Adversarial Evaluation of Autonomous Vehicles in Lane-Change Scenarios

Baiming Chen, Liang Li

Abstract—Autonomous vehicles must be comprehensively evaluated before deployed in cities and highways. Current evaluation procedures lack the abilities of weakness-aiming and evolving, thus they could hardly generate adversarial environments for autonomous vehicles, leading to insufficient challenges. To overcome the shortcoming of static evaluation methods, this paper proposes a novel method to generate adversarial environments with deep reinforcement learning, and to cluster them with a nonparametric Bayesian approach. As a representative task of autonomous driving, lane-change is used to demonstrate the superiority of the proposed method. First, two lane-change models are separately developed by a rule-based method and a learning-based method, waiting for evaluation and comparison. Next, adversarial environments are generated by training surrounding interactive vehicles with deep reinforcement learning for local optimal ensembles. Then, a nonparametric Bayesian approach is utilized to cluster the adversarial policies of the interactive vehicles. Finally, the adversarial environment patterns are illustrated and the performances of two lane-change models are evaluated and compared. The simulation results indicate that both models perform significantly worse in adversarial environments than in naturalistic environments, with plenty of weaknesses successfully extracted in a few tests.

Index Terms—autonomous vehicle, vehicle evaluation, reinforcement learning, unsupervised learning.

I. INTRODUCTION

AUTONOMOUS vehicles are commonly believed to be a promising approach to eliminate traffic accidents in future transportation because they can prevent unreasonable behaviors of human drivers that could lead to fatal crashes [1]. However, how to develop a safe autonomous driving system in complex environments is still an open problem [2].

One essential procedure to improve the safety of autonomous vehicles is to conduct systematic evaluation before their deployment. The most popular approach is a data-based method called Naturalistic Field Operational Tests (N-FOT) [3]. The idea of this method is to test autonomous vehicles in naturalistic traffic environments, which are constructed by the data collected by sensor-equipped vehicles over a long time [4].

One problem with this method is inefficiency because of the rareness of risky scenarios in naturalistic environments. According to NHTSA, there were 6,064,000 police-reported motor vehicle traffic crashes and 29,989 fatal crashes in USA in 2014, while the total distances that the vehicles traveled was 3,025,656 million miles [5], which means the average distance is 0.50 million miles for each crash and 100.90 million miles for each fatal crash. The rareness of risky events make the evaluation procedure extremely slow even in simulation. Zhao et al. [6] introduced importance sampling techniques with the cross-entropy method to accelerate the evaluation procedure in lane-change scenarios. While maintaining the accuracy, the evaluation is 2,000 to 20,000 times faster than the naturalistic driving tests in simulation.

Another issue of N-FOT is that it is static, which means that the testing environment can not evolve based on the behavior of the tested vehicles. There is no feedback loop for the evaluation, which makes it inefficient and costly. Intuitively, the evaluate would be much more efficient if there is an adversarial agent that can find the weakness of the tested vehicles based on their behaviors, and guide the change of testing environment to be more challenging, adaptively. In this way, the evaluation loop is built, and risky scenarios can be generated directly, which would cost a lot of time and money by N-FOT methods to find.

Recently, deep reinforcement learning has been used to generate evolving adversaries. By interacting with the ego vehicle, adversaries can find an optimal policy to achieve the highest discounted cumulative rewards, which is often set to be the inverse number of that of the tested vehicles to make it a fully-competitive zero-sum game. Pinto et al. [7] proposed Robust Adversarial Reinforcement Learning (RARL) to train an optimal adversarial agent for modeling disturbances, the policies of the protagonist and the adversary are trained following alternating procedure till convergence. However, system disturbances are limited and not able to represent the interactions of vehicles, which corresponds to a much larger state space. Bansal et al. [8] suggested that sufficient complexity of the environment for training is required for a highly capable agent. With adversarial reinforcement learning and self-play, the agents learned a wide variety of complex and interesting skills in 3D physically simulated environments.

Former methods using adversarial reinforcement learning to develop adversaries focus on finding the globally optimal policy to challenge the tested vehicles in the best way, which is pretty time consuming to train and hard to converge, especially in continuous state spaces. On the contrary, many reinforcement learning algorithms guarantee to converge to local optumums under some requirements [9], [10]. For the evaluation of autonomous vehicles, a variety of risky patterns is desired to test the robustness of the self-driving agent, while the riskiest pattern is not a necessity. Also, the optimal agent is...
Fig. 1. Comparison between current state methods and the proposed method for the evaluation of autonomous vehicles. In the proposed method, the tested vehicle is included in the evaluation loop, while the adversarial agents can generate risky environments based on the behaviors of the tested vehicles. The weaknesses of the tested vehicle are extracted by clustering adversarial environments.

probably to be found with various initializations. Thus, we care more about diversity rather than optimality when generating adversarial environments. In this paper, the goal is achieved by performing ensemble reinforcement learning with random initializations and no exploration, which aims to collect local optimums of adversarial policies.

There are other methods to generate adversarial environments by adaptively finding the most risk distribution of environment parameters [11], [12]. However, the parameters of environment are fixed in each episode, which is not an intelligent agent with interaction abilities.

To analyze the weaknesses of the tested autonomous vehicle, it is a good way to cluster adversarial environments. Since it is not possible to get the number of potential local optimums, a nonparametric Bayesian approach is used to cluster the local optimums with unsupervised learning. However, a direct clustering of policies is not trivial, since they are approximated by neural networks with different functional regions. Thus, clustering will be done based on the stationary state distributions.

The comparison between current static evaluation methods and the proposed method is shown in Fig. 1. The main contribution of this paper is that we proposed a novel method to evaluate autonomous vehicles by including the tested vehicle in the evaluation loop and generating adversarial environments based on the behaviors of the tested vehicles, which is a weakness-aiming and evolving evaluation method. With this method, the evaluation of autonomous vehicles can become much more efficient, comprehensive and objective than the current static methods.

The rest of the paper is organized as follows. Section II will introduce the testing scenarios and terms used in the paper. Section III will demonstrate how to generate adversarial environments with ensemble reinforcement learning, and cluster them with DP-Means. In Section IV, the adversarial environments will be visualized, and the tested lane-change models will be evaluated.

II. LANE-CHANGE SCENARIO

The lane-change scenario is used to show the benefit of the adversarial evaluation method. Lane change is regarded as a challenging task for autonomous driving. According to NHTSA, in the US, there are 610,000 reported lane-change crashes, leading to 60,000 injuries annually [13]. One typical scene is shown in Fig. 2. In this scenario, there are three surrounding vehicles namely follow vehicle, leader vehicle and target vehicle, and an ego vehicle trying to make a left lane change. There can be many risky patterns in this scenario caused by different surrounding vehicles, which can be directly generated and clustered by training surrounding vehicles as adversarial agents but are difficult to extract from naturalistic driving data. This section will first define the environment settings, then develop the lane-change models for the ego vehicle with both rule-based and learning-based algorithms. The developed lane-change models can perform very well in the lane-change task under naturalistic environments. They will serve as autonomous driving system examples to be evaluated in the later sections.

A. Environment Settings Based on Naturalistic Data

To develop lane-change models for the ego vehicle, we must first be able to model naturalistic traffic in the environment and the driving behaviors of the three surrounding vehicles. In this paper, the initial conditions of the scenario (e.g., relative distances between vehicles, velocities of vehicles) are decided based on naturalistic traffic data. The surrounding vehicles are controlled by a widely used Intelligent Driver Model (IDM) [14].

1) Scenario Settings: Inspired by [6], the data used in this paper is from the Safety Pilot Model Deployment database [15]. The SPMD program recorded naturalistic data in Ann Arbor, Michigan with 2,842 equipped vehicles for more than 2 years. The MobilEye camera installed on the vehicles will provide the relative distance between the ego vehicle and the front vehicle. The relative velocity can be calculated based on temporal difference. The velocities as well as distances between the four vehicles will be modeled based on the SPMD dataset.

The initial longitudinal distance between the ego vehicle and the leader vehicle $x_{leader}$ is sampled from the empirical distribution ranging from 5 m to 50 m. The relative velocity can be calculated based on temporal difference. The velocities as well as distances between the four vehicles will be modeled based on the SPMD dataset.

The initial longitudinal distance between the ego vehicle and the leader vehicle $x_{leader}$ is sampled from the empirical distribution from SPMD dataset ranging from 5 m to 50 m. The longitudinal distance between the ego vehicle and the tar-
get vehicle \(x_{\text{target-follower}}\) is following the same distribution. Since the longitudinal distance between the ego vehicle and the follow vehicle \(x_{\text{follow}}\) is not in the dataset, it is set to follow a Gaussian distribution: \(x_{\text{follow}} \sim N(\mu_x, \sigma_x^2)\), where \(\mu_x\) is 0 m and \(\sigma_x\) is 5 m. The longitudinal distance between the ego vehicle and the target vehicle \(x_{\text{target}}\) is then equal to \(x_{\text{follow}} + x_{\text{target-follower}}\).

The initial velocity of the four vehicles in the scenario \(v_{\text{ego}}\) is sampled from a Gaussian distribution: \(v_{\text{ego}} \sim N(\mu_v, \sigma_v^2)\), where \(\mu_v\) is set to be 10 m/s, and \(\sigma_v\) is 4 m/s.

Based on above initial settings, the lane change can be regarded as an episodic game. The ego vehicle is trying to make a left lane change within limit distance \(x_{\text{lim}} = 300\) m and time \(t_{\text{lim}} = 30\) s. The lane change is judged to be successful if the whole body of the ego vehicle is in the left lane and the yaw angle error is less than 30°.

2) Surrounding Vehicle Model: The surrounding vehicles will stay in their current lanes and are controlled based on the Intelligent Driver Model (IDM) [14]. With IDM, the acceleration of the surrounding vehicle \(\alpha\) is

\[
\frac{dv_{\alpha}}{dt} = \alpha \left(1 - \frac{v_{\alpha}}{v_0}\right)^\delta - \left(\frac{sv_{\alpha}}{s_{\alpha}}\right)^2,
\]

where

\[
s^*(v_{\alpha}, \Delta v_{\alpha}) = s_0 + v_{\alpha} T - \frac{v_{\alpha} \Delta v_{\alpha}}{2 \sqrt{ab}}.
\]

The description and values of parameters used for IDM in this paper are shown in Table 1.

### Table 1: Parameters of IDM

| Parameter | Description       | Value |
|-----------|-------------------|-------|
| \(v_0\)   | Desired velocity  | 10 m/s|
| \(T\)     | Safe time headway | 1.5 s |
| \(a\)     | Maximum acceleration | 1 m/s²|
| \(b\)     | Comfortable Deceleration | 1.67 m/s²|
| \(\delta\) | Acceleration exponent | 4     |
| \(s_0\)   | Minimum distance  | 2 m   |

B. Ego Vehicle Lane-Change Model

Two lane-change models are developed in this paper for evaluation. One is a traditional rule-based model using gap acceptance concepts, the other is trained by reinforcement learning. In this section, only the architectures of the two lane-change models are introduced. The performances of them in both naturalistic and adversarial environments are later shown in Section IV.

1) Gap Acceptance Model for Lane Change: Gap acceptance is an important concept in most lane change models [16], [17]. Before executing a lane change, the driver assesses the positions and speeds of the target vehicle and the follow vehicle in the target lane (see Fig 3) and decides whether the gap between them is sufficient for the lane change behavior. The front spacing between the ego vehicle and the leader vehicle is also critical for avoiding a front crash. The gap acceptance lane change model is implemented based on [17], which is extracted from naturalistic data.

2) Reinforcement Learning Lane-Change Model: Reinforcement learning has become a powerful tool for the development of autonomous vehicles. However, end-to-end reinforcement learning can take a relatively long time to converge to the optimal policy. For autonomous driving, it is a good approach to develop a hierarchical framework [18], where reinforcement learning is only used in the high-level decision-making part, while the motion planning and control part is developed by hard-code. Specifically, in this paper, deep Q-learning [19] is used in the decision-making part to decide whether or not to start a lane change. The desired lane change trajectory is generated by an optimal lattice planner [20], while the longitudinal and lateral control of the vehicle is achieved by a model predictive controller [21].

III. Adversarial Evaluation

This paper proposes a method to generate adversarial environments \(P_{\text{adv}}\) and cluster them, which are used to evaluate autonomous driving strategies. The formulation is introduced in this section. Specifically, in a lane-change scenario, an adversarial environment means that, in the environment, the surrounding vehicles (including the follow vehicle, the front vehicle and the target vehicle) are trying to challenge the ego vehicle and prevent it from a successful lane change. This is an evolving method since the adversaries will adapt based on the behaviors of the tested ego vehicle. The output of this method is the patterns of the challenging adversarial environments, which can reflect the weaknesses of the tested ego vehicle. The proposed adversarial evaluation method can greatly help for the evaluation and further improvement of autonomous driving systems.

A. Adversarial Environment Generation

Autonomous driving systems are usually developed and evaluated based on naturalistic data. However, the database is always limited in two ways. First, the capacity of the database can never be infinitely large, which means it won’t be able to generate every scenario that could happen in the real world. Second, risky events are rare in the database [6]. For these two reasons, the autonomous driving systems developed based on the database are likely to be overfitted, which makes them unpredictable under scenarios that never appear in the database. Thus, it is not sufficient to just test them in environments generated by data \(P_{\text{data}}\). They should be tested in the real-world environment \(P_{\text{real}}\), which is unfeasible. In this section, we propose a method to learn to generate adversarial environments \(P_{\text{adv}}\) for the tested vehicle, which is directly aimed to find the weaknesses of it.

The lane-change scenario can be regarded as a two player Markov game expressed as a tuple \((S, A_1, A_2, P, r_1, r_2, \gamma, s_0)\), where \(S\) is the state space initialized at \(s_0\); \(A_1\) is the action space for the ego vehicle, and \(A_2\) is the action space for the adversaries, which are the three surrounding vehicles in the lane-change scenario; \(P : S \times A_1 \times A_2 \times S \rightarrow \mathbb{R}\) is the state transition probability; \(r_1 : S \times A_1 \times A_2 \rightarrow \mathbb{R}\) and \(r_2 : S \times A_1 \times A_2 \rightarrow \mathbb{R}\) are the immediate rewards for the ego and surrounding vehicles. In a Markov game, each agent \(i\) aims to
maximize its own total expected return \( R_i = \sum_{t=0}^{T} \gamma^t r_i^t \) with a policy \( \pi_i : S \rightarrow A_i \), where \( T \) is the time horizon.

For adversaries, the ego vehicle can also be regarded as a part of the environment, and then the Markov game degrades to a Markov decision process (MDP). Here, we consider the multi-adversary as one agent, which assumes that each adversary has a perfect observation and they are fully-cooperative to challenge the ego vehicle. This assumption is achievable in intelligent transportation systems thanks to V2X infrastructures. This assumption can simplify the problem and can better generate risky environments.

Reinforcement learning is a powerful tool to solve the MDP. Specifically, deep deterministic policy gradient (DDPG) is usually used to solve MDPs with continuous action space [22]. DDPG is a reinforcement learning method with actor-critic architecture. The actor \( \mu(s|\theta^\mu) \) is a parameterized function that specifies the current policy which deterministically maps states to a specified action. The critic \( Q(s,a) \) is the action-value function which describes the expected return after taking an action \( a_t \) in state \( s_t \) an following the policy \( \mu \) afterwards:

\[
Q^\mu(s_t, a_t) = \mathbb{E}_{r_{t+1}, s_{t+1} \sim R, a_t \sim \mu} \left[ R_t | s_t, a_t \right].
\]

The critic is updated following the Q-learning [23] which is based on Bellman equation. Consider the function approximator parameterized by \( \theta^Q \), the critic is optimized by minimizing the loss:

\[
\mathcal{L}(\theta^Q) = \mathbb{E}_{s_t \sim \rho, a_t \sim \beta, r_t \sim R} \left[ (Q^\mu(s_t, a_t|\theta^Q) - y_t)^2 \right],
\]

where

\[
y_t = r(s_t, a_t) + \gamma Q \left( s_{t+1}, \mu(s_{t+1})|\theta^Q \right),
\]

\( \beta \) is different behavior policy and \( \rho \) represents the state distribution. This indicates that Q-learning is an off-policy algorithm. Thus, experience replay buffer can be used to eliminate the time correlation and improve the sample efficiency [19].

The actor is updated by following the policy gradient [24]:

\[
\nabla_{\theta^\mu} J \approx \mathbb{E}_{s_t \sim \rho} \left[ \nabla_a Q \left( s, a|\theta^Q \right) | s=s_t, a=\mu(s_t) \right] \nabla^\mu \mu(s|\theta^\mu) | s=s_t,
\]

To make the update iterations stable, a copy of the actor and the critic network is created:

\[
\mu'(s|\theta'^\mu) \text{ and } Q'(s,a|\theta'^Q).
\]

The parameters of these target networks are slowly updated to track the learned models: \( \theta' = \tau \theta + (1-\tau) \theta' \).

\( r(s_t, a_t) \) is the immediate reward when taking action \( a_t \) in state \( s_t \), which is essential in this framework, since it determines how the adversaries vehicles will behave. Intuitively, one term of the reward function is \( -r_{goal} \), which indicates that the adversaries in the scenario will try to prevent the ego vehicle from achieving the goal (In the lane-change scenario, the goal of the ego vehicle is to perform a successful lane change within specified time). The other term in the reward function is a penalty for violation of traffic rules \( r_{rule} \), which prevents the adversaries from being irrational, e.g., directly rush to the ego vehicle to make a collision. With the reward function

\[
r_{adv} = -r_{goal} + r_{rule},
\]

the adversarial vehicles in the environment will try to prevent the ego vehicle from achieving the driving goal with behaviors allowed by traffic rules.

The idea of training adversarial environments with reinforcement learning to develop robustness policies has been proposed before [7], [8]. However, these methods focus on finding the optimal policy for the adversaries, which is very time-consuming with broad exploration, and can get only one risky pattern after a long time training, which is inefficient.

Local optimum, a solution that is optimal within a neighboring set of candidate solutions, is where a reinforcement learning agent can easily stuck [25]. Many researchers have studied how to avoid local optimal solutions, and the most effective method is to encourage efficient and directed exploration [26], which is very time-consuming and still lacks a theoretical convergence guarantee.

In this paper, instead of avoiding the local optimum, we embrace it for efficiency and use ensemble models for diversity. For the evaluation of autonomous vehicles, we prefer to find multi-weaknesses of the tested vehicle in a relatively short time, rather than only one weakness which is hard to converge. Based on this intuition, we proposed the ensemble DDPG for local optimums (Algorithm 1-A). Instead of training one agent, we train \( N \) agents with random initializations of the actor and the critic. The exploration is canceled for fast convergence to a local optimum. For each agent, we stop training if a local optimum has been reached, or the cumulative reward of one episode \( \sum_{t=1}^{T} \gamma^t r_t \) has reached some boundary \( c \), which indicates that a challenging environment has been found for the tested ego vehicle (e.g., a responsible collision happened).

**Algorithm 1 Ensemble DDPG for local optimums**

1: **for** adversarial agent **do**
2: randomize the initial parameters for the actor \( \mu(s|\theta^\mu) \) and the critic \( Q(s,a|\theta^Q) \)
3: copy for the target networks \( \mu'(s|\theta'^\mu) \) and \( Q'(s,a|\theta'^Q) \)
4: initialize replay memory \( D \)
5: **while** not Converged **do**
6: update \( \theta^\mu, \theta^Q, \theta'^\mu \) and \( \theta'^Q \) with DDPG algorithm without exploration
7: **end while**
8: save \( \mu(s|\theta^\mu) \)
9: **end for**

**B. Environment clustering**

Adversarial patterns for tested vehicles can be generated by unsupervised clustering of adversaries. However, a direct clustering of the learned adversaries \( \{\mu(s|\theta^\mu)\} \) is infeasible since they’re deep neural networks. Instead, this paper proposes an indirect way to cluster adversarial environments by different state distributions.

First, state distributions \( \{p_i(s)\} \) is approximated by collecting Monte-Carlo simulation memories, where \( p_i(s) \) represents the state distribution of the scenario when the ego vehicle is in the \( i \)th adversarial environment.
Next, since the number of clusters is unknown, a non-parametric method: Dirichlet-Process-Means (DP-Means) [27] is used for the clustering problem. To use DP-Means, we must first find a suitable hyperparameter: λ, which indicates the approximate distance between different clusters. We look for it with a heuristic method suggested in [27]: Given an approximate number of desired clusters k, we first initialize a set T with the mean distribution of \{ρ_i(s)\}. Then, iteratively add the distribution to T which has the maximum distance to T. Repeat this k times and set λ as the maximum distance in the last round. To calculate the distance of two distributions, Jensen-Shannon divergence is used:

\[ JSD(P \parallel Q) = \frac{1}{2}D(P \parallel M) + \frac{1}{2}D(Q \parallel M), \]

where \(M = \frac{1}{2}(P + Q),\) and \(D(\cdot \parallel \cdot)\) is the KL divergence:

\[ D(P \parallel Q) = \int_s p(s) \log \frac{p(s)}{q(s)} ds. \]

Then, with \(λ\) and distance function, we can use DP-Means to cluster the adversarial environments. The whole algorithm of environment clustering is shown in Algorithm [II-B]. After the clustering procedure, adversarial patterns for the tested vehicle can be extracted.

### IV. Simulation

The simulation in this paper is conducted in the CARLA environment [28], see Fig. 3. In the lane-change scenario, there are four vehicles in total: one ego vehicle that is being tested, and three adversarial vehicles namely follow vehicle, leader vehicle, and target vehicle. Simulation is used to demonstrate the effectiveness of the proposed adversarial evaluation approach.

#### A. Ego Vehicle Development

To demonstrate adversarial evaluation, we must first develop the lane-change models for the ego vehicle. As mentioned in [II-B] for comparison, we developed two kinds of lane-change models. One is a rule-based model based on the gap-acceptance concept and is implemented following [17]. The other is a hierarchical lane-change model, whose upper layer is the decision-making part trained using reinforcement learning method DQN [19], middle layer is a lattice planner for trajectory generation, and bottom layer is a model predictive controller [21]. For convenience, we will denote the two lane-change models as \(M_{\text{gap}}\) and \(M_{\text{rl}}\), respectively.

The ego vehicle is developed and trained based on the SPMD database [15]. The surrounding vehicles are controlled by IDM, as mentioned in [II-A].

After parameter tuning and training, both of \(M_{\text{gap}}\) and \(M_{\text{rl}}\) achieved lane-change goal in all 1000 runs of stochastic simulation without a single collision. Thus, they can be regarded as good lane-change models in the proposed naturalistic environment. Next, we will illustrate their performances in adversarial environments.

#### B. Adversarial MDP Setting

As mentioned in Section [III], DDPG is used to train adversarial environments for the ego vehicle. We first introduce the settings of the lane-change MDP and the DDPG agent.

The state space \(S\) of the MDP is a 9-dimension vector space: \([x_{\text{leader}}, x_{\text{follow}}, x_{\text{target}}, v_{\text{leader}}, v_{\text{follow}}, v_{\text{target}}, v_{\text{ego}}, \phi_{\text{ego}}, y_{\text{ego}}]\), where \(x\) denotes the distance between the adversarial vehicle and ego vehicle, \(v\) denotes velocity of the vehicle, \(\phi\) denotes the yaw angle of the ego vehicle, and \(y\) denotes the lateral position of the ego vehicle. To simplify the problem, we assume that perfect state information is shared by every participant.

The adversarial vehicles will stay in the lane they’re currently in, so the action space \(A\) of the adversarial agent is a vector space 3-dimension that decides their longitudinal control action: \([a_{\text{leader}}, a_{\text{follow}}, a_{\text{target}}]\). \(a\) is a float number in the range \([-1, 1]\), where +1 indicates a full-throttle and -1 indicates a full brake. Since DDPG can handle continuous action space, no discretizations are needed.

The reward function of adversaries is \(r_{\text{adv}} = -r_{\text{goal}} + r_{\text{rule}},\) as mentioned in [III-A]. Specifically, in the lane-change sce-
the velocity term is used to encourage the ego vehicle to run faster, and a responsible collision means that the collision happened because of the ego vehicle’s fault. For the adversarial vehicles, $r_{rule} = -100$ if a collision happened because of the adversarial vehicles’ fault otherwise 0. The simulation time-step is set to be 0.1 seconds.

The actor model used in DDPG is a three-layer fully-connected neural network with the number of hidden units: \[64, 64, 32, 1\]. The activation is ReLU for the first two layers and Tanh for the output layer to get a \([-1, 1]\) output. The critic model is a four-layer fully-connected neural network with the number of hidden units: \[64, 64, 32, 1\]. The activation is ReLU for the first three layers and Identity for the output layer.

Other hyperparameters are: buffer size = 10,000, batch size = 128, discount factor $\gamma = 0.99$, learning rate of the actor $\alpha_a = 0.001$, learning rate of the critic $\alpha_c = 0.01$, and soft update rate $\tau = 0.01$.

### C. Adversarial Evaluation

With Algorithm III-A and III-B we can train multi-adversaries and get adversarial policies and state distributions with $\mathcal{M}_{gap}$ and $\mathcal{M}_{rl}$ separately. In this paper, we trained 100 adversarial environments for each lane-change model. The clustering results, as well as simulation returns (discounted cumulative rewards) of adversarial agents in the same cluster, are shown in Fig. 4. It can be noticed that the mean value of the returns of adversarial agents developed for the gap-acceptance lane-change model $\mathcal{M}_{gap}$ is much lower than that developed for the reinforcement learning model $\mathcal{M}_{rl}$. Most of the returns with the gap-acceptance model concentrate near 0, which indicates that neither a collision nor a successful lane change is achieved. However, with the reinforcement learning model, most returns are near 100, indicating that many collisions happened during the testing. Based on this, we can state that among the two lane-change models developed in [V-A], the gap acceptance model $\mathcal{M}_{gap}$ is safer when facing risky scenarios that were never seen before.

The robustness of the two models can also be compared by goal-achievement rate and collision rate, as shown in Table II. In fact, both of the two lane-change models can hardly make a successful lane change in adversarial environments (success rate is less than 10%). However, the gap-acceptance model $\mathcal{M}_{gap}$ has a collision rate of only 3%, which is 90% for the reinforcement learning model $\mathcal{M}_{rl}$. In fact, this is reasonable because we injected human knowledge when designing and developing $\mathcal{M}_{gap}$. The gap-acceptance rule is what we believe human drivers follow when trying to make a lane change. But for $\mathcal{M}_{rl}$, we only fed limited naturalistic data and control the surrounding vehicles with a static IDM. When the adversarial agent generates states that never appear during the training process, the $\mathcal{M}_{rl}$ will act unpredictably. Note that, for both lane-change models, the success rate is 100% without any collision in the training naturalistic environment, which proves that the generated adversaries raise truly big challenges for the lane-change models.

Next, we will visualize some adversarial patterns for both lane-change models. For $\mathcal{M}_{gap}$, as shown in Fig. 5, the adversaries in most patterns are trying to prevent a successful lane change by blocking in front of the ego vehicle. The blocking is usually done by the leader vehicle and the target vehicle, while the follow vehicle is trying to minimize the lane-change gap in most times. In general, the gap acceptance lane-change model is able to brake in time and avoid collisions in most experiments, but the success rate of lane change is pretty low, as mentioned in Table II.

For $\mathcal{M}_{rl}$, the adversarial scenarios are shown in Fig. 6. The ego vehicle could be too aggressive to collide with

\[ r_{goal} = \begin{cases} 
100 & \text{if lane-change is finished} \\
-100 & \text{if a responsible collision happened} \\
0.01 r_{ego} & \text{otherwise} 
\end{cases} \]

![Gap acceptance model](image1.png)

![Reinforcement learning model](image2.png)

**Fig. 4.** Clustering results of different lane-change models. (a): The gap-acceptance model has 16 adversarial environment clusters. The mean return of adversaries is -7.26. (b): The reinforcement learning model has 18 adversarial environment clusters. The mean return of adversaries is 82.26.

| Environment   | Model     | Avg. return | Collision rate | Success rate |
|---------------|-----------|-------------|----------------|--------------|
| Adversarial   | $\mathcal{M}_{gap}$ | -7.26       | 3%             | 7%           |
|               | $\mathcal{M}_{rl}$ | 82.26       | 90%            | 6%           |
| Training      | $\mathcal{M}_{gap}$ | -100        | 0%             | 100%         |
|               | $\mathcal{M}_{rl}$ | -100        | 0%             | 100%         |
Fig. 5. Adversarial lane-change scenarios developed for $M_{\text{gap}}$. Trajectories of vehicles are scattered with color: red for the ego vehicle, orange for the follow vehicle, green for the leader vehicle, and blue for the target vehicle. Each scenario is selected from one of the adversarial clusters. In (a), the green dot at the up-left corner indicates that a left lane-change is successfully done. However, in the other 15 scenarios, neither a successful lane change nor a collision takes place. The ego vehicle is blocked by the leader vehicle in (d)(g)(h)(i)(m)(p), by the target vehicle in (e)(f)(j)(k)(l)(n)(o), and by both of them in (b)(c). In general, the gap acceptance lane-change model is able to make a brake in time and avoid collisions in most experiments, but the success rate of lane change is low (%7).

Fig. 6. Adversarial lane-change scenarios developed for $M_{\text{rl}}$. Trajectories of vehicles are scattered with color: red for the ego vehicle, orange for the follow vehicle, green for the leader vehicle, and blue for the target vehicle. Each scenario is selected from one of the adversarial clusters. The red crosses in (a)(b)(c)(d)(e)(k)(i)(n)(o) represent responsible collisions of the ego vehicles in corresponding adversarial scenarios. The ego vehicle could be too aggressive to collide with the target vehicle (in (a)(b)(d)(i)(n)(o)) or the leader vehicle (in (c)(e)(k)), and could also be too conservative to just stop with safe lane-change space in (m). Successful lane change is found in (f) and (p).
the target vehicle or the leader vehicle, and could also be too conservative to just stop with safe lane-change space. Specifically, in Fig. 6(m), the ego vehicle stays in the same place even when the leader vehicle and the target vehicle have been far away. The behavior of the ego vehicle can be explained by the state mismatch between the generated adversarial environment and the training environment: the ego vehicle does not know what to do in the states that never appear before. This scenario is a good example to show the advantage of the proposed adversarial evaluation method: normal behaviors of adversarial vehicles can lead to the unnormal behavior of the ego vehicle. This is infeasible by rule-based searching and vanilla reinforcement learning methods since the adversarial environment is only a local optimal solution.

Comparing Fig. 5 and Fig. 6, we can notice that the adversarial patterns for $M_{gap}$ are more diverse than those for $M_{rl}$. The reason is that $M_{gap}$ is more robust, and most adversaries can only find behaviors to stop the ego vehicle by blocking it in front. However, for $M_{rl}$, things are different because many kinds of actions of adversarial vehicles can lead to a responsible collision for the ego vehicle. We can illustrate this by comparing Fig. 5(k) and Fig. 6(o), where the behaviors of adversarial vehicles in the two scenarios are pretty similar but got different results: $M_{gap}$ can make an in-time brake while $M_{rl}$ is not able to prevent a collision.

We can also notice that the return variance of the 5th cluster for the reinforcement learning model is high, as shown in Fig. 5(b). To find out why, we visualize the two scenarios in the same cluster, as shown in Fig. 7. It turns out that the clustering result is correct since the state distributions of the two scenarios are pretty similar. The high variance of return comes out of the different simulation results of the two adversarial agents. In Fig. 7(a), the lane change is successful; while in Fig. 7(b), the ego vehicle collided with the target vehicle because of a sudden brake.

![Figure 7](image-url)

Fig. 7. Adversarial lane-change scenarios developed for $M_{rl}$ in the 5th cluster. The state distributions of the two scenarios are pretty similar. However, the simulation result is different for the different actions of the target vehicle.

V. CONCLUSION

In this paper, we propose an adversarial evaluation method for autonomous vehicles. Adversarial environments are generated with ensemble deep reinforcement learning and clustered with a nonparametric Bayesian method based on the state distributions. The simulation results show that the adversarial environments can significantly reduce the success rate and increase the collision rate for both rule-based and learning-based lane-change models. Thus, this method can be used as a supplementary for the current evaluation of autonomous vehicles.

A promising future direction is to improve the tested autonomous vehicle with extracted weaknesses from adversarial evaluation. Multi-agent games can also be introduced in more complex scenarios, e.g., intersections. Hidden states and observation errors can be assumed to make the problem a partially observable MDP.

REFERENCES

[1] S. A. Bagloee, M. Tavana, M. Asadi, and T. Oliver, “Autonomous vehicles: challenges, opportunities, and future implications for transportation policies,” Journal of modern transportation, vol. 24, no. 4, pp. 284–303, 2016.
[2] P. Koopman and M. Wagner, “Autonomous vehicle safety: An interdisciplinary challenge,” IEEE Intelligent Transportation Systems Magazine, vol. 9, no. 1, pp. 90–96, 2017.
[3] FESTA-Consortium et al., “Festa handbook version 2 deliverable 16. 4 of the field operational test support action,” Brussels: European Commission, 2008.
[4] M. Aust, “Evaluation process for active safety functions: Addressing key challenges in functional, formative evaluation of advanced driver assistance systems,” Chalmers University of Technology, pp. 1243–1262, 2012.
[5] NHTSA, “Traffic safety facts 2014,” 2014.
[6] D. Zhao, H. Lam, H. Peng, S. Bao, D. J. LeBlanc, K. Nobukawa, and C. S. Pan, “Accelerated evaluation of automated vehicles safety in lane-change scenarios based on importance sampling techniques,” IEEE transactions on intelligent transportation systems, vol. 18, no. 3, pp. 595–607, 2016.
[7] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, “Robust adversarial reinforcement learning,” in Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017, pp. 2817–2826.
[8] T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, and I. Mordatch, “Emergent complexity via multi-agent competition,” arXiv preprint arXiv:1710.03748, 2017.
[9] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” in Advances in neural information processing systems, 2000, pp. 1057–1063.
[10] L. Baird, “Residual algorithms: Reinforcement learning with function approximation,” in Machine Learning Proceedings 1995. Elsevier, 1995, pp. 30–37.
[11] A. Rajeswaran, S. Ghotra, B. Ravindran, and S. Levine, “Epopt: Learning robust neural network policies using model ensembles,” arXiv preprint arXiv:1610.01283, 2016.
[12] K. A. Ciosek and S. Whiteson, “Offer: Off-environment reinforcement learning,” in Thirty-First AAAI Conference on Artificial Intelligence, 2017.
[13] G. Fitch, S. Lee, S. Krause, J. Hankey, J. Sudweeks, and T. Dingus, “Analyses of lane-change crashes and near-crashes,” US Department of Transportation, National Highway Traffic Safety Administration, 2009.
[14] M. Treiber, A. Hennecke, and D. Helbing, “Congested traffic states in empirical observations and microscopic simulations,” Physical review E, vol. 62, no. 2, p. 1805, 2000.
[15] D. Bezvina and J. Sayer, “Safety pilot model deployment: Test conductor team report,” Report No. DOT HS, vol. 812, p. 171, 2014.
[16] K. Ahmed, M. Ben-Akiva, H. Koutsopoulos, and R. Mishalani, “Models of freeway lane changing and gap acceptance behavior,” Transportation and traffic theory, vol. 13, pp. 501–515, 1996.
[17] T. Toledo, H. N. Koutsopoulos, and M. E. Ben-Akiva, “Modeling integrated lane-changing behavior,” Transportation Research Record, vol. 1857, no. 1, pp. 30–38, 2003.

[18] M. Müller, A. Dosovitskiy, B. Ghanem, and V. Koltun, “Driving policy transfer via modularity and abstraction,” arXiv preprint arXiv:1804.09364, 2018.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski et al., “Human-level control through deep reinforcement learning,” Nature, vol. 518, no. 7540, p. 529, 2015.

[20] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, “Optimal trajectory generation for dynamic street scenarios in a frenet frame,” in 2010 IEEE International Conference on Robotics and Automation. IEEE, 2010, pp. 987–993.

[21] C. E. Garcia, D. M. Prett, and M. Morari, “Model predictive control: theory and practice a survey,” Automatica, vol. 25, no. 3, pp. 335–348, 1989.

[22] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” arXiv preprint arXiv:1509.02971, 2015.

[23] C. J. Watkins and P. Dayan, “Q-learning,” Machine learning, vol. 8, no. 3-4, pp. 279–292, 1992.

[24] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic policy gradient algorithms,” 2014.

[25] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction. MIT press, 2018.

[26] E. Conti, V. Madhavan, F. P. Such, J. Lehman, K. Stanley, and J. Clune, “Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents,” in Advances in Neural Information Processing Systems, 2018, pp. 5027–5038.

[27] B. Kulis and M. I. Jordan, “Revisiting k-means: New algorithms via bayesian nonparametrics,” arXiv preprint arXiv:1111.0352, 2011.

[28] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “Carla: An open urban driving simulator,” arXiv preprint arXiv:1711.03938, 2017.