Adversarial Deep Reinforcement Learning for Improving the Robustness of Multi-agent Autonomous Driving Policies

Aizaz Sharif
Simula Research Laboratory
Oslo, Norway
aizaz@simula.no

Dusica Marijan
Simula Research Laboratory
Oslo, Norway
dusica@simula.no

***Abstract*—Autonomous cars are well known for being vulnerable to adversarial attacks that can compromise the safety of the car and pose danger to other road users. To effectively defend against adversaries, it is required to not only test autonomous cars for finding driving errors but to improve the robustness of the cars to these errors. To this end, in this paper, we propose a two-step methodology for autonomous cars that consists of (i) finding failure states in autonomous cars by training the adversarial driving agent, and (ii) improving the robustness of autonomous cars by retraining them with effective adversarial inputs. Our methodology supports testing autonomous cars in a multi-agent environment, where we train and compare adversarial car policy on two custom reward functions to test the driving control decision of autonomous cars. We run experiments in a vision-based high-fidelity urban driving simulated environment. Our results show that adversarial testing can be used for finding erroneous autonomous driving behavior, followed by adversarial training for improving the robustness of deep reinforcement learning-based autonomous driving policies. We demonstrate that the autonomous cars retrained using the effective adversarial inputs noticeably increase the performance of their driving policies in terms of reduced collision and offroad steering errors.**

***Index Terms*—autonomous car, self-driving car, autonomous driving, multi-agent, adversarial testing, AI testing, simulation testing, deep reinforcement learning, robustness**

I. Introduction

Autonomous cars (ACs, also known as self-driving cars) are complex technologies that are prone to failures [1]. ACs integrate deep learning based software, which is known to be difficult to validate [2] [3]; yet, testing and validating ACs is indispensable for their deployment in practice [4] [5] [6]. While there has been progress made by researchers on testing AI-based models [7] [8], testing of ACs is another complex area to tackle, due to several reasons.

First, a lot of research has been proposed on scenario and test case generation [9] [10] [11] [12] [13] and input validation [14] [15] [16] [17] [18] for testing and validating AC driving models. While such approaches can expose failures in ACs, they are only focused on error *detection* and not *correction*. There is limited research work on analyzing the failed AC driving systems for understanding out-of-distribution scenes and edge cases that need to be induced in the training of AC models. Therefore, there is a need for a comprehensive AC testing methodology able to not only discover errors in AC driving models but also improve the performance of failing AC models given the same attacking inputs. This would help in overcoming existing errors and improving robustness in the evolving AI-based AC systems.

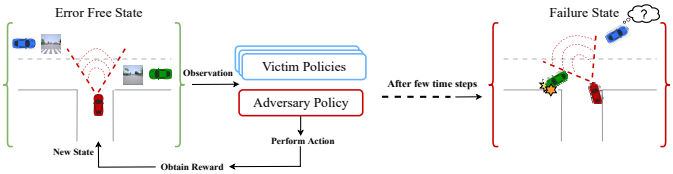


Fig. 1: Illustration of the first step of the MAD-ARL framework, where adversary is driving AC under test into failure states.

Second, existing research efforts often address AC testing in simplistic evaluation scenarios. For example, AC testing is often considered as a *single-agent* problem, where only one car is taken as a system under test (SUT) [9] [19] [20] [17] [18]. While a single-agent self-driving environment still has open challenges, there is a need to advance and test a multi-agent self-driving, as it represents a more realistic environment. In the near future, multiple ACs will co-exist on the road, and the more such cars start interacting with each other, as well as with human drivers, the more complex their testing becomes [21]. Another type of simplistic scenario is validating ACs in *lane keeping* [11] [22] and *mixed traffic systems* [23] simulated environments. Existing industrial-grade AC testing uses vision-based end-to-end driving systems where ACs are tested on the high dimensional stream of inputs in a partially observable environment, unlike in lane-keeping simulated environments where such scenarios rely on low dimensional inputs and a fully observable environment for making driving decisions. Moreover, multi-agent driving environments consider cooperative and dependent driving agents, while the majority of AC research considers driving agents as independent and non-communicating competitive agents. Recent work [23] made progress toward testing connected and automated vehicles by

injecting adversarial noise in a mixed-traffic-based network environment. However, this work is of limited practical use as it does not consider multiple non-communicating AC agents. Another example of a simplistic evaluation scenario is testing of *rule-based* driving systems in a simulated environment [24] [25]. Such driving systems are based on deterministic logic and do not scale to corner cases, unlike ACs based on deep learning systems. One more type of simplistic evaluation scenario is *offline testing* of deep learning based AC models [26]. Offline testing involves validating the performance and control decisions of a vision-based AC model by only using datasets of urban simulated driving scenarios, collected by humans. Even though offline testing contributes to the machine learning and autonomous driving (AD) community, it is very restricted to a single-agent setting (offline testing). Adding multi-agent actions induced by other driving policies in a non-stationary environment [27] is a very crucial and new area to tackle in the AC driving and testing community.

Third, deep reinforcement learning (DRL) algorithms are extensively used in training vision-based safe AC models in urban driving environments [28] [29] [30] [31] [32]. One way to test their driving behavior is using adversarial RL (ARL) since DRL is proven to be vulnerable multiple times against adversarial attacks. Existing research suggests that ARL-based agents can be effective in exposing vulnerabilities of DRL-based agents in a blackbox manner [33]. However, the idea has been explored in a simplistic driving environment [24] [23]. In our work, we make use of ARL for discovering effective attacking inputs that we further use to improve the robustness of DRL-based AC policies in a complex non-communicating vision-based urban driving environment. Specifically, we introduce ARL as part of a driving simulation in order to add adversarial actions against the AC policies under test. By doing so, we show not only find failure scenarios of the DRL-based ACs interacting with the adversarial drivers but to leverage effective adversarial actions to improve the AC driving robustness.

To address these three challenges, in this paper we propose a framework **'M**ulti-**A**gent **D**riving with **A**dversarial **R**einforcement **L**earning **MAD-ARL'** which is a novel approach for improving the robustness of non-communicating multi-agent ACs using an adversarial agent in an urban driving scenario. In the first step, we train an adversarial agent car that aims to create *natural observations* that are adversarial for the ACs under test. Using the trained adversary, we test multi-agent policies of ACs under test, with the goal of exposing faults in the cars' driving policies as illustrated in Figure 1. In the second step, we retrain the ACs under test with the adversarial policies to defend against adversarial attacks. The results are compared with the baseline autonomous models (adversary-free trained policies) to evaluate the effectiveness of the retraining strategy as a defense against adversarial attacks. In our experiments, we demonstrate that a trained adversarial player can improve the robustness of more than one vision-based AC policy in terms of fewer collisions and offroad steering accidents. The main idea is to use adversarial examples beyond testing purposes to improve the robustness of ACs, since adversarial attacks are usually not considered when AC models are being trained in urban driving scenarios.

The key research contributions in this paper are:

1) Proposing a two-step methodology for finding failure scenarios in ACs and improving the robustness of ACs given the effective test scenarios.
2) Introducing a novel DRL framework for testing and improving driving policies of independent non-communicating agents in a multi-agent AC environment. The implementation of the framework is open [1] and reusable, which supports the reproducibility of research in the AD domain.
3) Designing an RL-based adversarial agent that can be generalized for testing more than one AC driving policy by only training against a single victim AC.
4) Designing an adversary that can effectively drive an AC into error states by creating natural (i.e. realistic) observations for the AC's driving policies, without whitebox access to its input state.
5) Experimentally demonstrating that retraining DRL-based AC driving policies using adversarial driving models can be an effective defense against adversarial attacks.

II. Related Work

The majority of related work has focused on generating test scenarios for discovering errors in ACs and adversarial testing of ACs. The main limitations of these works are the lack of focus on improving the robustness of ACs once errors are discovered, as well as simplistic evaluation conditions. Next, we summarize the main approaches, discussing their benefits and limitations.

A. Test Scenario Generation for AC

Authors in [14] use GANs to generate synthetic images to validate the driving robustness of deep learning-based autonomous driving systems. They also use metamorphic testing to check the consistency of the model outputs against different types of synthetic images. Another close work [15] proposes a systematic testing tool for evaluating DNN-based AC models. They do so by generating test cases using real-world conditions such as rain and lightning conditions. They perform DNN logic coverage by adding transformations to test inputs within Udacity self-driving car challenge simulator [34]. Similar to [14], they also use domain-specific metamorphic relations to find fault behaviors of DNN. While the proposed work has achieved great results, it is limited in way that the driving scenarios are only tested in a single-agent environment. Also, both [15] and [14] would benefit from adding the same synthetic images in the retraining of the tested DNN AC models to compare the robustness with the baseline models.

The authors in [11] use a search-based testing technique to automatically create challenging virtual scenarios for testing

[1] https://github.com/T3AS/MAD-ARL

self-driving cars. These scenarios are used to test AI driving models such as DeepDriving [35] to perform systematic testing of lane-keeping systems. While the work contributes to having more complex evaluation scenarios, it does not address the problem of AC testing in a realistic multi-agent driving environment. Authors in [10] use Bayesian optimization for test case generation for ACs. The proposed work learns parameters using the system's output to create test case scenarios that lead AC into failure states. While the authors are able to identify test cases for complex black boxes like autonomous systems, the work lacks complex driving scenarios with more than one AC in the same environment for testing purposes.

A thorough case study by authors in [26] performs a comparative study of the pros and cons of testing deep learning-based AC models in offline datasets versus online simulation testing. Offline testing focuses on prediction errors against the dataset, while online testing looks for safety violations within driving scenarios. The tests are performed using a pretrained Udacity car simulator driving model. As a limitation, the work needs an extension of multi-agent testing configurations within online and offline driving scenarios to observe which method will be more beneficial for multi-agent ACs testing. Authors in [36] use OpenStreetMap traffic simulator SUMO to suggest a workflow for generating a collection of challenging and safety-critical test scenarios for the safety validation of motion planning algorithms in automated vehicles. As a limitation, the work requires multi-agent AC testing to take advantage of the publicly available generated scenarios.

Authors in [9] propose an automated fuzzing framework to produce AC safety violation driving scenarios. Using the industrial-grade autonomous driving platform Baidu Apollo, they use domain knowledge of vehicle dynamics and genetic algorithms to find failure scenarios. The experiments are performed in a partial multi-agent environment with one AC under test driving alongside non-AC traffic, as non-players. As a limitation, the work does not address the problem of improving the robustness of the same ACs using industrial-grade urban driving simulators. Furthermore, the work could benefit from adding more than one AI-based AC for testing. Another work in [19] proposes a programmatic interface that enables designing parameterized environments and test cases for ACs. These test parameters control the behavior and positioning of various actors alongside AC under test, and support test input generation strategies. While the experiments are performed by training the neural network-based driving models, the work is limited to a single-agent AC environment. Also, the work could benefit from parameterized environments for generating edge cases that can increase the robustness of ACs. Furthermore, authors in [37] propose a whitebox method for testing ACs by triggering as many neurons in the driving model as possible for finding failure scenarios. They pose an optimization problem and apply gradient ascent over the results of test inputs in order to maximize the chance of finding corner cases. As a limitation, the driving scenarios are only tested in a single-agent environment. In contrast, our work is focused on blackbox adversarial testing of ACs in a multi-agent driving environment.

B. Adversarial Testing of ACs

Recent work [23] proposes to use RL-based driving agents to test connected cars by perturbing both the inputs and outputs of a car during training. However, this approach targets mixed-traffic driving with a single AC and multiple human-driven cars, thus it does not consider complex scenarios having more than one non-communicating AC agent. Another work [24] performs adversarial RL for testing a multi-agent driving environment by training more than one adversarial RL agent against one rule-based driving model. While the results look promising, the approach only covers the cases where the trained adversarial cars are exposed to a single non-AI model. As another limitation, the approach has not been evaluated on more complex adversarial driving scenarios, such as T-intersection, which we target in our work.

Another work [22] uses RL to stress-test ACs in a simulated environment. The extension of this work [25] proposes the idea of reward augmentation for increasing the search space and also finding failure cases in driving policies. Compared to our work, they lack multi-agent test cases even on a small scale. Besides, the work is tested neither in a vision-based simulator nor in real-world driving conditions. Furthermore, while the work improves driving conditions for experiments, it uses adversarial perturbations as noise in the simulation model itself. In contrast, our work adds perturbations by the adversarial car's policy, thus adding adversarial actions as example trajectories for improving AC's driving policies.

Authors in [20] proposes a Bayesian optimization-based method for testing ACs. Their method involves creating adversarial scenarios in a Carla-based urban driving simulation [38] to expose the weaknesses of autonomous driving policy. Another work [17] [18] also uses an optimization technique for producing physical attacks on driving lanes, in order to attack vision-based driving models. Compared to our work, these works are lacking multi-agent AC scenarios. Authors in [13] use a GAN model to generate adversarial objects able to attack LiDAR-based driving systems. Another work [39] uses GAN to apply metamorphic testing to CNN-based driving models. Authors in [40] propose a stress testing methodology for LiDAR based perception. Using a real-world driving dataset, they use various weather conditions to test the performance of autonomous driving systems. However, neither of these approaches has been tested in an RL-based multi-agent AC environment.

III. MAD-ARL Formulation

Our work addresses the problem of adversarial testing of autonomous cars in a multi-agent driving environment for the purpose of (**i**) finding failures in AC driving models, and (**ii**) improving the robustness of AC driving models against these failures.

We model our problem as a 2-player Markov game [41], where one type of a player is the autonomous driving agent under test, which we call a ***victim***, and the other player is

the adversarial driving agent, which we call an ***adversary***, and who is trying to exploit the weakness of the victim. We denote our victims and adversary agent as T_1, T_2 and α, respectively (we consider two ACs under test). The Markov game $M = (S, O, (A_{T1}, A_{T2}, A_\alpha), P, (R_{T1}, R_{T2}, R_\alpha))$ in a multi-agent environment consists of O set of state observations and A_T A_α represents action set. P is a joint state transition probability function $P : S \times A_T \times A_\alpha \mapsto \triangle(S)$, where $\triangle(S)$ defines the probability distribution of the next state. Reward function R is based on maximizing the cumulative sum of rewards as $R : S \times A_T \times A_\alpha \mapsto \mathbb{R}$. Each player in the set $\{T_1, T_2, \alpha\}$ depends on the current state observation to perform actions and reach the next state while receiving the desired rewards.

A. Finding Failures in AC Driving Policies

The adversary and victim agents work as ***independent non-communicating competitive*** players. This means that they have no white box access to each other's input state, as well as no shared information to weights parameters. The victim agents are first given the shared environment to train their policies π_{T1} and π_{T2} in the absence of an adversarial player. The adversary, however, is provided access to the action state sampled from π_T. Since the adversary's policy π_α is trained using pre-trained AC policies, we assume that the victim players have fixed weights during adversarial policy training. This represents a scenario where RL-trained policies for ACs are deployed to the real world and their weights are fixed in order to train any adversarial agent for testing. At this point, the Markov game consisting of two players can be treated as one-player MDP problem, since the victim policy π_T is held fixed.

The goal of the adversarial player is to learn a policy π_α maximizing the sum of discounted rewards:

$$\pi_\alpha = \sum_{t=0}^{\infty} \gamma^t R_\alpha(s^{(t)}, a_\alpha^{(t)}, s^{(t+1)})$$

where $a_\alpha \sim \pi_\alpha(.|s^{(t)})$ are actions sampled from the adversary policy and $s^{(t+1)} \sim P_\alpha(s^{(t)}, a_\alpha^{(t)})$ is the next state given the transition probability. Since the current problem is scoped as a model-free approach, the MDP dynamic model P_α is unknown.

When the adversarial policy is trained, we use it to find uncommon behavior patterns for the victim's players by adding natural observations (see Section IV) for the victim DRL policies π_T.

B. Improving the Robustness of AC Policies by Retraining

Once we observe the effectiveness of the adversary in finding failure test cases for the ACs, we retrain the victim models by unfreezing their weight parameters, while keeping the trained adversarial player as part of the training environment. This leads to improved *robustness* which in terms of DRL performance is its resistance towards out-of-distribution inputs and adversarial attacks [42]. Thus, the goal of the autonomous agents $\{\pi_{T1}, \pi_{T2}\}$ is to maximize the sum of the discounted reward independently, that is:

$$\pi_{T1} = \sum_{t=0}^{\infty} \gamma^t R_{T1}(s^{(t)}, a_{T1}^{(t)}, s^{(t+1)})$$

$$\pi_{T2} = \sum_{t=0}^{\infty} \gamma^t R_{T2}(s^{(t)}, a_{T2}^{(t)}, s^{(t+1)})$$

IV. MAD-ARL Framework

In this section, we present the proposed framework and the methodology for improving the robustness of AC driving policies in a multi-agent environment. An overview of the two-step methodology using the MAD-ARL framework is illustrated in Figure 2.

In the MAD-ARL framework, we consider two victim driving agents and one adversary driving agent. An ***agent*** is an entity that is able to observe the environment and perform actions in order to make an intelligent decision in the given environment. The observations of our multi-agent driving environment for the victim agents are manipulated by adding an adversarial agent in the environment. The adversarial agent is trained to take actions such to create observations that appear *natural* to the victim agents while being adversarial in nature. As an example, the adversary is learning to steer offroad most of the time while crossing the intersection. Such unusual behavior will act as an adversarial noise to the visual observations of victim ACs. The same framework is utilized to retrain the weights of the victims against the adversary in order to increase robustness and improve driving performance by lowering the number of collisions and offroad steering failures.

A. Proximal Policy Optimization

Our AC agents use Proximal Policy Optimization (PPO) [43] as a policy gradient method to learn a driving policy by encountering a simulated environment in each training episode. The PPO helps perform on-policy learning within simulation instead of a dataset (replay buffer) type of learning. It also helps focus on policy updates with stability while learning over a change in data distributions, as well as address a large hyperparameter initializing space.

The details of the hyperparameters selected for the training of the victim and adversary driving agents are given in Table I.

TABLE I: Hyperparameters for the PPO DRL model.

Stage	**Hyperparameter**	**Value**
Gathering Experience	Minibatch Range	64
	Epochs per Minibatch	8
	Batch Mode	Complete Episodes
Updating Policy	Discount factor (γ)	0.99
	Clipping (ϵ)	0.3
	KL Target	0.03
	KL Initialization	0.3
Other Hyparameters	Value Loss Coefficient	1.0
	Entropy Regularizer	0.01

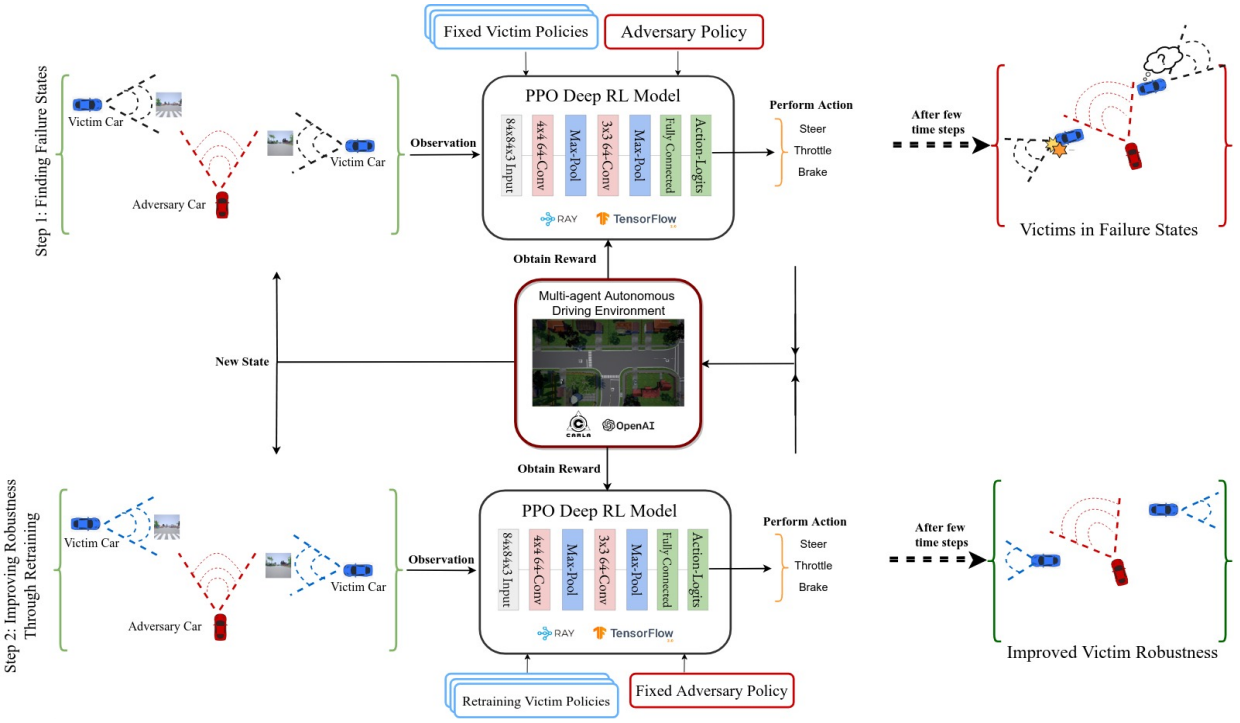


Fig. 2: Illustration of MAD-ARL framework for improving the robustness of AC driving policies in a multi-agent environment. Each agent receives an input image of 84x84x3 which is passed to a PPO-based DRL model. The actions are selected at the output layer of every agent and are performed in the next time step of the simulation in order to obtain reward and a new observation state. Top row of the diagram displays the first step where we find failure scenarios of the victim policies, whereas the bottom diagram shows the second step that involves retraining of victims. Both steps of the framework are performed in an urban driving simulated environment.

B. Deep Neural Network Model

The summary of the DRL architecture, including the input, hidden, and output layer is displayed in Figure 2. The input state $S \subset \mathbb{R}$ of our DRL algorithm receives a partial observation of 84x84x3 dimension images from the camera sensors. Cameras are mounted in front of each driving model which provides feeds as an input state observation to the autonomous and adversary cars model at each step of the simulation. The 3-dimensional input images are passed through convolutions and hidden layers to reach the output layer for control commands.

At the output layer, we have nine discrete values as the action space which are used by each driving agent to make control decisions. All of the discrete actions can be summed into three main actions: Steer, Throttle, and Brake.

C. Reward Functions

Each agent is following MDP described in the MAD-ARL formulation, and therefore, at each time step, the driving models collect trajectories of (S, R, A). R is the reward gained in return for the actions chosen by the driving car's policy function, given the input observations.

1) Victim Reward Function: The victim policies are trained as driving ACs with the goal to safely reach as close to the desired destination as possible. The victim agents reward can be described as:

$$R_{Victim} = \underbrace{(D_{t-1} - D_t)}_{\text{Distance}} + \underbrace{(F_t)}_{\text{Forward Speed}} /10 - 100.0\,\underbrace{(CV_t + CO_t)}_{\text{Collision}} - 0.5\,\underbrace{(IO_t + IOL_t)}_{\text{Offroad steering}} + \beta$$

where D is the distance covered, F is the forward speed of the agent, CV and CO are the boolean values telling whether there is any collision with other vehicles and environment objects, and IO along with IOL refers to driving offroad at the intersections and outside the desired lane represented as boolean. At the end of the equation is a constant β used to encourage driving in a desired ground truth lane. From the above equation, it is clear that the victim driving policies are sensitive toward any offroad steering errors and collisions.

2) Adversary Reward Function: We are defining two different types of reward functions for the adversarial player to see which one performs better in testing and retraining the policies of the victim ACs. The two adversarial reward functions associated with the policies $\pi_{\alpha 1}$ and $\pi_{\alpha 2}$ are named $R_{collision}$ and $R_{offroad}$. The motivation behind two different reward based adversaries is to show that the adversary with no collision and minimal offroad steering reward function

is enough to create effective adversarial actions than the collision-focused adversarial agent.

$R_{collision}$ aims to maximize the rate of collision and offroad steering during the adversarial training. $R_{collision}$ is formulated as:

$$R_{Collision} = (D_{t-1} - D_t) + (F_t)/10 \overbrace{+5.0\,(CV_t + CO_t) + 0.05\,(IO_t + IOL_t)}^{\text{Collision and Offroad steering in } R_{Collision} \text{ Adversary}}$$

$R_{offroad}$ on the other hand aims to maximize the rate of offroad steering, and is thus formulated as:

$$R_{Offroad} = (D_{t-1} - D_t) + (F_t)/10 \overbrace{+0.05\,(IO_t + IOL_t)}^{\text{Offroad steering only in } R_{Offroad} \text{ Adversary.}}$$

D. Hyperparameters

The hyperparameters used in different phases of training of all ACs are shown in Table II. During the testing phase, explained in Section VI, we run 50 total episodes, each having 2000 simulation steps per driving agent.

TABLE II: Hyperparameters for the training of the baseline victim AC models, the adversarial model, and retrained victim AC models

Hyperparameter	Baseline	Adversarial	Retrained Victim
Total Training Steps	300672	57728	133888
Total Training Episodes	610	101	306
Learning Rate	0.0006	0.0006	0.0006
Batch Size	128	128	128
Optimizer	Adam [44]	Adam	Adam

The details of the hyperparameters for all the AC and adversary agents are provided in the GitHub repository [2].

V. Experiments

The experiments aim to demonstrate the effectiveness of the proposed framework for testing and improving driving policies in a multi-agent car environment. To this end, first, we train a single adversarial driving agent against one victim AC agent to test more than one victim AC driving policy. The purpose of the adversary is to expose errors in the driving policies of the AC agents, such as the inability to avoid collisions and offroad steering accidents. Later we retrain the AC agents using the adversarial inputs and evaluate how much their driving policies improved compared to their baseline performance.

Specifically, the research questions aim to evaluate:

RQ1: How effective is the adversarial driving policy in finding failure driving scenarios in victim ACs?

RQ2: Does retraining the victim ACs using the adversarial inputs improve the agent's performance in terms of reduced collisions and offroad steering errors?

[2]https://github.com/T3AS/MAD-ARL

A. Evaluation Metrics

We evaluate the driving performance of victim ACs using the following metrics:

- C_V: rate of collision with another vehicles
- C_R: rate of collision with any other road objects
- O_S: rate of offroad steering from a ground truth driving lane
- $TTFC$: time it takes to have the first collision

To evaluate the effectiveness of the adversarial driving policy in finding failure driving scenarios in victim ACs, we compare the AC's baseline performance (no adversary in the environment) with its performance when driving in the environment with an adversary present. To evaluate the effectiveness of adversarial retraining as a strategy to improve the driving performance of victim ACs, we also compare the performance of victim AC's performance and retrained victim AC's performance when driving in the environment with an adversary present.

B. Experimental Setup

We use *Town 3* scenario provided by the Python Carla API and Macad-gym [45] in our partially-observable urban-based driving environment. This environment has three independent non-communicating agents spawned close to the T-intersection throughout the training and testing steps, where two are the victim ACs T_1 and T_2, and one is the adversarial agent α. The choice of T-intersection as a driving scenario is based on its higher complexity for an AC agent to handle, as the adversarial agent can be easily faced by victim policies during testing episodes.

The goal of T_1 and T_2 is to drive straight across the intersection without errors, while α aims to take a left turn in the same driving situation. The starting and ending state locations of each driving agents are:

- T_1 start: [188, 59, 0.4], end: [167, 75.7, 0.13]
- T_2 start: [147.6, 62.6, 0.4], end: [191.2, 62.7, 0]
- α start: [170.5, 80, 0.4], end : [144, 59, 0]

where victim ACs strictly follow the mentioned coordinates as ground truth to improve their driving policies. On the contrary, the adversary player is less focused on reaching the desired destination and aims to deviate towards collision and offroad steering behavior.

The sequence of the training and testing for victim and adversary agents are as follows.

1) Training Victim AC Agents for Baseline: We train both AC policies π_{T1}, π_{T2} in a multi-agent environment with the absence of any adversarial policy as shown in Figure 3(a). After 610 episodes and 300672 steps mentioned in Section IV-D, we move towards our first testing phase to record the baseline performance of both autonomous cars.

2) Training Adversarial Agent: Next, we introduce the adversarial driving agent. We train its policy π_α by providing the victim AC policy, keeping their weight parameters constant during the adversarial training phase as shown in Figure 3(b). The number of episodes for training the adversary agent is kept

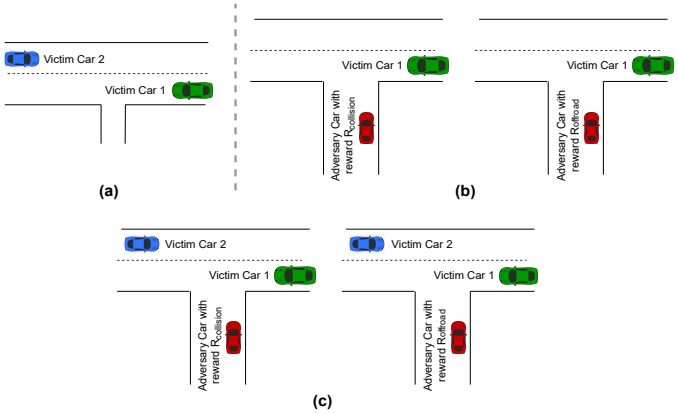


Fig. 3: Illustration of the different phases of experimental setup. (a) shows the training phase of victim policies in the absence of an adversarial agent. (b) shows the training phase of both adversarial policies against one of the victim ACs. While (c) illustrates Step 1 of the experiments where $R_{collision}$ and $R_{offroad}$ based adversaries are used to separately test against victim ACs. The same setup is used to retrain victim ACs for model evaluation in Step 2.

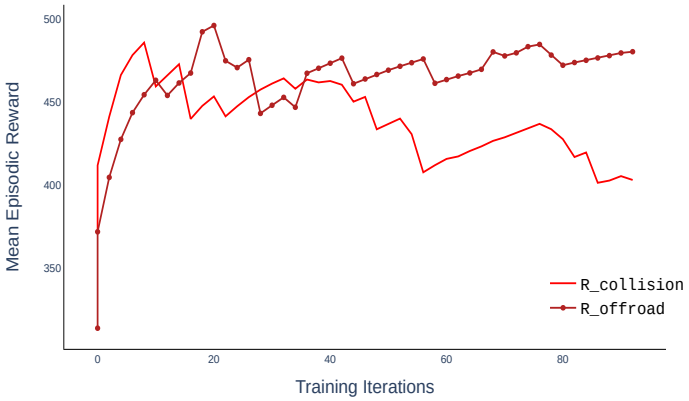


Fig. 4: Scatter plot for showing the model training performance of the adversary agent using two reward functions. Reward values are averaged during each adversary's training episode.

lower than the number of episodes assigned for the victim's baseline model training.

We train the adversarial policy using two different adversarial reward functions, $R_{collision}$ and $R_{offroad}$, separately, as shown in Figure 3(b). This helps evaluate which adversarial agent is more effective in exposing errors in the victim ACs. We use 101 episodes to train the adversarial policy using both reward functions.

The performance of the adversarial policies that are individually trained is depicted in Figure 4. By training the DRL policies for 101 episodes, the adversarial policy $\pi_{\alpha 2}$ trained on $R_{offroad}$ reward function converges faster than $\pi_{\alpha 1}$ trained on $R_{collision}$. The policy $\pi_{\alpha 2}$ gets to a stable mean episodic reward state after crossing half of the training steps, while $\pi_{\alpha 1}$ tends to fluctuate throughout the training phase. Still, both reward-based adversarial policies lead the victim ACs into error states when tested.

3) Two-step Improvement of the Robustness of Victim ACs: **Step 1: Finding Failure States** We test the behavior and control decisions of both victim AC agents when exposed to the trained adversarial driving agent and compare the results with our baseline victim policies. The adversary is able to learn a generalized agent that is used to simultaneously attack both victim policies in a shared driving scenario as displayed in Figure 3(c). Using the evaluation metrics described in Section V-A, we compare the driving behavior of the victim ACs as baselines against adversary agents. The results are described in Section VI.

Step2: Retraining Victim ACs for Improved Robustness Finally, we unfreeze the weights of the victim AC agents to retrain their end-to-end driving policies by keeping the adversarial agent in the same environment. Since there are two different adversarial reward-based policies involved, the retraining of the victim ACs is done twice separately, as depicted in Figure 3(c). After the retraining is done, we test the victim ACs to see how much they improved compared to their baseline performance.

C. Simulation Setup

We use *RLlib* [46] from Ray framework which is an open source project providing a very fine-tuned and scalable RL implementation interface. We also use *Carla* [38] urban driving simulation framework for training, testing, and validating ACs. For integrating Carla and Open AI's Gym toolkit in a multi-agent urban driving environment, we utilize open source platform *Macad-gym* [45]. We are also using *Tensorflow* [47] version 2.1.0 within the RLlib library for creating DRL based model architectures.

VI. Results & Analysis

In this section, we present and discuss experimental results, which are made available in the GitHub repository [3].

A. Effectiveness of Adversarial Driving Policy in Finding Failure Driving Scenarios in Victim ACs

We measure the effectiveness of the adversarial driving policy in exposing failures in victim ACs in terms of four metrics: C_V, C_R, O_S, and $TTFC$. For C_V, C_R, and O_S, we calculate the percentage of error within each episode (as a value between 0 and 1). In each episode, we run 2000 simulation steps and at the end of 50 episodic test runs, we compute the average error rate for each metric across all the episodes. The results are shown in Figure 5 and the average error rate is presented in Table III. For the first three metrics in the table C_V, C_R, O_S victim policies having values closer to 0 are performing error-free driving, whereas values closer to 1 indicate a higher failure rate of the victim policy. As for the fourth metric $TTFC$, the bottom most row in the table

[3]https://github.com/T3AS/MAD-ARL

TABLE III: Comparison of the behavior of victim ACs before and after adding adversarial car in the environment, and after retraining using adversarial policies, in terms of C_V, C_R, O_S, and $TTFC$ metrics, averaged across 50 episodes. Victim ACs have more collisions and offroad steering errors under the presence of an adversarial agent, compared with baseline victim models. Retraining victim ACs with adversarial inputs improves their driving policies.

	Baseline		After Adversarial Training				After AC Retraining			
			$R_{collision}$		$R_{offroad}$		$R_{collision}$		$R_{offroad}$	
	Victim 1	Victim 2	Victim 1	Victim 2	Victim 1	Victim 2	Victim 1	Victim 2	Victim 1	Victim 2
Collision with cars	0.0	0.0	0.19468	0.0956	0.5484	0.4048	0.2563	0.3934	0.0831	0.0698
Collision with other objects	0.0184	0.0398	0.0	0.1533	0.6465	0.278	0.0	0.1912	0.0	0.0566
Offroad steering error	0.0929	0.2828	0.1645	0.3769	0.9069	0.9747	0.0425	0.2112	0.0358	0.1688
Time To First Collision (seconds)	-	-	59.736	27.98	13.8292	14.5	92.1116	113.5248	15.1688	14.23

shows the time in seconds it takes to detect the first collision in a testing episode.

In the baseline scenario, both victims made no collision with each other during test episodes. Victim policies made uncertain decisions by creating collisions with footpaths and performing offroad steering errors. This is because we are testing the victim agents more than they have explored the environment in each episode. Victim 1, which is on the right side of the scenario, has a better baseline driving policy than Victim 2, as it has a lower rate of collision and offroad steering.

After introducing the adversarial policies to the environment ($R_{collision}$ and $R_{offroad}$), we see that the overall decision making process of both victim ACs is disturbed and their driving performance is decreased. Specifically, the rate of collision with other cars increased for both victims, as they ended up colliding with each other, as well as with the adversarial agent. Although both adversarial policies are finding the AC collision failure cases, $R_{offroad}$-based policy works better in this regard, which results in a higher rate of collision and offroad steering for both victim ACs (colored red in the table). With only offroad steering actions as adversarial action, $R_{offroad}$-based adversary forced victims into collision with each other in the T-intersection scenario. Also, due to early stopping after collision during $R_{collision}$-based policy, the rate of collision with other road objects has not been detected much. On the other hand, $R_{offroad}$-based adversarial policy is also able to find trajectories where victims end up hitting road objects after facing adversarial actions. Furthermore, both victims encountered more offroad steering errors. Victim ACs encountering $R_{offroad}$-based policy in a driving scenario ended up taking high percentage of offroad steering mistakes. Rather than attacking victims aggressively as by $R_{collision}$, $R_{offroad}$-based policy did it better by adding adversarial actions as natural adversarial observations produced in a shared driving environment.

In terms of the time to first collision evaluation metric, there were no collisions in the baseline scenario for both victim ACs (denoted with dash in Table III). After introducing the adversarial policies in the environment, collisions occurred after some seconds. Specifically, $R_{offroad}$-based policy drove victim ACs into collisions earlier than $R_{collision}$-based policy.

In summary, the results of the experiments demonstrate that introducing an adversarial policy to the environment is an effective strategy for finding failure driving scenarios in ACs.

B. Improving Victim ACs Performance by Retraining

By retraining the victim ACs using inputs from $R_{collision}$- and $R_{offroad}$-based adversaries, we check whether the ACs' driving performance improved in terms of reduced collisions and offroad steering errors, compared to the stage before retraining. The evaluation results are shown in Table III. Specifically, the rate of collision with cars and other objects decreased for the victims retrained using $R_{offroad}$-based adversary. The reason is that $R_{offroad}$-based adversary provides adversarial examples for victim ACs by maintaining collision-free distance, therefore helping victims to learn how to avoid collisions while crossing an intersection. $R_{collision}$ did not help much during the retraining process of the victim agents. It is mainly due to its collision-focused driving nature and thus the victims were unable to learn to avoid collisions with each other. Victims face collisions that are intentional by the adversary and most of the time they are unable to recover from such collisions in the episodic runs. Victim 2, being a weaker policy among the two ACs, also ended up colliding with road objects. Similarly, the number of offroad steering errors is reduced for the victims retrained using $R_{offroad}$ based adversarial policy since they slowed down after having collisions with other cars while retraining against $R_{collision}$, resulting in less space for improving the offroad steering behavior.

Victims have neither seen such collision-focused drivers during baseline training, nor they are prepared for recovery steps when they face one in retraining. This is usually the case in training ACs in simulated or real-world datasets. The reason we have added $R_{collision}$ in our experiments is to show that it is not practical to add collision-focused driving cars around victims in a multi-agent environment. We need a better and more efficient framework where not only the RL-based AC victims are tested but also their robustness is improved.

Furthermore, the results show that the time to detect the first collision has been increased after retraining with the $R_{collision}$-based adversarial policy. For the $R_{offroad}$-based adversarial policy, even if the time to the first collision increased for only one victim AC, the overall rate of collision has decreased significantly for both ACs. This is because, after the first collision, the ACs were able to recover from failure states and continue with the safe driving behavior.

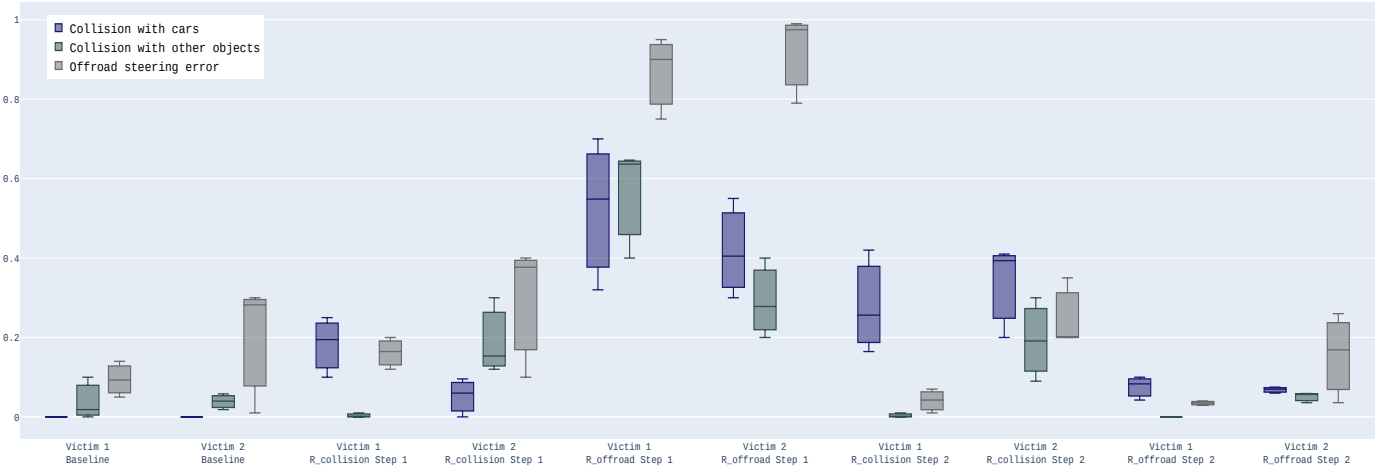
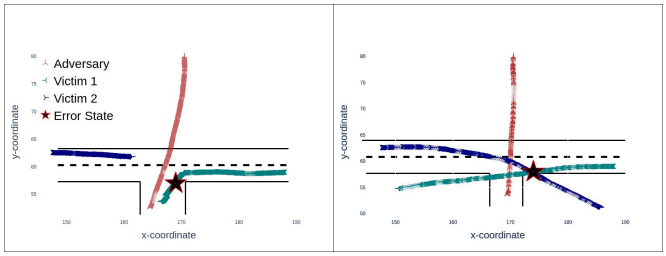


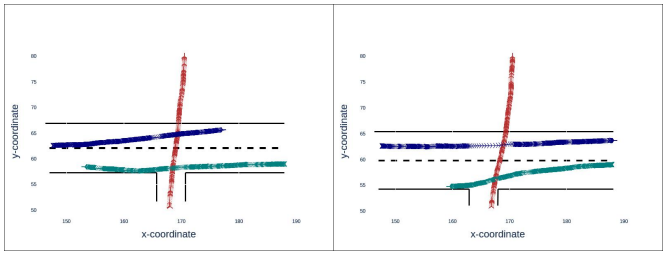
Fig. 5: Overall performance of victim ACs before and after adding adversarial car in the environment, and after retraining using adversarial policies, in terms of C_V, C_R, and O_S metrics.

In summary, these results show that the retraining of victim ACs with adversarial policies helps in increasing the robustness of victims' driving performance. $R_{offroad}$-based adversarial policy once again proves to be more effective in adding better adversarial actions as observations to the driving scenes of the victims. Victims colliding in the testing episodes after retraining were able to recover and drive safe with minimal offroad steering errors. Overall, the results show that the $R_{offroad}$-based adversary is more effective in making ACs more robust.

We visualize the driving performance of the victim AC agents before and after retraining in Figure 6. The figure shows a 2-dimensional aerial view of the victim ACs' driving coordinates. Plots (a) and (b) represent the performance when the victim agents are exposed to the adversary for the first time, while plots (c) and (d) represent the improvement in their driving policies as the result of retraining. The plots do not take a time factor into consideration, which is important to mention since any victim car overlapping with the adversary does not necessarily mean a collision state. Plot (a) depicts Victim 1 driving offroad without crossing the intersection, due to the adversarial agent. Plot (b) depicts a failure scenario where Victim 2 collides with the adversary and drives offroad. Error states in these two plots are marked with red stars. Plot (c) and (d) depict cases of improved (retrained) driving policies of the victim agents, who are now able to avoid collisions with cars and other road objects, as well as to stay in the driving lane while crossing the intersection.



(a) Victim ACs before retraining (b) Victim ACs before retraining



(c) Victim ACs after retraining (d) Victim ACs after retraining

Fig. 6: 2D visualization of the victim and adversary driving coordinates. (a) and (b) display two failure scenarios found while testing victim cars in the presence of an adversarial agent. (c) and (d) illustrate the same victim policies performing better once they are retrained with the adversarial policies.

VII. Conclusion & Future Work

In this work, we propose a framework named MAD-ARL which is a multi-agent driving environment designed for improving the robustness of autonomous cars using adversarial driving models. ARL is trained against a victim player in order to find unwanted driving decisions of autonomous cars that are also trained on a DRL-based policy. By exposing the same adversarial car against the victim agents for retraining, the agents show improvements in their end-to-end decision driving controls, mainly in terms of fewer collisions and offroad steering errors compared to their originally trained (adversary-free) policies.

Future Work: This work can be further extended to ACs operating in mass-traffic scenarios having more cars, pedestrians, and a traffic light network as part of the multi-agent environment. In such complex environments, mixed competitive ACs need to be tested in larger state space for finding edge cases using adversarial agents. Furthermore, we

plan to investigate how retraining the adversarial agent affects the performance of victim autonomous cars. We also plan to explore and compare the robustness of different DRL algorithms used for autonomous driving research, when they are exposed to different types of adversaries. Also, we will extend current driving scenario with different training and testing episodic steps for evaluating the driving performance of RL-based models.

REFERENCES

[1] J. Garcia, Y. Feng, J. Shen, S. Almanee, Y. Xia, Chen, and Q. Alfred, "A comprehensive study of autonomous vehicle bugs," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ser. ICSE '20, 2020.

[2] V. Riccio, G. Jahangirova, and A. e. a. Stocco, "Testing machine learning based systems: a systematic mapping," *Empir Software Eng*, p. 5193, 2020.

[3] D. Marijan and A. Gotlieb, "Software testing for machine learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[4] H. Schäfer, "End-to-end lateral planning," in *comma.ai*, 2021.

[5] P. Koopman and M. Wagner, "Challenges in autonomous vehicle testing and validation," *SAE International Journal of Transportation Safety*, 2016.

[6] P. Koopman and M. Wagner, "Autonomous vehicle safety: An interdisciplinary challenge," 2017.

[7] K. L. Tan, Y. Esfandiari, X. Y. Lee, Aakanksha, and S. Sarkar, "Robustifying reinforcement learning agents via action space adversarial training," in *ACC*, 2020.

[8] X. Y. Lee, Y. Esfandiari, K. L. Tan, and S. Sarkar, "Query-based targeted action-space adversarial policies on deep reinforcement learning agents," in *ACM/IEEE 12th International Conference on Cyber-Physical Systems (ICCPS)*, 2021.

[9] G. Li, Y. Li, S. Jha, T. Tsai, M. Sullivan, S. K. S. Hari, Z. Kalbarczyk, and R. Iyer, "Av-fuzzer: Finding safety violations in autonomous driving systems," in *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, 2020.

[10] B. Gangopadhyay, S. Khastgir, S. Dey, P. Dasgupta, G. Montana, and P. Jennings, "Identification of test cases for automated driving systems using bayesian optimization," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019.

[11] A. Gambi, M. Mueller, and G. Fraser, *Automatically Testing Self-Driving Cars with Search-Based Procedural Content Generation*, 2019.

[12] C. Sitawarin, A. N. Bhagoji, A. Mosenia, M. Chiang, and P. Mittal, "Darts: Deceiving autonomous cars with toxic signs," in *arXiv*, 2018.

[13] Y. Cao, C. Xiao, D. Yang, J. Fang, R. Yang, M. D. Liu, and B. Li, "Adversarial objects against lidar-based autonomous driving systems," in *arXiv*, 2019.

[14] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, *DeepRoad: GAN-Based Metamorphic Testing and Input Validation Framework for Autonomous Driving Systems*, 2018.

[15] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the 40th International Conference on Software Engineering*, 2018.

[16] M. Uřičář, P. Křížek, D. Hurych, I. Sobh, S. Yogamani, and P. Denny, "Yes, we gan: Applying adversarial techniques for autonomous driving," in *Society for Imaging Science and Technology*, 2019.

[17] "Attacking vision-based perception in end-to-end autonomous driving models," in *Journal of Systems Architecture*, 2020.

[18] J. Yang, A. Boloor, A. Chakrabarti, X. Zhang, and Y. Vorobeychik, "Finding physical adversarial examples for autonomous driving with fast and differentiable image compositing," in *arXiv*, 2020.

[19] R. Majumdar, A. Mathur, M. Pirron, L. Stegner, and D. Zufferey, "Paracosm: A test framework for autonomous driving simulations," in *International Conference on Fundamental Approaches to Software Engineering*, 2021.

[20] Y. Abeysirigoonawardena, F. Shkurti, and G. Dudek, "Generating adversarial driving scenarios in high-fidelity simulators," in *ICRA*, 2019.

[21] K. Jang, E. Vinitsky, B. Chalaki, B. Remer, L. Beaver, A. A. Malikopoulos, and A. Bayen, "Simulation to scaled city: Zero-shot policy transfer for traffic control via autonomous vehicles," in *10th ACM/IEEE ICCPS*, 2019.

[22] M. Koren, S. Alsaif, R. Lee, and M. J. Kochenderfer, "Adaptive stress testing for autonomous vehicles," in *IEEE Intelligent Vehicles Symposium (IV)*, 2018.

[23] B. Chalaki, L. E. Beaver, B. Remer, K. Jang, E. Vinitsky, A. M. Bayen, and A. A. Malikopoulos, "Zero-shot autonomous vehicle policy transfer: From simulation to real-world via adversarial learning," in *ICCA*, 2020.

[24] A. Wachi, "Failure-scenario maker for rule-based agent using multi-agent adversarial reinforcement learning and its application to autonomous driving," in *IJCAI*, 2019.

[25] A. Corso, P. Du, K. Driggs-Campbell, and M. J. Kochenderfer, "Adaptive stress testing with reward augmentation for autonomous vehicle validation," in *ITSC*, 2019.

[26] F. Haq, D. Shin, S. Nejati, and L. C. Briand, "Comparing offline and online testing of deep neural networks: An autonomous car case study," in *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, 2020.

[27] G. Papoudakis, F. Christianos, A. Rahman, and S. V. Albrecht, "Dealing with non-stationarity in multi-agent deep reinforcement learning," 2019.

[28] M. Holen, R. Saha, M. Goodwin, C. W. Omlin, and K. E. Sandsmark, "Road detection for reinforcement learning based autonomous car," in *Proceedings of the The 3rd International Conference on Information Science and System (ICISS)*. Association for Computing Machinery, 2020.

[29] B. Tan, N. Xu, and B. Kong, "Autonomous driving in reality with reinforcement learning and image translation," 2018.

[30] P. Almási, R. Moni, and B. Gyires-Tóth, "Robust reinforcement learning-based autonomous driving agent for simulation and real world," 2020.

[31] "Deep reinforcement learning for autonomous driving," 2018.

[32] H. Porav and P. Newman, "Imminent collision mitigation with reinforcement learning and vision," in *21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

[33] A. Gleave, M. Dennis, C. Wild, N. Kant, S. Levine, and S. Russell, "Adversarial policies: Attacking deep reinforcement learning," in *ICLR*, 2020.

[34] "Chauffeur model," 2019. [Online]. Available: https://github.com/udacity/self-driving-car/tree/ master/steering-models/community-models/chauffeur

[35] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proceedings of the IEEE international conference on computer vision*, 2015.

[36] M. Klischat, E. I. Liu, F. Holtke, and M. Althoff, "Scenario factory: Creating safety-critical traffic scenarios for automated vehicles," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020.

[37] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated whitebox testing of deep learning systems," in *SOSP*, 2017.

[38] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *CoRL*, 2017.

[39] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems," in *ASE*, 2018.

[40] H. Delecki, M. Itkina, B. Lange, R. Senanayake, and M. J. Kochenderfer, "How do we fail? stress testing perception in autonomous vehicles," 2022.

[41] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *11th International Conference on International Conference on Machine Learning*, 1994.

[42] N. Drenkow, N. Sani, I. Shpitser, and M. Unberath, "Robustness in deep learning for computer vision: Mind the gap?" *arXiv preprint arXiv:2112.00639*, 2021.

[43] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, 2017.

[44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, Y. Bengio and Y. LeCun, Eds., 2015.

[45] P. Palanisamy, "Multi-agent connected autonomous driving using deep reinforcement learning," in *IJCNN*, 2020.

[46] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, W. Paul, M. I. Jordan, and I. Stoica, "Ray: A distributed framework for emerging AI applications," *CoRR*, 2017.

[47] "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org.