Generating Socially Acceptable Perturbations for Efficient Evaluation of Autonomous Vehicles

Songan Zhang[1], Huei Peng[2], Subramanya Nageshrao[3] and H. Eric Tseng[3]

Abstract— **Deep reinforcement learning methods have been considered and implemented for autonomous vehicle's decision-making in recent years. A key issue is that deep neural networks can be fragile to adversarial attacks through unseen inputs, and thus the reinforcement learning policy, that uses deep neural network would be also fragile to malicious attacks or benign but out of distribution perturbations. In this paper, we address the latter issue: we focus on generating socially acceptable perturbations (SAP), so that the autonomous vehicle (AV agent under evaluation), instead of the challenging vehicle (challenger), is primarily responsible for the crash. In our process, one challenger is added to the environment and trained by deep reinforcement learning to generate the desired perturbation. The reward is designed so that the challenger aims to fail the AV agent in a socially acceptable way. After training the challenger, the AV agent policy is evaluated in both the original naturalistic environment and the environment with one challenger. The results show that the AV agent policy which is safe in the naturalistic environment has many crashes in the perturbed environment.**

I. INTRODUCTION

Over the past two decades, deep learning algorithms (DLA) and deep natural network (DNN) have been widely-used in different fields, from image recognition [1]–[3], speech recognition [4], [5], to natural language processing [6]–[8]. However, recent studies show that DNNs may be vulnerable to adversarial perturbations. In [9], the authors found that adversarial image patches can lead white-box DNNs to erroneous classification results. Papernot et al. in [10] further developed an attack using synthetic data generation, to craft adversarial image examples mis-classified by black-box DNNs. A more comprehensive overview of adversarial attack on DNN can be found in [11].

DNNs have been used in the field of deep reinforcement learning (DRL), where the goal is to train an agent to maximize the expected return. DNN can work as an actor net or a critic net, to provide the optimal policy directly or estimate the expected future return for different actions. Therefore,DRL policies are also vulnerable to adversarial perturbations. In [12], the authors characterize different types of adversarial attack on DRL, which can be attacked by adding perturbation to observations or environment transition probabilities. To perturb observations, researchers first follows the same ideas as attacking DNN, which leads the DRL policy to a different action other than the original optimal one [13], [14]. Furthermore, in [12], the environment transition model is perturbed. However, as pointed out by the authors, these attacks are useful only under very specific conditions or requirements.

In the field of autonomous vehicles (AV), researchers also tried to attack existing AV system. In a recent report [15], Tencent's Keen Security Lab showed how they were able to bamboozle a Tesla Model S into switching lanes to drive directly into the oncoming traffic by manipulating the input video. In a recent news [16], Tesla's first-generation Autopilot is reported to be tricked into accelerating from 35 to 85 mph with modified speed limit sign. These attacks are in the category of observation attack. To our best knowledge, there has been little study on attacking the environment transition model, i.e., perturbing the behavior of environment (or surrounding interactive agents), which is the main contribution of this paper.

From the perspective of AV testing, it is difficult to evaluate an AV in a high automation level (i.e. level $\geq$ 3). Zhao et al. have evaluated the Autonomous Emergency Braking (AEB) system in the lane change scenario [17] and the car-following scenario [18] with importance sampling method to accelerate the testing progress. However, the AEB system is still a level 2 feature. O'Kelly et al. [19] manage to train a DNN-based environment model using Generative Adversarial Imitation Learning (GAIL) method with highway driving data on I-80, California. And then the AV features (including level 3 features) are tested as an integral system in the environment and its crash rate is estimated. However, if the statistics of the environment changed, evaluation result is a mis-estimation of the system. In this paper, rather than estimating crash rate based on stationary environment model, we perturb the environment and find the failed cases directly.

This paper will focus on finding an adversarial policy (for the challenger) to attack a given victim AV agent, with manners that are socially acceptable. In other words, we focus on manipulating the environment transitions characteristics by adding a challenger whose objective is to induce a collision without the challenger being at fault. The victim AV agent is evaluated as an integral system with all related features (including the lane change decision making policy) in the environment containing a challenger. The paper is organized as follows. In Section II, related works are introduced, followed by Section III, where we give a brief review on reinforcement learning. Section IV describes the details of the victim policy. In Section V, we introduce the training

*This work was supported by Ford Motor Company

[1]Songan Zhang is a Ph.D. Student in Mechanical Engineering, University of Michigan, email: songanz@umich.edu

[2]Huei Peng is a professor of Mechanical Engineering, University of Michigan, email: hpeng@umich.edu

[3] Subramanya Nageshrao and H. Eric Tseng are research scientists in the Ford Motor Company.

environment and develop socially acceptable attack design. The simulation setup and results are shown in Section VI and VII, respectively. Our paper is concluded by Section VIII.

II. RELATED WORKS

One of the most widely used adversarial attack techniques is the fast gradient sign method (FGSM). It is originally used in image recognition attack. With small modification, policies obtained through reinforcement learning can also be attacked by this method. With the assumption of white-box attacks, in [13], the authors use FGSM to generate adversarial observations, leading to a pre-trained policy to lose the Pong game. Lin et al. in [14] further developed an enchanting attack aimed at maliciously luring an agent to a certain state. Xiao et al. in [12] extended FGSM to black-box attacks via imitation learning and other methods.

In general, an attacker does not have direct access to the victim's observations. Under this assumption, Gleave et al. [20] demonstrate the existence of adversarial policies in zero-sum games (created by Bansal et al. in [21]) between humanoid robots against black-box victims trained via state-of-the-art reinforcement learning. The adversarial policy reliably win against the victim by generating seemingly random and uncoordinated behavior without even touching the victims which is definitely not a direct attack behavior in a regular sense. It shows that the RL policy can be confused when its opponent's behavior is out of distribution than the one the policy is trained in. Using the multi-agent reinforcement learning to solve a zero-sum Markov game has a long history (from 1994 [22]), and [20] inspired us to pose the question: Can one challenger car which "attacks" the victim car by inducing a collision involving the victim without the challenger itself being at fault?

To be specific about which car is at fault, we utilize the Responsibility-Sensitive Safety (RSS) interpretation introduced in [23]. The RSS model formalizes an interpretation of "Duty of Care" from tort law. The Duty of Care states that an individual should exercise "reasonable care" while performing acts that have the potential harm others. The RSS is a mathematical model formalizing an interpretation of the law which is applicable to AVs, and has been implemented in simulation with NHTSA pre-crash scenarios in [24].

Analogous to the approach described in [12], we will test an AV (the victim in the context of [12]) in a wide variety of traffic situations by adding a challenger vehicle (the attacker in the context of [12]) to induce socially acceptable perturbation. The challenger/attacker has to exercise duty of care by obeying RSS when exploring possible weakness of the AV policy. In the rest of this paper, we will use the term "victim" referring to the AV that is under evaluation and the term "attacker" referring to the challenger to be consistent with the Markov game framework in [12]. Finding an adversarial policy is a single-agent reinforcement learning problem since the victim's policy is fixed in our setup.

III. REINFORCEMENT LEARNING BASICS

In reinforcement learning, an agent is trying to learn an optimal policy to maximize the cumulative reward interacting with the environment. Usually the problem is modeled as a Markov decision process (MDP), defined as: $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma)$, where $\mathcal{S} \subseteq \mathbb{R}^n$ is the state, $\mathcal{A} \subseteq \mathbb{R}^m$ is the action, and $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is the transition dynamics which is usually stochastic. And $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function while $\gamma \in [0, 1)$ is the discount factor.

For each time step t, the agent tries to learn a policy $\pi_\alpha(s_t) = a_t$ with parameters α, where $s_t \in \mathcal{S}$ is the state at time t and $a_t \in \mathcal{A}$ is the action at time t. The expectation of future cumulative reward starting from state s following policy π_α can be described as:

$$V(s|\pi_\alpha) = E_{\pi_\alpha,\mathcal{M}}\left[\sum_{t=0}^{\infty} \gamma^t r_t \Big| s_0 = s\right], \tag{1}$$

where r_t is the reward at time t and V is the value function. And Q function (action value function) is defined as:

$$Q(s, a|\pi_\alpha) = E_{\pi_\alpha,\mathcal{M}}\left[\sum_{t=0}^{\infty} \gamma^t r_t \Big| s_0 = s, a_0 = a\right]. \tag{2}$$

The goal of reinforcement learning is to maximize the expected accumulative reward, i.e. $E_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$. For a discrete action space, the optimal policy can be obtained by learning the accurate Q function Q^* following the optimal policy π_{opt} and thus we have $\pi_{opt}(s) = \arg\max_a(Q^*(s, a|\pi_{opt}))$ which is the greedy policy. Accurate Q function with the optimal policy must satisfy the Bellman equation:

$$Q^*(s, a|\pi_{opt}) = r + \gamma E_{s'}\left[\max_{a'} Q^*(s', a')\right], \tag{3}$$

where s' is the next state.

For high dimensional state space or continuous state space, functional approximation of the Q value is necessary to ensure tractability of the solution. While Q-learning for continuous state space with approximation could cause the Q-network to diverge [25], Deep Q-Network (DQN) [26] has successfully demonstrated value function convergence with empirical results using experience replay buffer and target network.

In this work, we use Double Deep Q-Network (DDQN) as the backbone reinforcement learning algorithm. DDQN is a variation of DQN which uses a target Q-Network to select actions for the evaluation of the next Q value. The technique addresses the problem of overestimating future return (i.e. overoptimism). For details, please refer to the original paper [27].

IV. THE VICTIM AV POLICY

In this section, the environment for training the victim AV agent is described. We study a discretionary lane change decision making problem in this paper. The state space, action space, the victim training reward, and the simulation environment are introduced. For benchmark purposes, the problem definition and the simulation environment are the same as the one used in [28]. This victim AV agent will be evaluated by the SAP later in this paper.

A. Training Environment

The victim environment used in this work is a three lane highway simulator based on [28]. The AV is driving with up to six surrounding vehicles (three vehicles in front, three vehicles behind) as shown in Fig. 1. The blue box is the AV and the 6 red boxes are the six surrounding vehicles whose states are observed. The remaining boxes are environment vehicles whose states are not observed. The surrounding vehicles driving strategy is also descried in [28].

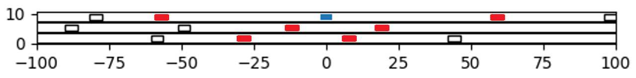

Fig. 1: Three lane highway simulator. The blue box: the AV; red boxes: 6 nearest surrounding vehicles; empty boxes: unobserved surrounding vehicles

The state space $\mathcal{S} \subseteq \mathbb{R}^n$ of the learning agent (AV) includes AV's lateral position y, longitudinal velocity v_x and the relative longitudinal position of the i^{th} surrounding vehicle Δx^i, and the relative lateral position of the i^{th} surrounding vehicle Δy^i and the relative longitudinal velocity of the i^{th} surrounding vehicle Δv_x^i. In total, we have a state space of $2 + 3 \times 6(\text{cars}) = 20$ dimensions, i.e. $\mathcal{S} \subseteq \mathbb{R}^{20}$.

The actions of both the AV and the surrounding vehicles are discrete. As defined in [28], we consider four action choices along the longitudinal direction a_x, namely, *maintain speed*, *accelerate*, *brake*, and *hard brake*. Whereas for lateral actions a_y, we assume three choices, *lane keep*, *change lane to right*, and *change lane to left*. In total, we define 12 different discrete actions $a = [a_x, a_y]$. For detailed action parameters, please refer to [28].

B. Reward Function

The reward r^v is as defined in [28]. It is formulated as a function of (dx, y, v_x), where dx is the distance between the AV and its lead vehicle, y is the lateral position of the AV and v_x is its longitudinal velocity. The reward is defined as:

$$r_x = \begin{cases} \exp\left(-\frac{(dx - dx_{safe})^2}{10dx_{safe}}\right) - 1, & \text{if } dx < dx_{safe} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

$$r_y = \exp\left(-\frac{(y - y_{des})^2}{y_{norm}}\right) - 1, \tag{5}$$

$$r_v = \exp\left(-\frac{(v_x - v_{des})^2}{v_{norm}}\right) - 1, \tag{6}$$

where dx_{safe}, y_{des} and v_{des} are the safe longitudinal distance to the lead vehicle, the target lane position, and desired speed, respectively. These three rewards are normalized by $10dx_{safe}$, y_{norm} and v_{norm}, respectively, so that no single reward dominates the total reward. Then we have $r^v = \frac{1}{3}(r_x + r_y + r_v)$ if no collision and $r^e = -2$ if collision occurs.

After learning in an environment without attacker, the AV learned to keep a safe distance from the front vehicle, drive in the center of a lane and drive as fast as possible without violating the speed limit. On top of that, we implemented a "short-horizon safety check" that evaluates the action chosen by the agent based on user-defined safety rules and provides an alternative default safe action when needed [28].

V. SOCIALLY ACCEPTABLE PERTURBATIONS

In this Section, the process for obtaining socially acceptable perturbations will be introduced. As discussed in Section I and II, we want to attack the victim AV by perturbing the environment transition probability with the requirement that the victim AV takes the responsibility of the crash. To do that, we add an attacker near the victim AV and train it with reward considering socially acceptable attacks.

A. Attacker Training Environment

The state space of the input of the attacker's policy includes information from 6 surrounding vehicles and the victim AV. Same as defined in IV-A, the information includes each vehicle's relative longitudinal position, relative lateral position, and relative speed. In addition, we also include the victim AV's action in the attacker's state space. Therefore, in total we have a 24-dimension state space, i.e. $\mathcal{S} \subseteq \mathbb{R}^{24}$. All the states are scaled for neural network training.

At initialization, the attacker is located near the AV. As shown in Fig. 2, the attacker (the red box) can be observed by the victim AV (the blue box). Although the attacker can observe AV's relative position, velocity and its action, the attacker do not have direct access to the AV's policy. Therefore, we are preforming a black-box attack.

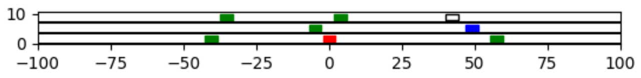

Fig. 2: Attacker's training environment. The blue box: the AV; the red box: the attacker; green boxes: other surrounding vehicles in attacker's state space; empty boxes: unobserved surrounding vehicles by the attacker

B. Reward Considering Socially Acceptable Attack

The key component for training an attacker for generating SAPs is the reward. The attacker is rewarded if it causes a collision between the AV and one of the environment cars (not necessarily the attacker), in which the AV is at fault per crash. In this work, we encode the RSS [23] model and traffic rules through associated reward to train the attacker. Here we recall the 5 "common sense" rules followed by RSS [23]:

1) Do not hit someone from behind.
2) Do not cut-in recklessly.
3) Right-of-way is given, not taken.
4) Be careful of areas with limited visibility.

5) If you can avoid an accident without causing another one, you must do it.

The first three "common sense" principles above are related to traffic rules and can be implemented through associating rewards with pre-crash scenarios. The forth is not applicable in our highway driving scenario. To implement the fifth, we refer to another related paper [24]. Shashua et al. have implemented the RSS model on NHTSA pre-crash scenarios in [24], where they define "proper response" to dangerous situations (related to the fifth rule) as using Minimal Evasive Effort (MEE). MEE deals with cases in which extra caution is applied to prevent potential situations in which responsibility might be shared. Therefore, to design the reward for training the attacker that provides SAPs, we also need to define the pre-crash scenarios and the corresponding MEEs in our application.

In reinforcement learning, the reward function is $r(s, a)$, where the s is the state and a is the attacker's action at state s. To separate the responsibility, we predict the next time-step situation with the victim's action (observed in s) and the attacker's action at current time-step, getting the next state s' and the attacker's action a'. Since the simulator is deterministic, the reward function can be further extended to $r(s, a, s', a')$. If at the next state s' the victim crash, the s is the pre-crash state. In this work, instead of implementing the whole RSS model which considers an entire pre-crash scenario, we only consider one pre-crash state. Therefore, no blame time concept as in [24] is implemented in this application. In the future, if one wants to implement the RSS model considering the entire scenario, we recommend modeling the attacker's policy with recurrent neural network and design the reward function accordingly.

The MEE is implemented in the reward corresponding to each pre-crash situation. As only one time step is being considered and the action space is discrete, we define the right choice of action as MEE for each pre-crash state. As mentioned above, the MEE deals with cases in which responsibility might be shared. Here we extend the MEE concept to situations in which only one car is responsible for the crash. For instance, in the rear-end collision (which is the rear car's fault), we expect the rear car to brake before crash. In this work, we only define the MEE for the responsible car for simplicity, but the MEE for the irresponsible car can be defined similarly. In the future, if one wants to implement the MEE concept in the environment with continuous action space, the MEE should be calculated according to the RSS model, as conducted in [24].

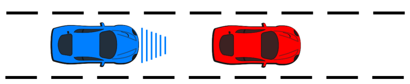
Fig. 3: Pre-crash instance with no car on the lane marker

The pre-crash state can be categorized as no car is on the lane marker, one car is on the lane marker and both cars are on the lane markers. As shown in Fig. 3, when no car is on the lane marker, the rear car (blue one) is the responsible car for the crash. Therefore, the rear car is expected to conduct a hard brake to avoid the crash, i.e., the MEE for the responsible car in this pre-crash state is hard brake.

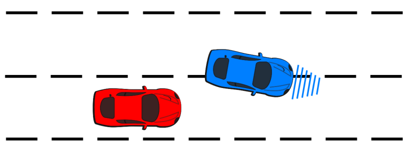
Fig. 4: Pre-crash instance with only one car on the lane marker and crash with the car in the target lane

Fig. 4 shows the pre-crash state in which only one car is on the lane marker and it crash with the car in the target lane. In this case, the lane change car (blue one) is the responsible car and is expected to abort the lane change to avoid the crash. When the car on the lane marker crash with the car in the original lane, the reward is designed as analogous to the situation when no car is on the lane marker.

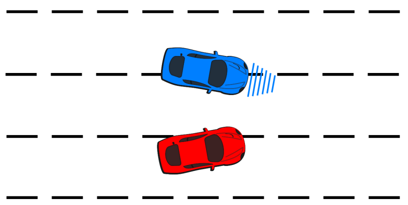
Fig. 5: Pre-crash instance with both cars are on the different lane markers

As shown in Fig 5, we illustrate the pre-crash situation in which both cars are on different lane markers. We assume that vehicles travel on the right, then in this case, both cars are at fault according to multiple traffic laws [29]–[31]. According to the Chinese traffic law [29], the vehicle from the left lane should yield to the vehicle from the right lane. While in the Texas traffic law [30], the vehicle from the right lane should yield. And the right-of-way is not clarified in the New York traffic law [31]. Here we take the Chinese traffic law as an example, i.e., the left car (blue car) is mainly responsible for the collision and is expected to abandon the lane change to avoid the crash. When both cars are on the same lane marker, the reward is designed as analogous to the situation when no car is on the lane marker.

According to the responsibility and the defined MEE for each pre-crash situation, we further assigned the reward. Positive reward is given to the attacker if it lures the victim to end up with a crash which is the victim's responsibility. If the attacker can find a way to further lure the victim to directly crash into another car without MEE, the attacker will be rewarded even more. On the contrary, if the victim is not responsible for the collision, negative reward is given to the attacker. The reward is summarised in Table I

The attacker also has a time cost of -0.05, which encourages the attacker to cause a collision as soon as possible. An episode is terminated if either of the following happens:

- The AV crashes with another car;

TABLE I: Reward design for socially acceptable attack

Pre-crash state	The Responsible Car	Minimal Evasive Effort (MEE)		Attacker's reward			FC
		Responsible car	Irresponsible car	Fault	MEE	Reward	
No car is on the lane marker	The rear car	Hard brake	-	Env. car	No	-1	0
					Yes	-0.5	1
				Victim	No	1	2
					Yes	0.5	3
Only one car is on the lane marker and crash with the car in the original lane: Same as no car is on the lane marker							
Only one car is on the lane marker: crash with the car in the target lane	The lane-change car	Abandon the lane change	-	Env. car	No	-1	0
					Yes	-0.5	1
				Victim	No	1	4
					Yes	0.5	5
Both cars are on the same lane marker: Same as no car is on the lane marker							
Both cars are on the lane marker: different lane markers	Share responsibility, but left car is the principal responsible car	Abandon the lane change	-	Env. car	No	-0.8	0
					Yes	-0.3	1
				Victim	No	0.8	6
					Yes	0.3	7

- The attacker crashes with one car and the reward for the attacker is -1;
- The AV leaves the neighborhood of the attacker, i.e. not being one of the 6 surrounding cars of the attacker and the reward is -1.

We also record the Failure Code (FC) for each episode, as shown in Table I. The definition of each code is described as follows:

- 0: The other car is responsible for the crash (w/o. MEE);
- 1: The other car is responsible for the crash (w. MEE);
- 2: The AV crashes into the front vehicle w/o. hard brake;
- 3: The AV crashes into the front vehicle w. hard brake;
- 4: The AV changes lane and causes collision w/o. trying to abandon the lane change;
- 5: The AV changes lane and causes collision while trying to abandon the lane change;
- 6: The AV changes lane from the left to the middle lane and crashes with the car changing lane from the right to the middle lane, w/o. trying to abandon the lane change;
- 7: The victim changes lane from the left to the middle lane and crashes with the car changing lane from the right to the middle lane, while trying to abandon the lane change.

As shown in Table I, the AV-responsible crashes, in which the victim AV did not use MEE, are valued most by the attacker (i.e. FC: 2, 4 and 6). This kind of crash is the most harmful crash. Moreover, the action chosen by the victim AV's policy before these crashes definitely deserve a closer look and possible revision. In summary, instead of finding a crazy attacker, we trained a "socially acceptable" attacker which explores the weakness of the AV and helps to improve its policy by generating SAPs.

VI. SIMULATION SETUP

In this Section, the simulation setup for training the attacker is described. The reinforcement learning algorithm we use is DDQN. The hyper-parameters used during training is shown in Table II. To accelerate the training and achieve better exploration, we use two replay buffers, one for storing normal cases and the other for storing AV-responsible crash cases, and implementing the model-based exploration method developed previously in [32]. Therefore, the reward for socially acceptable attack and the failure code can be seen as a way to prioritise all the collected samples. Use of two replay buffers is a simpler version of prioritized experience replay as discussed in [33], which has been implemented similarly in [28]. And the model-based exploration method in [32] can help the attacker explore the weakness of the victim based on its understanding of the victim.

TABLE II: Hyper-parameters for the attacker training

	Description	Value
γ	Discount factor	0.9
Δt	Sampling time	1 sec
ρ	Learning rate	$1e{-}6$
ϵ_0	Starting value for ϵ-greedy exploration	0.2
C	Annealing factor for ϵ-greedy exploration	$2e{-}6$
T	Steps for each episode	200
E	Total training episode	$1e5$

During the evaluation phase, the AV is evaluated in the original environment (without the attacker) and then the same environment with one attacker. The AV is evaluated in both environments, with the total number of cars being 10, 15 and 20. The AV is evaluated for $1e6$ episodes and each episode lasts for 200 steps unless terminated early due to a crash.

VII. RESULTS

In this Section, both the training curve of the attacker and the evaluation results of the AV in different environments are reported. As described in [28], after training, the AV's policy become safe and stable in the original environment.

As shown in Fig 6, the attacker is trained for $1e5$ episode and evaluated by 10 roll-outs every 100 episodes. We train the attacker 10 times and average the evaluation roll-outs rewards. As can be seen from the Fig. 6, the attacker's policy is stable after $1e5$ training episodes.

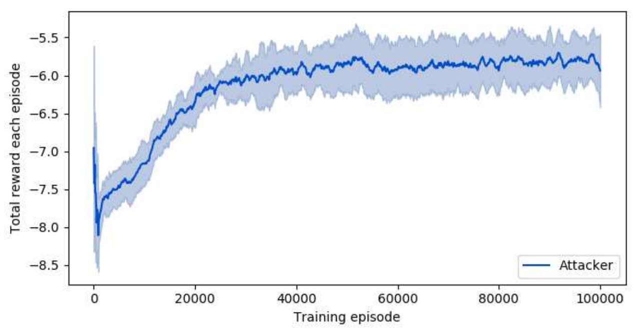

Fig. 6: Average reward from 10 evaluation roll-outs with confidence bound

As described in Section VI, the victim agent is then evaluated in two environments: the original environment and the environment with one trained attacker. In both environment, the victim is evaluated for $1e6$ episodes. The results are shown in Table III. The definition of failure code is described in Section V-B and here we only focus on the AV's responsible crashes (failure code 1 to 7). In the bottom of Table III, we report both the number of crashes between the attacker and the victim, and the total number of crashes.

TABLE III: Number of crashes during evaluation; The victim AV is tested in the env. w/o. attacker and the env. w. one attacker. In the env. w. one attacker: first number is the # of crashes happen between the victim and the attacker; the second number is the total # of crashes.

# of env. cars		Failure Code						
		1	2	3	4	5	6	7
w/o. attacker	10	0	0	5	0	46	0	0
	15	0	0	14	0	55	0	0
	20	0	0	12	0	33	0	0
w. one attacker	10	656/657	239/239	180/186	43/78	101/156	504/504	9/9
	15	447/448	172/172	163/164	26/45	63/88	283/283	3/3
	20	419/598	168/168	137/139	20/32	59/64	237/273	1/1

As can be seen from the table, there are many more crashes when one attacker is introduced. For crashes with failure code 2, 4 and 6, where the victim is mainly responsible, the number of crashes jump from 0 to hundreds. This is a proof that our proposed method can modify the environment transition model to increase risky but socially acceptable behaviors from other vehicles, which can provide hints on how the AV policy can be further improved.

One reason why our method works, is that the original reward function for training the AV does not consider the crash responsibility. As can be seen from Fig 7, the victim (blue car) is facing a situation where it will either crash into the front vehicle or crashed by the rear vehicle. Since the two situations are valued the same by the victim, in this episode, the victim actually crashes into the front vehicle.

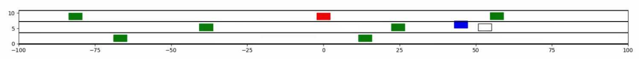

Fig. 7: The victim (blue car) instead of braking and crashed by the following environment car or changing lane, it actually abandons the lane change and crashes into the front car

VIII. CONCLUSIONS

This paper shows that the AV policy learned by deep reinforcement learning can be fragile, i.e., can still result in accidents and collisions even when the vehicles around it behave in a socially acceptable fashion. We design an attack agent by perturbing the environment transition model to induce collisions that AV would be at fault. In this work, one attacker agent is added to the environment, and the safety of the AV drops significantly. The identified collision cases can also be used to train a new AV policy.

There is a lot of potential to extend the work presented in this paper. In this paper, we apply this socially acceptable attack in an environment with discrete action space. However, it can be extended to environment with continuous action space by modeling the MEE with continuous action. Although we add only one attacker to the environment, it can be extended to multi-attacker scenarios. Finally, one can also extend our approach by designing a Markov Game and train both the victim and the attacker together.

ACKNOWLEDGMENT

We would like to thank the authors in paper [28] for making the our AV policy implementation possible.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[2] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[4] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6645–6649, IEEE, 2013.

[5] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, *et al.*, "Deep speech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv:1412.5567*, 2014.

[6] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of machine learning research*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[7] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Advances in neural information processing systems*, pp. 3104–3112, 2014.

[8] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.

[9] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, “Adversarial patch,” *arXiv preprint arXiv:1712.09665*, 2017.

[10] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, “Practical black-box attacks against machine learning,” in *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pp. 506–519, ACM, 2017.

[11] E. R. Balda, A. Behboodi, and R. Mathar, “Adversarial examples in deep neural networks: An overview,” in *Deep Learning: Algorithms and Applications*, pp. 31–65, Springer, 2020.

[12] C. Xiao, X. Pan, W. He, J. Peng, M. Sun, J. Yi, B. Li, and D. Song, “Characterizing attacks on deep reinforcement learning,” *arXiv preprint arXiv:1907.09470*, 2019.

[13] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, “Adversarial attacks on neural network policies,” *arXiv preprint arXiv:1702.02284*, 2017.

[14] Y.-C. Lin, Z.-W. Hong, Y.-H. Liao, M.-L. Shih, M.-Y. Liu, and M. Sun, “Tactics of adversarial attack on deep reinforcement learning agents,” *arXiv preprint arXiv:1703.06748*, 2017.

[15] Tencent Keen Security Lab, “Experimental security research of tesla autopilot,” 2019.

[16] F. Lambert, “Tesla autopilot gets tricked into accelerating from 35 to 85 mph with modified speed limit sign.” `https://electrek.co/2020/02/19/tesla-autopilot-tricked-accelerate-speed-limit-sign/`, 2020. [Online; accessed Mar-2020].

[17] 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.

[18] D. Zhao, X. Huang, H. Peng, H. Lam, and D. J. LeBlanc, “Accelerated evaluation of automated vehicles in car-following maneuvers,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 733–744, 2017.

[19] M. O’Kelly, A. Sinha, H. Namkoong, R. Tedrake, and J. C. Duchi, “Scalable end-to-end autonomous vehicle testing via rare-event simulation,” in *Advances in Neural Information Processing Systems*, pp. 9827–9838, 2018.

[20] A. Gleave, M. Dennis, N. Kant, C. Wild, S. Levine, and S. Russell, “Adversarial policies: Attacking deep reinforcement learning,” *arXiv preprint arXiv:1905.10615*, 2019.

[21] T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, and I. Mordatch, “Emergent complexity via multi-agent competition,” *arXiv preprint arXiv:1710.03748*, 2017.

[22] M. L. Littman, “Markov games as a framework for multi-agent reinforcement learning,” in *Machine learning proceedings 1994*, pp. 157–163, Elsevier, 1994.

[23] S. Shalev-Shwartz, S. Shammah, and A. Shashua, “On a formal model of safe and scalable self-driving cars,” *arXiv preprint arXiv:1708.06374*, 2017.

[24] A. Shashua, S. Shalev-Shwartz, and S. Shammah, “Implementing the rss model on nhtsa pre-crash scenarios,” tech. rep., 2018.

[25] J. N. Tsitsiklis and B. Van Roy, “Analysis of temporal-diffference learning with function approximation,” in *Advances in neural information processing systems*, pp. 1075–1081, 1997.

[26] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.

[27] H. Van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double q-learning,” in *Thirtieth AAAI conference on artificial intelligence*, 2016.

[28] S. Nageshrao, E. Tseng, and D. Filev, “Autonomous highway driving using deep reinforcement learning,” *arXiv preprint arXiv:1904.00035*, 2019.

[29] Government of China, “中华人民共和国道路交通安全法，第三十五条[Law of the People’s Republic of China on Road Traffic Safety, No. 35].” `http://www.gov.cn/banshi/2005-08/23/content_25575.htm/`, 2004. [Online; accessed Mar-2020].

[30] “Texas traffic code law, chapter 545.” `https://statutes.capitol.texas.gov/docs/TN/htm/TN.545.htm`, 2017. [Online; accessed Mar-2020].

[31] “New york state law, vehicle and traffic law, article 25, section 1128.” `http://ypdcrime.com/vt/article25.htm#t1128`. [Online; accessed Mar-2020].

[32] S. Zhang, H. Peng, S. Nageshrao, and E. Tseng, “Discretionary lane change decision making using reinforcement learning with model-based exploration,” in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pp. 844–850, IEEE, 2019.

[33] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, “Prioritized experience replay,” *arXiv preprint arXiv:1511.05952*, 2015.