soft Computing*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 240–249.
4. Kuderer, M.; Gulati, S.; Burgard, W. Learning driving styles for autonomous vehicles from demonstration. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 2641–2646.
5. Sim, G.; Min, K.; Ahn, S.; Sunwoo, M.; Jo, K. Deceleration Planning Algorithm Based on Classified Multi-Layer Perceptron Models for Smart Regenerative Braking of EV in Diverse Deceleration Conditions. *Sensors* **2019**, *19*, 4020. [\[CrossRef\]](#) [\[PubMed\]](#)
6. Li, Y.; Lu, X.; Ren, C.; Zhao, H. Fusion Modeling Method of Car-Following Characteristics. *IEEE Access* **2019**, *7*, 162778–162785. [\[CrossRef\]](#)
7. Wang, X.; Jiang, R.; Li, L.; Lin, Y.; Zheng, X.; Wang, F. Capturing Car-Following Behaviors by Deep Learning. *IEEE Trans. Intell. Transp. Syst.* **2018**, *19*, 910–920. [\[CrossRef\]](#)
8. Hao, S.; Yang, L.; Shi, Y. Data-driven car-following model based on rough set theory. In *IET Intelligent Transport Systems*; Institution of Engineering and Technology: London, UK, 2018; Volume 12, pp. 49–57.
9. Papanthanasopoulou, V.; Antoniou, C. Towards data-driven car-following models. *Transp. Res. Part C Emerg. Technol.* **2015**, *55*, 496–509. [\[CrossRef\]](#)
10. Zhou, M.; Qu, X.; Li, X. A recurrent neural network based microscopic car following model to predict traffic oscillation. *Transp. Res. Part C Emerg. Technol.* **2017**, *84*, 245–264. [\[CrossRef\]](#)
11. Zhu, M.; Wang, X.; Wang, Y. Human-like autonomous car-following model with deep reinforcement learning. *Transp. Res. Part C Emerg. Technol.* **2018**, *97*, 348–368. [\[CrossRef\]](#)
12. Codevilla, F.; Santana, E.; López, A.M.; Gaidon, A. Exploring the limitations of behavior cloning for autonomous driving. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019; pp. 9329–9338.
13. Ross, S.; Bagnell, D. Efficient reductions for imitation learning. In Proceedings of the 13th International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 13–15 May 2010; pp. 661–668.
14. Gao, H.; Shi, G.; Xie, G.; Cheng, B. Car-following method based on inverse reinforcement learning for autonomous vehicle decision-making. *Int. J. Adv. Robotic Syst.* **2018**, *15*, 1729881418817162. [\[CrossRef\]](#)
15. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
16. Kuefler, A.; Morton, J.; Wheeler, T.; Kochenderfer, M. Imitating driver behavior with generative adversarial networks. In Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, USA, 11–14 June 2017; pp. 204–211.
17. Abbeel, P.; Ng, A.Y. Apprenticeship learning via inverse reinforcement learning. In Proceedings of the Twenty-First International Conference on Machine Learning, Banff, AL, Canada, 4–8 July 2004; p. 1.
18. Zhang, Q.; Zhu, M.; Zou, L.; Li, M.; Zhang, Y. Learning Reward Function with Matching Network for Mapless Navigation. *Sensors* **2020**, *20*, 3664. [\[CrossRef\]](#)
19. Ng, A.Y.; Russell, S.J. Algorithms for inverse reinforcement learning. *Icml* **2000**, *1*, 2.
20. Wulfmeier, M.; Ondruska, P.; Posner, I. Maximum entropy deep inverse reinforcement learning. *arXiv* **2015**, arXiv:1507.04888.
21. Finn, C.; Levine, S.; Abbeel, P. Guided cost learning: Deep inverse optimal control via policy optimization. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 49–58.
22. Pipes, L.A. An Operational Analysis of Traffic Dynamics. *J. Appl. Phys.* **1953**, *24*, 274–281. [\[CrossRef\]](#)
23. Chandler, R.E.; Herman, R.; Montroll, E.W. Traffic dynamics: Studies in car following. *Oper. Res.* **1958**, *6*, 165–184. [\[CrossRef\]](#)
24. Gipps, P.G. A behavioural car-following model for computer simulation. *Transp. Res. Part B Methodol.* **1981**, *15*, 105–111. [\[CrossRef\]](#)

25. Bando, M.; Hasebe, K.; Nakanishi, K.; Nakayama, A. Analysis of optimal velocity model with explicit delay. *Phys. Rev. E* **1998**, *58*, 5429–5435. [[CrossRef](#)]
26. Kesting, A.; Treiber, M. Calibrating Car-Following Models by Using Trajectory Data: Methodological Study. *Transp. Res. Record* **2008**, *2088*, 148–156. [[CrossRef](#)]
27. Panwai, S.; Dia, H. Comparative evaluation of microscopic car-following behavior. *IEEE Trans. Intel. Transp. Syst.* **2005**, *6*, 314–325. [[CrossRef](#)]
28. Saifuzzaman, M.; Zheng, Z. Incorporating human-factors in car-following models: A review of recent developments and research needs. *Transp. Res. Part C Emerg. Technol.* **2014**, *48*, 379–403. [[CrossRef](#)]
29. Jia, H.; Juan, Z.; Ni, A. Develop a car-following model using data collected by “five-wheel system”. In Proceedings of the 2003 IEEE International Conference on Intelligent Transportation Systems, Shanghai, China, 12–15 October 2003; Volume 341, pp. 346–351.
30. Chong, L.; Abbas, M.M.; Medina, A. Simulation of Driver Behavior with Agent-Based Back-Propagation Neural Network. *Transp. Res. Record* **2011**, *2249*, 44–51. [[CrossRef](#)]
31. Lefèvre, S.; Carvalho, A.; Borrelli, F. A Learning-Based Framework for Velocity Control in Autonomous Driving. *IEEE Trans. Autom. Sci. Eng.* **2016**, *13*, 32–42. [[CrossRef](#)]
32. Wang, W.; Xi, J.; Hedrick, J.K. A Learning-Based Personalized Driver Model Using Bounded Generalized Gaussian Mixture Models. *IEEE Trans. Veh. Technol.* **2019**, *68*, 11679–11690. [[CrossRef](#)]
33. Arora, S.; Doshi, P. A survey of inverse reinforcement learning: Challenges, methods and progress. *arXiv* **2018**, arXiv:1806.06877.
34. Ziebart, B.D.; Maas, A.L.; Bagnell, J.A.; Dey, A.K. Maximum entropy inverse reinforcement learning. In Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, Chicago, IL, USA, 13–17 July 2008; pp. 1433–1438.
35. Ziebart, B.D.; Bagnell, J.A.; Dey, A.K. Modeling interaction via the principle of maximum causal entropy. *J. Contrib.* **2010**. [[CrossRef](#)]
36. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 2672–2680.
37. Finn, C.; Christiano, P.; Abbeel, P.; Levine, S. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv* **2016**, arXiv:1611.03852.
38. Ho, J.; Ermon, S. Generative adversarial imitation learning. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 4565–4573.
39. Fernando, T.; Denman, S.; Sridharan, S.; Fookes, C. Learning temporal strategic relationships using generative adversarial imitation learning. *arXiv* **2018**, arXiv:1805.04969.
40. Clevert, D.-A.; Unterthiner, T.; Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). *arXiv* **2015**, arXiv:1511.07289.
41. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
42. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
43. Yu, L.; Zhang, W.; Wang, J.; Yu, Y. Seqgan: Sequence generative adversarial nets with policy gradient. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017.
44. Zhu, M.; Wang, X.; Tarko, A.; Fang, S.E. Modeling car-following behavior on urban expressways in Shanghai: A naturalistic driving study. *Transp. Res. Part C Emerg. Technol.* **2018**, *93*, 425–445. [[CrossRef](#)]
45. Thiemann, C.; Treiber, M.; Kesting, A. Estimating Acceleration and Lane-Changing Dynamics from Next Generation Simulation Trajectory Data. *Transp. Res. Record* **2008**, *2088*, 90–101. [[CrossRef](#)]
46. Welch, G.; Bishop, G. *An Introduction to the Kalman Filter*; University of North Carolina: Chapel Hill, NC, USA, 1995.
47. Boer, E.R. Car following from the driver’s perspective. *Transp. Res. Part F Traffic Psychol. Behav.* **1999**, *2*, 201–206. [[CrossRef](#)]
48. Mata-Carballeira, Ó.; Gutiérrez-Zaballa, J.; del Campo, I.; Martínez, V. An FPGA-Based Neuro-Fuzzy Sensor for Personalized Driving Assistance. *Sensors* **2019**, *19*, 4011. [[CrossRef](#)] [[PubMed](#)]
49. Liu, T.; Fu, R.; Ma, Y.; Liu, Z.; Cheng, W. Car-following Warning Rules Considering Driving Styles. *China J. Highw. Transp.* **2020**, *33*, 170–180.

50. Aranganayagi, S.; Thangavel, K. Clustering categorical data using silhouette coefficient as a relocating measure. In Proceedings of the International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007), Sivakasi, Tamil Nadu, 13–15 December 2007; pp. 13–17.
51. Punzo, V.; Montanino, M. Speed or spacing? Cumulative variables, and convolution of model errors and time in traffic flow models validation and calibration. *Transp. Res. Part B Methodol.* **2016**, *91*, 21–33. [[CrossRef](#)]



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).