

Physically Realizable Targeted Adversarial Attacks on Autonomous Driving

by

Prasanth Buddareddygar

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2021 by the
Graduate Supervisory Committee:

Yezhou Yang, Chair
Yi Ren
Georgios Fainekos

ARIZONA STATE UNIVERSITY

May 2021

ABSTRACT

Autonomous Driving (AD) systems are being researched and developed actively in recent days to solve the task of controlling the vehicles safely without human intervention. One method to solve such task is through deep Reinforcement Learning (RL) approach. In deep RL, the main objective is to find an optimal control behavior, often called policy performed by an agent, which is AD system in this case. This policy is usually learned through Deep Neural Networks (DNNs) based on the observations that the agent perceives along with rewards feedback received from environment.

However, recent studies demonstrated the vulnerability of such control policies learned through deep RL against adversarial attacks. This raises concerns about the application of such policies to risk-sensitive tasks like AD. Previous adversarial attacks assume that the threats can be broadly realized in two ways: First one is targeted attacks through manipulation of the agent's complete observation in real time and the other is untargeted attacks through manipulation of objects in environment. The former assumes full access to the agent's observations at almost all time, while the latter has no control over outcomes of attack. This research investigates the feasibility of targeted attacks through physical adversarial objects in the environment, a threat that combines the effectiveness and practicality.

Through simulations on one of the popular AD systems, it is demonstrated that a fixed optimal policy can be malfunctioned over time by an attacker e.g., performing an unintended self-parking, when an adversarial object is present. The proposed approach is formulated in such a way that the attacker can learn a dynamics of the environment and also utilizes common knowledge of agent's dynamics to realize the attack. Further, several experiments are conducted to show the effectiveness of the proposed attack on different driving scenarios empirically. Lastly, this work also studies robustness of object location, and trade-off between the attack strength and attack length based on proposed evaluation metrics.

DEDICATION

This thesis is dedicated to the memory of my late parents, B Raja Reddy, and B Hemalatha.

And to my elder sisters, B Yamuna and B Pavani who supported me throughout my life.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Prof. Yezhou Yang for welcoming into Active Perception Group (APG) and helping me find my own research path. His invaluable guidance and motivation has made a huge impact on my research journey and encouraged me at almost each and every step of my Masters degree milestone. His kind advice has made me realize my priorities and actively engage in the research culture along with other members. Next, I would like to thank Prof. Yi (Max) Ren for providing me the opportunity to engage in such an interesting topic which eventually led to this thesis. I am privileged to have him around every time whenever I got stuck with my research and helped me understand the bits and pieces of a good research. A special thanks to Travis Zhang for being my continuous partner in my MS journey, keeping my motivation up and showing great enthusiasm.

I am grateful to be a part of adversarial group and its members for useful discussions and suggestions. A big thanks to Changhoon Kim for engaging with me when I was struggling in the initial days of my research. Also, thanks to Benjamin Danek for helping me to get onboard on to this project. I thank each and every person of the group, Sheng Cheng, Joushua Feinglass, Xin Ye, Zhe Wang, Yongbaek Cho and others.

I am privileged to be a part of amazing and bright members of APG. I extend my thanks to Tejas Gokhale, Zhiyuan Fang, and Varun Jammula for some of the useful discussions. I would like to thank Mr. Hong Yuan from GoDaddy for being my manager at internship and always allowing me to focus on my college and academics first. Finally, I thank my roommate Vishnu Teja Yalakuntla for being supportive during my MS journey.

TABLE OF CONTENTS

	Page
LIST OF TABLES	vi
LIST OF FIGURES	vii
CHAPTER	
1 INTRODUCTION	1
1.1 Overview	1
1.2 Motivation	4
1.3 Challenges	5
1.4 Contribution	6
2 BACKGROUND	7
2.1 Digital Adversarial Attacks	7
2.2 Physically Realizable Attacks	8
2.2.1 Deep Neural Networks	9
2.2.2 Deep Reinforcement Learning	9
2.2.3 Autonomous Driving	10
3 TARGETED PHYSICAL ADVERSARIAL ATTACK	12
3.1 Physical Object Dynamics	12
3.2 Environment Dynamics Model	14
3.2.1 Data Collection	14
3.2.2 Learning Procedure	15
3.3 Attack Formulation	16
3.4 Optimization	18
4 EXPERIMENTS	21
4.1 Car Driving Simulator	21
4.2 Baselines	22

CHAPTER	Page
4.3 Evaluation Metrics	22
5 RESULTS	23
5.1 Driving Scenarios	23
5.2 Baselines Comparison	23
5.3 Robustness of Object Position	27
5.4 Trade-off between Attack Strength and Attack Length	28
6 CONCLUSION AND FUTURE WORK	30
REFERENCES	32
APPENDIX	
A VIDEO RESULTS	36

LIST OF TABLES

Table		Page
5.1	Targeted and Random Attacks Visualization in Three Driving Scenarios. Agent Is Highlighted in Red Boxes. Visit Appendix for Complete Video Details.	24
5.2	Comparison with Random Noise Baseline in Terms of Evaluation Metrics. .	27
5.3	Attack Strength vs Attack Length	29

LIST OF FIGURES

Figure	Page
1.1 An Example of DNNs Being Used to Control the Vehicle from Input Image.	2
1.2 A Deep RL Framework to Learn Optimal Control Policy for AD System. . . .	2
1.3 An Illustration of Adversarial Attacks on AD System.	3
1.4 An Illustration of Physical Adversarial Attack on AD System by Planting an Adversarial Physical Object.	4
3.1 VAE and MD-RNN Architecture for Learning Dynamics Model of Envi- ronment.	16
3.2 Illustration of Targeted Physical Adversarial Attack in OpenAI Gym’s CarRacing-v0 Environment. The Blue Panel Shows the Adversary Craft- ing the Modified Observation Through Planting and Updating the Physical Object Seen by the Agent. Adversary Learns a Dynamics Model and It Is Assumed That Pre-trained Agent Policy Is Known as Shown in the Green Panel. The Orange Panel Shows the Optimization Performed by Adversary to Learn the Perturbation by Minimizing the Loss Between Prediction from Dynamics Model and a Predefined Target State.	18
5.1 Trajectories of AD Agent with No Attack, Random Attack and Optimized Attack for Straight Track Scenario.	25
5.2 Trajectories of AD Agent with No Attack, Random Attack and Optimized Attack for Left Turn Track Scenario.	25
5.3 Trajectories of AD Agent with No Attack, Random Attack and Optimized Attack for Right Turn Track Scenario.	26
5.4 Attack Robustness on Position of Physical Object During Test Time.	28

Chapter 1

INTRODUCTION

1.1 Overview

Autonomous Driving (AD) is considered a robotic task where a vehicle containing the AD system can control on its own without human intervention. To achieve this behavior, the AD system must need to perceive the surroundings and intelligently plan actions just like a human. Unlike human eyes, AD systems are equipped with different sensors such as Camera, LiDAR, and Radar that are capable of perceiving wide variety of surrounding information. In recent years, due to the advancement of computer vision, AD systems are developed mainly by using images captured from RGB cameras. These advancements are largely influenced by a fundamental technique called Deep Neural Networks (DNNs).

DNNs are heavily used in recent years for image classification (Pham *et al.*, 2021), language translation (Vaswani *et al.*, 2017), speech recognition (Baevski *et al.*, 2020) and other tasks. Inspired by this, several researchers applied DNNs for control task in AD systems. Figure 1.1 shows an example of DNNs that decide vehicle's action based on the input image. These DNNs are usually learned from a large set of data consisting of different driving examples collected by humans. This type of learning procedure to mimic human behavior is considered as Imitation Learning (IL).

Even though IL methods for building AD systems performed well (Bojarski *et al.*, 2016), they contain certain limitations such as limited dataset availability, covariate shift between training data and evaluation, and certain design choices (Osa *et al.*, 2018). To mitigate these issues, learning a control task is modeled in a different way of using reward system instead of human data which is called Reinforcement Learning (RL) (Sutton and

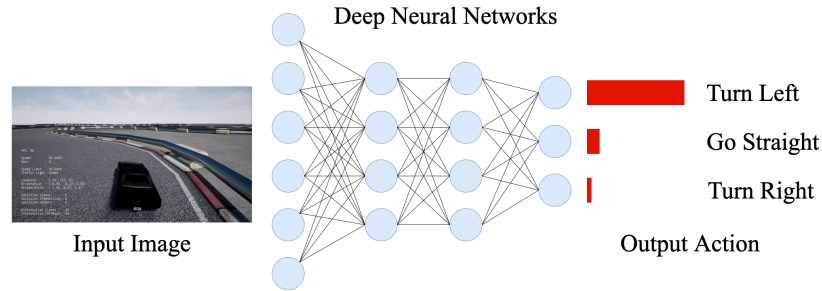


Figure 1.1: An Example of DNNs Being Used to Control the Vehicle from Input Image.

Barto, 2018). If the control actions are learned from sensor data using DNNs, it is referred as deep reinforcement learning or simply deep RL.

In a deep RL framework, the objective is to learn a optimal control behavior, often called policy that maps input images to actions. Then, the actions are performed on a environment which can be either real world or simulator. The environment then returns a reward and next image to be perceived by the AD system and thus forming a closed loop mechanism. In deep RL, the sensory information perceived is generalized as state or observation, and the system that performs the action is called agent. In our AD scenario, the state is an image and the agent is AD system. Figure 1.2 shows an AD system set up in a deep RL framework to learn an optimal control policy. Recently, several methods (Wang *et al.*, 2018; Liang *et al.*, 2018) are proposed using deep RL to solve the AD task.

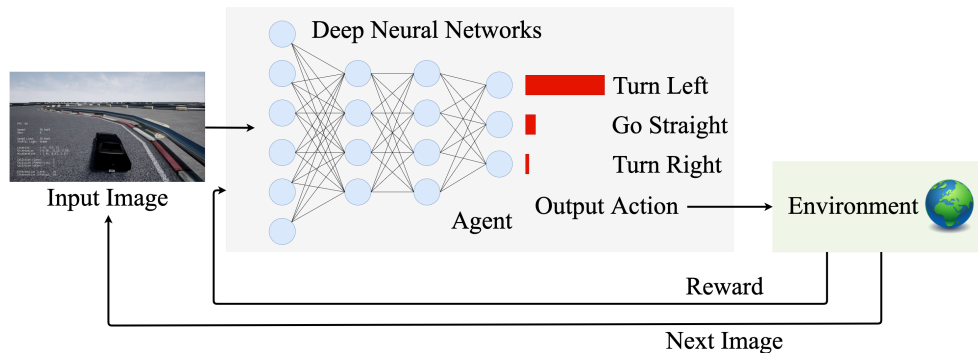


Figure 1.2: A Deep RL Framework to Learn Optimal Control Policy for AD System.

On the other side, as the progress in usage of DNNs increased, the vulnerabilities associated with them are also avidly increasing through adversarial attacks. An adversarial attack is a procedure generally initiated by attacker or adversary by perturbing small percentage of the DNNs input to generate outputs that are different from true output. For example in AD system, an adversarial attack consists of adding a small noise to input image which is indistinguishable from original image such that the action generated can be right turn instead of original left turn as shown in Figure 1.3. These attacks actively expanded to wide variety of DNN applications such as computer vision (Goodfellow *et al.*, 2015; Akhtar and Mian, 2018), natural language processing (Jia and Liang, 2017) and speech (Qin *et al.*, 2019).

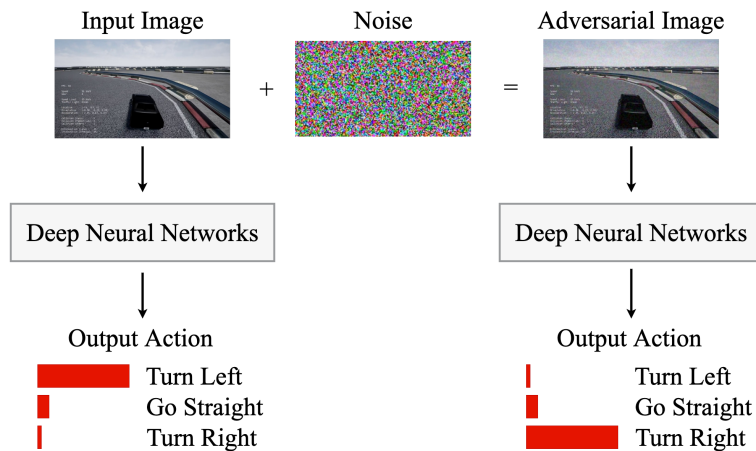


Figure 1.3: An Illustration of Adversarial Attacks on AD System.

Since DNNs are explicitly used to learn policies, deep RL agents are also susceptible to adversarial attacks (Huang *et al.*, 2017; Lin *et al.*, 2017). Further, recent study has shown that AD systems learned with deep RL can also be exploited by adversarial attacks (Huang and Wang, 2018; Sun *et al.*, 2020). However, these adversarial attacks on AD systems may not be practical as one need direct access to the agents' sensors to modify the states.

To introduce a more feasible attack, one can create and place adversarial objects in the environment to form a physically realizable attack. These attacks are shown to be effective

in general application of DNNs such as image classification (Kurakin *et al.*, 2016; Eykholt *et al.*, 2018) and face recognition (Sharif *et al.*, 2016). Further, these physical attacks can be easily extended to fool AD systems by adding an adversarial physical object beside road to take wrong action as illustrated in Figure 1.4. Lastly, few recent works empirically showed the existence of physical adversarial examples in AD system by modifying advertising sign boards (Kong *et al.*, 2020), and patterns painted on road (Yang *et al.*, 2020). The practicality of such attacks makes it interesting to research further about vulnerabilities of AD systems.

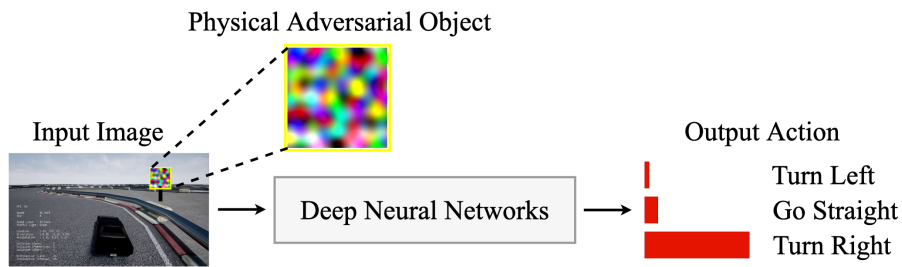


Figure 1.4: An Illustration of Physical Adversarial Attack on AD System by Planting an Adversarial Physical Object.

1.2 Motivation

In view of physical adversarial attacks on AD systems, this work aims to question the possibility of launching more controlled yet effective physical attack. To that end, this work indirectly argues and pushes to introduce a sophisticated threat that can be practical and effective. By doing this, the AD community could be enlightened with such threats and posits to make more robust and reliable deep RL algorithms.

Upon reviewing the existing methods on physical attacks (Kong *et al.*, 2020; Yang *et al.*, 2020), it is found that these attacks on AD systems are untargeted attacks, i.e., the attacker objective is to maximize the deviation between steering angle taken by agent with

and without presence of physical object. In specific, the attacker has no control over the behavior or positions of vehicle within a time window. This further questions whether we can propose an attack where the exact position of car in environment can be specified by the attacker and thereby achieving practicality, outcome control and effectiveness. Motivated by this, our work proposes to design a static perturbation on the environment such that the agent is led to a target state specified by the attacker after certain time. Learning an adversarial plan of actions implicitly by just making the agent observe physical adversarial example makes this a challenging task.

1.3 Challenges

Even though physical adversarial attack seems practical, there are few challenges that need to be addressed to achieve this. Existing digital attack threats assume that the adversary is allowed to manipulate the agent's observation at all time, such as through hijacking the camera module of AD system. However, in physical attack, the perturbation on object is fixed and it is not allowed to change at all time. Moreover, this fixed perturbation should be able to mislead the agent continuously. Hence, finding a physical adversarial pattern that need to produce maximum overall attack effect within a time window is considered to be a key challenge.

Another key challenge in implementing physical adversarial attack is that the perturbation area is limited compared to the full image seen by the agent. Existing digital attacks consider this perturbation region to take entire image whereas in physical attack, only the region corresponding to the physical object can be modified. Further, the adversarial object will change its relative position with respect to the agent during the attack, while the agent's motion is determined by the object itself and thus forming a closed loop mechanism. Hence, identifying and manipulating the object region at each step by taking agent position into account is considered to be another challenging task.

1.4 Contribution

Since this research aims to discover the vulnerabilities of control policy, the optimal policy of AD system is considered to be white-box. To solve our proposed targeted physical adversarial attack, it is assumed that the attacker can learn a differentiable dynamics model that predict the transition of the environment and has access to dynamics of agent's own states with respect to the agent's actions. This assumption holds reasonably valid since the environment (road and surroundings) is accessible to everyone including the attacker, and AD agent's dynamics is commonly available knowledge, e.g., position of car is known based on the actions it performed. Overall, the contributions of this research are as follows.

- To the best of our knowledge, this work is among the first to present targeted physical attacks on deep RL agent for AD systems. The proposed attack algorithm generates a static perturbation that can be directly realized through an object placed in the environment to mislead the agent towards a pre-specified state within a time window.
- Ablation studies are performed to show that the choices of attack time window and the target state specified are correlated. Therefore, fine-tuning the loss function of the attack with respect to the time window is crucial for finding successful attacks.
- Robustness of the attack is studied with respect to relative positions of physical adversarial object with respect to the agent, and showed that moving the object partially out of the agent's sight can reduce the attack effectiveness.

Chapter 2

BACKGROUND

Adversarial attacks on deep RL agents or DNNs in general can be broadly classified into two types based on the attacker access limits. If the attacker gains access to manipulate the image seen by DNNs, then it is considered to be a digital adversarial attack. Whereas, if the attacker can manipulate a physical object in the environment, then it is considered to be a physically realizable attack.

2.1 Digital Adversarial Attacks

Digital adversarial attacks are first introduced by Szegedy *et al.* (2014) for AlexNet (Krizhevsky *et al.*, 2012) on Imagenet data (Deng *et al.*, 2009). Later, Goodfellow *et al.* (2015) showed that adversarial examples can be generalized further across different DNNs architectures. Moreover, Nguyen *et al.* (2015) showed that certain images unrecognizable to humans can be recognized and classified by DNNs with high confidence. Further, Papernot *et al.* (2017) showed that adversarial attacks can be achieved even without access to model weights or training data and considered them as black-box attacks. Carlini and Wagner (2017) proposed highly confident adversarial examples which can even fool defensively distilled DNNs. Even though such attacks are highly focused on computer vision tasks such as classification (Goodfellow *et al.*, 2015; Carlini and Wagner, 2017), semantic segmentation (Hendrik Metzen *et al.*, 2017), and face recognition (Dong *et al.*, 2019), they seem to be existing in other domains such as natural language processing (Zhang *et al.*, 2019) and speech (Carlini and Wagner, 2018).

Since DNNs are heavily used in deep RL, researchers have identified that deep RL

agents are also vulnerable to such attacks. Huang *et al.* (2017) observed the adversarial phenomenon in policies learned through DNNs using FGSM attack proposed by Goodfellow *et al.* (2015). In the Atari games environment, Lin *et al.* (2017) proposed strategically timed attack which focuses on finding the right time when an adversarial attack needs to be performed, and enchanting attack, a targeted attack that generates adversarial examples in order to find actions that lead to a target state. Kos and Song (2017) proposed methods for generating value function based adversarial examples and Behzadan and Munir (2017) studied adversarial attacks on deep Q networks (DQN) along with transferability to different DQN models. Further, Pattanaik *et al.* (2017) proposed a gradient-based attack on double deep Q networks (DDQN) and deep deterministic policy gradient (DDPG), and developed a robust control framework through adversarial training. Recently, Weng *et al.* (2020) proposed model-based adversarial attacks on MuJoCo domains using a target state as the attack goal similar to Lin *et al.* (2017)'s enchanting attack. More recently, Zhang *et al.* (2020) proposed state-adversarial Markov decision process and studied adversarial attacks on model-free deep RL algorithms.

In the context of AD, Huang and Wang (2018) performed digital adversarial attacks with similar idea of Huang *et al.* (2017) using FGSM. Further, Sun *et al.* (2020) improved these attacks on AD using critical point attack which relies on environment prediction model and Damage Awareness Metric. However, these attacks on AD require a strong assumption that the attacker can access the camera or state observations seen by AD agent. This assumption cannot be practical and realistic in general. Hence, digital adversarial attacks have only limited effect on AD systems.

2.2 Physically Realizable Attacks

Like mentioned previously, physically realizable attacks are targeted through a physical object present on part of the input image seen by DNN. These attacks are mainly studied

on DNNs and further extended to deep RL agents. More importantly, these attacks have become interestingly dangerous to AD systems raising a serious concern.

2.2.1 *Deep Neural Networks*

In the context of fooling DNNs with physical adversarial attacks, Kurakin *et al.* (2016) manipulated images captured from cell-phone to fool Inception network (Szegedy *et al.*, 2016) trained on Imagenet data (Deng *et al.*, 2009). For the same Inception network, Athalye *et al.* (2018) extended these attacks to 3D case by generating 3D adversarial objects using Expectation Over Transformation framework. Sharif *et al.* (2016) proposed an approach to fool state-of-the-art face recognition system by generating an adversarial eye-glass frame that can cause misclassification.

Brown *et al.* (2017) proposed an universal, location and size independent physical adversarial patch which when placed on image cause models to misclassify. Thys *et al.* (2019) showed that person detection model can be fooled by generating an adversarial patch and holding in front of body. Moreover, Liu *et al.* (2018) proposed DPatch that can perform attacks on object detection algorithms such as Faster R-CNN (Ren *et al.*, 2015) and YOLO (Redmon and Farhadi, 2017). All the above discussed methods are either classifiers or detectors and are not suitable for complete sequential decision making tasks.

2.2.2 *Deep Reinforcement Learning*

Unlike DNNs, physical adversarial attacks specific to deep RL agents is a relatively new topic and only a few works exist. Since most of the physical adversarial attacks on deep RL agents are suitable on AD systems, they are mentioned in detail in section 2.2.3. Apart from attacks on deep RL AD agents, Gleave *et al.* (2020) proposed adversarial policies in a multi-agent environment where policy of attack agent (which is a physical object that can perform actions) is crafted in a way that victim policy can be damaged. These adversarial

policies are shown to be effective in zero sum games such as kick and defend, sumo fight and others. Similar to this, Lin *et al.* (2020) proposed adversarial examples for attack agent that can generate certain target actions induced by adversarial policies. The method is performed on popular Starcraft multi-agent game and it is shown to be effective in decreasing total team reward. However, these methods being in a multi-agent environment are different from the threat proposed in this work.

2.2.3 Autonomous Driving

Early work of Lu *et al.* (2017) showed that physical attacks on AD systems have limited to no impact because of continuous changes in viewing angle, distance from a moving vehicle. However, Eykholt *et al.* (2018) proposed Robust Physical Perturbations (RP2) algorithm for generating physical adversarial examples on traffic signs considering different view points. The attack is shown to be effective in a real world physical drive-by test by mounting camera on top of car. Sitawarin *et al.* (2018) proposed out of distribution attack to generate any sign to targeted adversarial example, and lenticular printing attack which generates disguised adversarial images when viewed from varying heights of camera. Further, Liu *et al.* (2019) proposed a Generative Adversarial Networks (GAN) based approach for creating physical adversarial patches that provides high visual blending with original traffic image signs and provides same attack effectiveness.

Apart from 2D images, such attacks are found to be present in 3D scans from LiDAR as well. Cao *et al.* (2019) proposed *LiDAR-Adv* that generates a 3D adversarial object when placed on the side of road can successfully cause a LiDAR detector to be unnoticed. Recently, Tu *et al.* (2020) generated universal 3D adversarial objects and placed them on top of other vehicle to fool LiDAR detectors.

All the above discussed methods in this section showed that physical threats on AD systems are intriguingly possible. These methods performed physical world experiments by

collecting videos in the presence of benign objects and adversarial objects separately and then evaluated them. Such evaluation is considered as offline analysis. However, in realistic scenarios of online driving, an AD system may undergo a different behavior in presence of adversarial object and cannot choose same path when benign object is present. For example, a perturbed stop sign misclassified as right turn can make AD system take actions leading to right turn forming new view points. There might be many such view points and identifying every one of them is not scalable. In this work, online driving scenario is considered, i.e. action changes in AD system are taken into account in presence of adversarial object. To solve this, different paths or trajectories are collected and then a dynamics model is learned to simulate the behavior of AD system under attack.

Recently, few works have considered performing an online driving testing in physical-world scenarios. Kong *et al.* (2020) proposed PhysGAN, a generative model that takes 3D video slice as input and generates a single physical adversarial example. The adversarial example is then placed on environment as advertising sign board and tested for the attack effectiveness. Further, Yang *et al.* (2020) proposed an optimization method for finding physical adversarial examples using a differential approach to map physical patterns from 3D space to 2D image space. However, these methods are untargeted attacks since their objective is to maximize the steering angle error between with and without presence of physical adversarial example. There is no guaranteed outcome of where the vehicle will reach after few timesteps. In contrast to this, our proposed work make AD agent reach a target state or goal, specifically an image set by the attacker. Hence, the targeted physical adversarial attack is a novel approach that is practical, effective and more controlled.

TARGETED PHYSICAL ADVERSARIAL ATTACK

In this chapter, targeted physical adversarial attack is proposed, i.e. the aim of attacker is to make AD agent reach a specific target state in the presence of an adversarial physical object. To achieve this, the attacker makes use of agent’s dynamics, learn a dynamics model of environment, then formulate the attack for optimization in a differentiable fashion. The following sections describe in detail each of these methods.

3.1 Physical Object Dynamics

One of key challenges in any physical adversarial attack is to obtain the position of object in the frame seen by the agent at every time step as vehicle moves. This is referred as obtaining physical object dynamics. We consider an AD system as a deep RL agent that has its own state in the environment. The agent’s state is usually defined by position of vehicle, angle and velocity information. Remember that the position of object in the frame changes relative to the agent state. For example, the agent moving forward will have object appear bigger compared to previous agent state. In this work, it is assumed that attacker have access to agent state transition dynamics. That is, given an agent state and action performed by it, attacker can know the next agent state. This assumption is reasonable since environment is accessible to all including the attacker, e.g a particular road of highway. In realistic physical world scenario, attacker will be able to use a proxy vehicle and drive around environment to infer the agent dynamics.

Before looking into obtaining physical object dynamics from agent dynamics, let’s first mathematically formalize the whole system to make things clear. Let $s_t \in [0, 1]^{w \times h \times c}$ be a

normalized image with width w , height h , and channel size c respectively, that represents the state (images) seen by agent at time t of the underlying Markov Decision Process (MDP). Let $a_t \in [0, 1]^n$ be the action vector taken by the agent at time t , and n is the number of continuous actions to be determined. In this work, the actions of AD agent consist of normalized change of steering angle, acceleration and breaking rates in that order. Let $r(s_t, a_t) \in \mathbb{R}$ be the rewards received to agent when action a_t is performed on state s_t . Let $\pi_\theta : [0, 1]^{w \times h \times c} \rightarrow [0, 1]^n$ be an optimal policy learned on the MDP with objective shown in Equation (3.1). In deep RL, θ is referred as weights or parameters of policy learned using DNNs.

$$\begin{aligned} \max_{\theta} \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_t r(s_t, a_t) \right] \\ p_\theta(\tau) = p_\theta(s_1, a_1, \dots, s_T, a_T) = p(s_1) \prod_{t=1}^T \pi_\theta(a_t | s_t) p(s_{t+1} | s_t, a_t) \end{aligned} \quad (3.1)$$

Let $\delta_t \in [0, 1]^k$ be the agent state at time t , where k is the number of properties required to make up agent state. In this experiment, δ_t is represented by position, velocity and angle of vehicle. Let $g : [0, 1]^k \times [0, 1]^n \rightarrow [0, 1]^k$ be the dynamics of the agent.

Let p_t be the representation of adversarial object relative position in the image. If the object is a rectangle, then p_t represents coordinates of bounding box. Our goal is to find p_t so that only the area within it should be modified to form the physical attack. To get p_t , first the attacker place a physical object on the side of road at location, Φ fixed with respect to environment. Then, p_t is calculated using δ_t , g and Φ using Equation (3.2), where ψ is a transformation function that maps absolute position of object onto relative position in the image.

$$\begin{aligned} p_t &= \psi(\delta_t, \Phi) \\ \delta_t &= g(\delta_{t-1}, a_{t-1}) \end{aligned} \quad (3.2)$$

By inferring p_t at each timestep, the attacker can easily identify the relative position of object in the image seen by AD agent based on actions taken to craft the physical attack.

3.2 Environment Dynamics Model

Next step in our proposed attack is to learn a dynamics model of environment. By learning such model, attacker can successfully predict the future state of agent based on the actions taken by agent and current state. This is an essential step since our attack is gradient based and hence the environment dynamics model should be differentiable. The model is learned by first collecting the data and then choosing a necessary DNN architecture.

3.2.1 Data Collection

The dataset, D for learning dynamics model is collected in the format of (state, action, next state), i.e. (s_t, a_t, s_{t+1}) through multiple rollouts of agent in the environment, env . When performing an attack, remember that a successfully attacked rollout will encounter states different from those experienced through the benign policy, e.g., AD agent moving out of the road in presence of attack is different from AD agent staying on the road without attack. To collect a representative dataset, rollouts are performed using the pretrained policy with added noise of variable strength, ρ added to the actions i.e, $a_t = a_t + \mathcal{N}(0, 1) * \rho$. The noisy actions help explore the environment, allowing the adversary to predict the environment dynamics correctly when approaching the target state. The full data collection procedure is shown in Algorithm 1.

The resultant dataset is denoted by $D = \{(s_i, a_i, s_{i+1})\}_{i=1}^N$ with $N = r_n * r_l$, where r_n is number of rollouts and r_l is length of each rollout. Note that such data collection is achievable when launching a real-world attack, as long as the attacker can sample the state transitions towards the specified target by using a vehicle with dynamics similar to the attacked agent. Hence, the proposed attack corresponds to online driving case unlike most of the existing methods since the dynamics model is used to account for the changes in trajectory of AD agent under attack.

Algorithm 1: Collect D from env by rolling the AD agent

Input: environment env , number of rollouts r_n , length of each rollout r_l **Output:** a collection of state, action and next state dataset, D $r \leftarrow 0, l \leftarrow r_l$ $D \leftarrow \phi$ seed \leftarrow random seed**while** $r < r_n$ **do** $t \leftarrow 0$ $env.seed(seed)$ $s_t = env.reset()$ **while** $t < l$ **do** $a_t = \pi(s_t)$ $a_t = a_t + \mathcal{N}(0, 1) * \rho$ $s_{t+1} \leftarrow env.step(a_t)$ $D \cup \{(s_t, a_t, s_{t+1})\}$ $t \leftarrow t + 1$ $r \leftarrow r + 1$ **Return** D

3.2.2 Learning Procedure

After collecting the dataset, the next step is to learn a well behaved dynamics model to facilitate attack optimization. Since the environment states contains rich information like time-variant track and surroundings, traditional feed forward neural networks fail to perform well on the collected AD system dataset. Hence in this work, we follow Ha and Schmidhuber (2018) to construct a dynamics model using a Variational AutoEncoder (VAE) and a Mixture-Density Recurrent Neural Network (MD-RNN), combinely denoted by $f(s_t, a_t; w)$, where w are trainable parameters. The dynamics model takes in the current environment state, s_t and action performed, a_t as input, and predicts the next environment

state, s_{t+1} as output. Since the state image in our work is of shape 96×96 , the architectures of VAE and MD-RNN are tweaked a bit according to our convenience. Figure 3.1 shows the architecture of dynamics model used to realize our proposed attack.

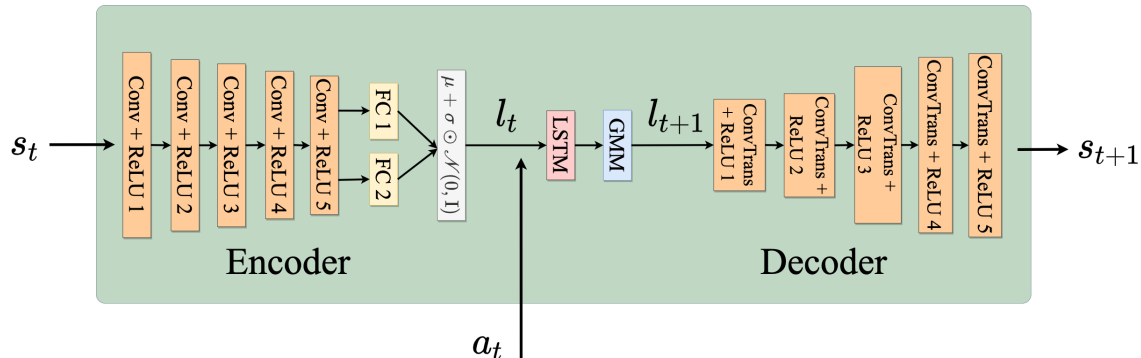


Figure 3.1: VAE and MD-RNN Architecture for Learning Dynamics Model of Environment.

As in the original paper Ha and Schmidhuber (2018), we use the same combination of Mean Square Error and Kullback–Leibler divergence as the loss for training the VAE, and the Gaussian Mixture Loss for training the MD-RNN.

3.3 Attack Formulation

Once the agent dynamics is known and dynamics model of environment is learned, our proposed attack can be realized in end-to-end differentiable fashion. To achieve this let's consider ν be the matrix of 1's. Recall that, we have p_t available as explained in section 3.1. A mask $m_{w_t} \in \{0, 1\}^{w \times h}$ is created based on p_t and ν by estimating homography and warping using *warp* function. m_{w_t} only has 1s within the region occupied by the physical object. To optimize adversarial perturbations, the *warp* function need to be differentiable and hence traditional OpenCV functions cannot be applied. In our work, we used a differentiable warping function available in Kornia library (Riba *et al.*, 2020).

Let $\Delta s \in [0, 1]^{a \times b}$ be the adversarial perturbation corresponding to the physical object. Then, a transformed adversarial image $m p_t \in [0, 1]^{w \times h}$ is created based on p_t and Δs , using

similar method of estimating homography and warping. Lastly, the adversarial image is integrated into the state image view seen by the agent using Equation (3.3) to form modified state image, s_{mt} .

$$s_{mt} = s_t \odot (1 - mw_t) + mw_t \odot mp_t \quad (3.3)$$

where \odot is the element-wise product.

Given the initial state of environment s_0 , the initial agent state δ_0 , the pre-trained policy π , the dynamics model $f(s_t, a_t; w)$, agent dynamics $g(\delta_t, a_t)$, and the transformation function, $\psi(\delta_t, \Phi)$, we search for an image Δs , with $\|\Delta s\|_\infty \leq \epsilon$, that leads the agent to a specific target state, s'_{target} within the time window $[0, T]$. Mathematically, the attack objective is defined in Equation (3.4).

$$\begin{aligned} \min_{\|\Delta s\|_\infty \leq \epsilon} \quad & \sum_{t=1}^T \|s_t - s_{target}\|_2^2 \\ \text{s.t.} \quad & a_t = \pi(s_{mt}), \\ & s_{mt} = s_t \odot (1 - mw_t) + mw_t \odot mp_t, \\ & mw_t = \text{warp}(v, p_t), \\ & mp_t = \text{warp}(\Delta s, p_t), \\ & p_t = \psi(\delta_t, \Phi), \\ & s_{t+1} = f(s_t, a_t), \\ & \delta_{t+1} = g(\delta_t, a_t). \end{aligned} \quad (3.4)$$

The dependency of variables involved in this problem is clearly visualized in Figure 3.2. The loss function of Equation (3.4) accepts early convergence of AD agent reaching the target state. Notice that, we use states without the adversarial perturbation in evaluating the loss, since the target state is specified before the attack problem is solved. The use of the learned dynamics model, agent's dynamics and a differentiable implementation of *warp* together make this formulation differentiable with respect to the perturbation Δs , allowing

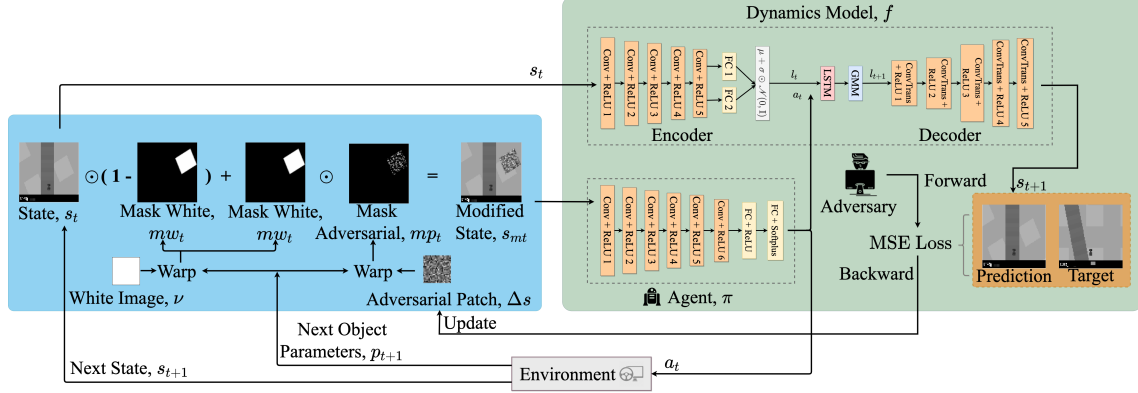


Figure 3.2: Illustration of Targeted Physical Adversarial Attack in OpenAI Gym’s CarRacing-v0 Environment. The Blue Panel Shows the Adversary Crafting the Modified Observation Through Planting and Updating the Physical Object Seen by the Agent. Adversary Learns a Dynamics Model and It Is Assumed That Pre-trained Agent Policy Is Known as Shown in the Green Panel. The Orange Panel Shows the Optimization Performed by Adversary to Learn the Perturbation by Minimizing the Loss Between Prediction from Dynamics Model and a Predefined Target State.

the problem to be solved using any gradient-based methods.

3.4 Optimization

The complete optimization procedure of our targeted physical adversarial attack formulated in Equation (3.4) is outlined in Algorithm 2. During each iteration, the modified state containing the adversarial image s_{mt} is calculated as described in Equation (3.3) by computing mp_t and mw_t . To respect the observation limits seen by agent, we clip s_{mt} between 0 and 1 so that valid image is yielded. The agent then performs an action on s_{mt} to get a_t . Using the dynamics model, f , future prediction s_{t+1}^\dagger is obtained to compute the loss. Finally, we backpropagate the sum of losses within the time window $[0, T]$ in order to update perturbation Δs .

Algorithm 2: Optimization for Targeted Physical Adversarial attack

Input: Number of Iterations, I , environment env , Attack length, T , pretrained

policy π , dynamics model, f , target state s_{target}

Output: learned physical perturbation example, Δs

$i \leftarrow 0$, seed \leftarrow random seed

$\Delta s \leftarrow \mathcal{N}(0, 1)$

while $i < I$ **do**

total_loss $\leftarrow 0$, $t \leftarrow 0$

$env.seed(seed)$

$s_t = env.reset()$

$\delta_t \leftarrow$ initial agent state

while $t < T$ **do**

$p_t = \psi(\delta_t, \Phi)$

$mw_t, mp_t = warp(v, p_t), warp(\Delta s, p_t)$

$s_{mt} = s_t \odot (1 - mw_t) + mw_t \odot mp_t$

clip s_{mt} between $[0, 1]$

$a_t = \pi(s_{mt})$

$s_{t+1}^\dagger = f(s_t, a_t)$

$\delta_{t+1} = g(\delta_t, a_t)$

total_loss += $d(s_{t+1}^\dagger, s_{target})$

$s_{t+1} \leftarrow env.step(a_t)$

$t \leftarrow t + 1$

backpropogate total_loss to update Δs

clip Δs between $[-\epsilon, \epsilon]$

$i \leftarrow i + 1$

Return Δs

During evaluation time, the obtain Δs is pasted on to the object to form a adversarial physical object. Then AD agent is then run in the presence of this adversarial object to evaluate the behavior. In next chapter, experimental results are provided to empirically show that our proposed approach is successfully able to make AD agent reach pre-specified target state set by the attacker.

Chapter 4

EXPERIMENTS

In this chapter, our planned experiments are discussed to evaluate and show the effectiveness of our proposed attack approach. First we show the experimental setting on one of the popular AD systems in deep RL and then consider baselines to compare our work with. Finally, we propose few evaluation metrics to quantify the effectiveness of our attack.

4.1 Car Driving Simulator

We use the CarRacing-v0 environment introduced by Klimov (2016) from OpenAI Gym to demonstrate the existence of adversarial objects that misguide deep RL agent of AD systems. We used a model-free Actor-Critic algorithm (Mnih *et al.*, 2016) to obtain the optimal policy π . The policy is trained with a batch size of 128 and 10^5 episodes. The policy takes a stack of 4 consecutive states (grayscale images of size 96×96) as the input and produces continuous actions $a \in \mathbb{R}^3$ as the output. Each of these action dimensions represent steering angle, acceleration and brake respectively. Steering action takes values between $[-1, 1]$ with -1 being maximum left turn, 1 being maximum right turn and 0 being no turn. Acceleration and brake actions both takes values between 0 and 1 . After training, the policy achieves an average reward of 860 when evaluated on 100 test episodes.

We collected a dataset consisting of 10^3 rollouts with length of each rollout being 80 by running agent with pretrained plus noise actions on environment. For the dynamics model f of the environment, the VAE is trained for 10^3 epochs using the Adam optimizer (Kingma and Ba, 2014). We set the batch size to 32 and learning rate to 0.001 with decreasing learning rate based on plateau and early stopping. For the MD-RNN, we train for 10^3

epochs using the same optimizer. We set the batch size to 16, the number of Gaussian models to 5, and the learning rate to the same value as the training of VAE.

For the attack optimization, we set the time span T to 25 and the adversarial bound to $\epsilon = 0.9$. An ablation study is shown in depth on these hyperparameters in section 5.4. We use the same optimizer as before, and set the learning rate to 0.005 for $I = 10^3$ iterations. We set the adversarial object area to be 25 pixels wide and 30 pixels tall.

4.2 Baselines

To our best knowledge, there has been only a few results on targeted physical attacks on deep RL agents of AD systems. Therefore, we use a baseline where Δs is drawn uniformly in $[0, 1]^{25 \times 30}$. For quantitative evaluations, we performed random baseline experiment for 10 times and averaged the result to compare. By comparing agent state trajectories, final state visualizations after T timesteps in the presence of random and optimized Δs , we will show that the proposed attack is more effective than random perturbations.

4.3 Evaluation Metrics

We introduce two metrics namely *actions error* and *change of value* to quantitatively evaluate the effectiveness of our attack. The *actions error* is defined as the average of squared l_2 distance between the attacked and unattacked actions taken by the agent when performed rollouts with and without the adversarial object, respectively.

Change of value is defined as the percentage change of sum of rewards received to the AD agent between rollouts performed with and without adversarial object. Both metrics are calculates over time period, $[0, T]$. Remember that even though we haven't considered rewards into account for formulating our attack, a successful attack may be indirectly construed as an abnormal behavior of AD agent and hence reduction of reward is inevitable. This shows the strength of our attack without optimizing for reward reduction.

Chapter 5

RESULTS

To show that our approach works across different set of circumstances, we evaluate our proposed attack approach on three driving scenarios, and compare the visual states after T timesteps, trajectories of agent with and without the attack. We also compare our attack with baselines to show that our method outperforms in terms of quantitative and qualitative results. Further, we conduct experiments to evaluate the robustness of our attack with respect to different locations of adversarial object placement in the environment. Finally, we compare the effectiveness of the attack by showing the trade-off between varying attack strength, i.e., different adversarial bounds ϵ , and varying timespan of attack, T based on the evaluation metrics.

5.1 Driving Scenarios

We consider three driving scenarios where the agent with the unattacked policy will go straight, left, and right, respectively. In each of the scenarios, the object is placed at a fixed location in the environment so that it is observable by the agent throughout the attack. We specify the target states in such a way that AD agent goes out of the track. The corresponding images for these target states are shown in the last row of Table 5.1.

5.2 Baselines Comparison

We compare the effectiveness of our attack by comparing final states reached by the agent under original policy, the proposed attack, and the random baseline attack. The visualization of final states are shown in Table 5.1. Also, trajectory visualizations are

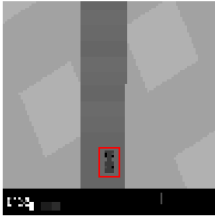
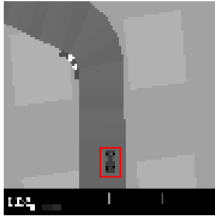
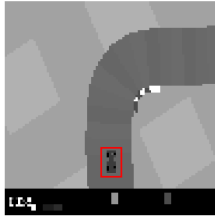
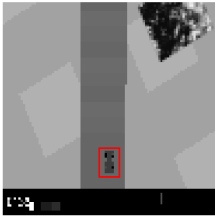
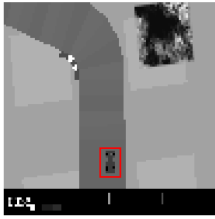
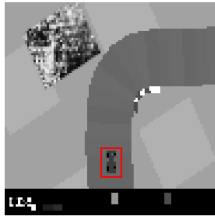
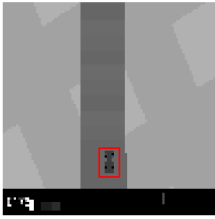
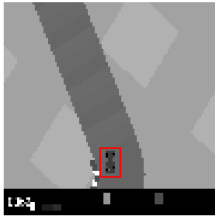
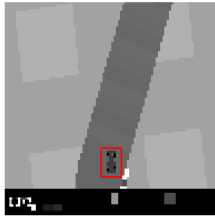
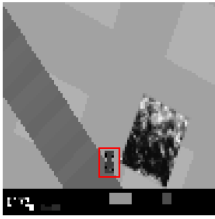
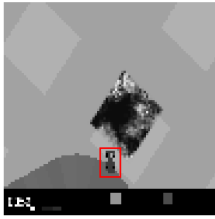
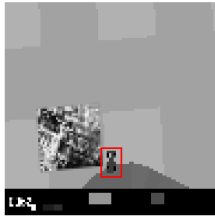

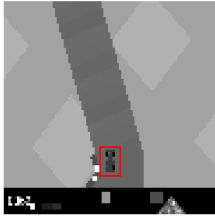
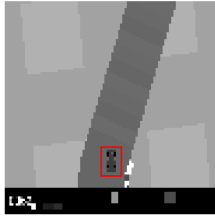
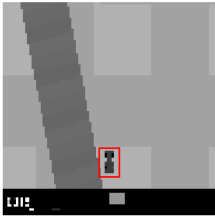

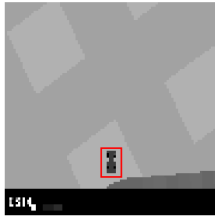
	Straight	Left turn	Right turn
$(t = 0)$ No attack			
Optimal attack			
$(t = T)$ No attack			
Optimal attack			
Random attack			
Target state			

Table 5.1: Targeted and Random Attacks Visualization in Three Driving Scenarios. Agent Is Highlighted in Red Boxes. Visit Appendix for Complete Video Details.

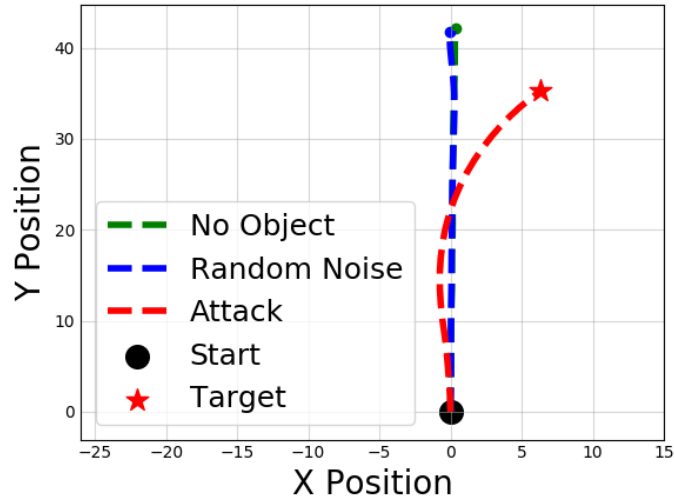


Figure 5.1: Trajectories of AD Agent with No Attack, Random Attack and Optimized Attack for Straight Track Scenario.

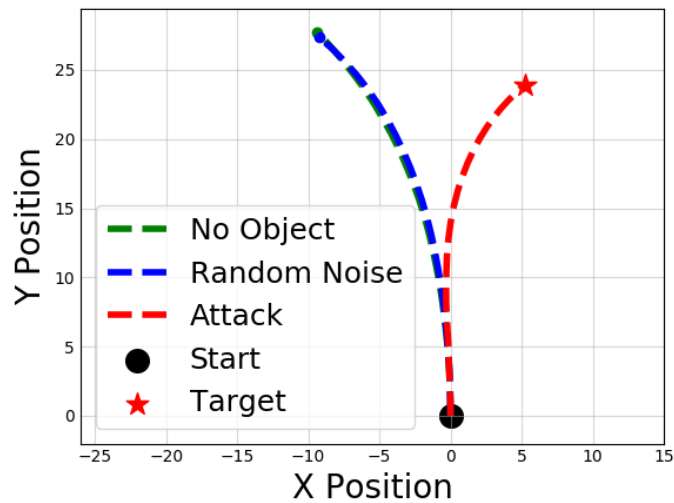


Figure 5.2: Trajectories of AD Agent with No Attack, Random Attack and Optimized Attack for Left Turn Track Scenario.

plotted and shown in Figure 5.1 for straight track scenario, Figure 5.2 for left track, and Figure 5.3 for right track scenario respectively. For the random attack, we conducted 10 independent simulations for each scenario to derive the mean trajectories. The standard

deviations in all three scenarios are negligible. X and Y axes in trajectory figures represent the global coordinates.

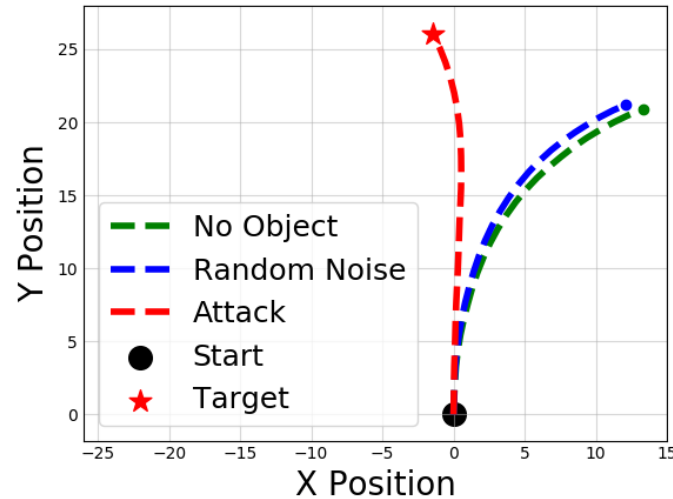


Figure 5.3: Trajectories of AD Agent with No Attack, Random Attack and Optimized Attack for Right Turn Track Scenario.

From the results it is evident that while our approach successfully misguides the AD agent in all scenarios, the agent is not affected as much by the random attacks. Specifically, in scenario 1, the agent goes straight with and without the presence of a random attack. In the presence of the proposed attack, however, the agent deviates from its intended trajectory to reach the target state. Similar behavior is observed for scenarios 2 and 3. It is worth noting that by comparing Table 5.1 and Figure 5.1, the agent reaches the target *location* but does not perfectly match the target *orientation*. Further exploration of the attack objective may potentially improve the matching of the orientation.

Lastly, Table 5.2 quantifies the comparison of our attack through the evaluation metrics. The proposed attack outperforms the random baseline on both the metrics.

Scenarios	Actions Error	Change of value (%)
Straight + Random	0.064	0
Left turn + Random	0.069	0
Right turn + Random	0.046	-10.72
Straight + Proposed	0.126	-17.70
Left turn + Proposed	0.138	-32.26
Right turn + Proposed	0.062	-32.15

Table 5.2: Comparison with Random Noise Baseline in Terms of Evaluation Metrics.

5.3 Robustness of Object Position

In this section, we study the robustness of our attack with respect to different global coordinates of the physical object (Φ) placed in the environment. The experiment is performed on the straight track scenario, with fixed dynamics model. In Figure 5.4, the (x, y) coordinates represent the relative position of the physical adversarial object with respect to the original attack (i.e., the one used in the experiments shown in Table 5.1), and the heat map represents the attack loss of Equation (3.4) relative to the original final loss during attack optimization, when the object is moved accordingly.

Therefore areas with blue region represent more successful attacks whereas with green region represent relatively unsuccessful attacks. Note that the ranges of the figure is bounded by the constraints that the object cannot be on the track and cannot be out of the scene. From this test, if the object is moved towards the track ($-X$ direction in the figure), the attack will still be effective. On the other hand, if the object is moved away from the track and partially out of the scene in subsequent time frames, the attack becomes less effective, which is reasonable since the agent will have only partial observation of the attack. Further investigating formulations of robust attacks will be valuable.

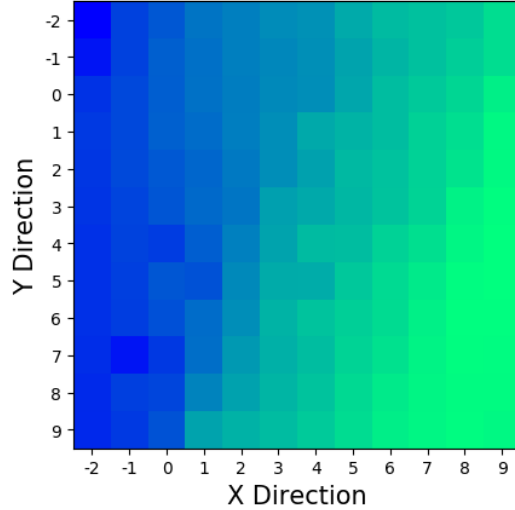


Figure 5.4: Attack Robustness on Position of Physical Object During Test Time.

5.4 Trade-off between Attack Strength and Attack Length

In this section, we perform an ablation study on the attack strength which is the adversarial bound, ϵ and attack time span, T , by enumerating $\epsilon \in \{0.1, 0.3, 0.5, 0.9\}$ and $T \in \{15, 25, 30\}$. The results in terms of the optimal loss of Equation (3.4), and the *actions error* metric are summarized in Table 5.3.

The experiments are performed on the straight track scenario, with fixed dynamics model. From the results, it is evident that larger ϵ improves the effectiveness of the attacks, i.e, increasing the ϵ value makes AD agent reach closer to the target state. Additionally, as we enlarge the time window, the *actions error* decreases in nearly all cases. Based on our experiments, we believe that if T is smaller, then the attack has a smaller action space to plan on, causing it to alter the actions more aggressively than a bigger T .

However, the attack loss increases from $T = 25$ to $T = 30$. By examining the results, we found that this is primarily because the attack object moves out of the scene between $T = 25$ and $T = 30$. As a result of the limited observation of the attack object to the agent, the optimizer struggles to find a way to keep the agent close to the target state, and thus the

Adversarial Bound ϵ	Attack Length T					
	$T = 15$		$T = 25$		$T = 30$	
	Attack Loss	Actions Error	Attack Loss	Actions Error	Attack Loss	Actions Error
0.1	0.091	0.064	0.090	0.064	0.088	0.063
0.3	0.088	0.078	0.087	0.069	0.085	0.066
0.5	0.086	0.113	0.077	0.107	0.083	0.070
0.9	0.081	0.125	0.076	0.126	0.078	0.093

Table 5.3: Attack Strength vs Attack Length

increased loss. This implies that the time window, T is coupled with the choice of the target state, and its careful selection is important for succeeding the attack.

CONCLUSION AND FUTURE WORK

Even though AD agents have been on rise using deep RL techniques, it is possible that they can be fooled by simply placing an adversarial object in the environment. While previous studies in this domain focused on untargeted attacks without long-term effects, this paper studies the *existence* of static adversarial objects that can continuously misguide a deep RL agent towards a target state within a time window. Also, unlike most of the existing methods which consider offline evaluation for physical attacks, our method considers the behavior of AD agent under presence of adversarial object during attack optimization and performs a realistic online driving scenario.

Using a standard simulator and a pretrained policy, we developed an algorithm for searching such targeted physical attacks and showed their existence empirically. For effective search of the attacks, we utilize differentiable dynamics model of the environment, which can be learned through experience collected by the attacker. Further, we also empirically verified and tested for the optimal parameters such as attack length and attack strength to produce a maximum success of attack. By demonstrating the existence of such new type of attack more practical than digital perturbations, we hope this study can motivate more rigorous research towards robust and safe AI methods for AD systems.

There are mainly two opposite and equally important future directions that can be evolved from this work. They are stronger attacks and robust defenses. In terms of stronger attacks, the improvements could be studying the existence of our targeted physical attack in more complex environments like richer visuals and 3D observations. Another interesting direction is to utilize reward function and incorporating them into attack formulation. Further, these attacks can be extended to multi-agent setting where one is a victim agent

and other can be attack agent. Studying these different attacks can provide useful insights into behavior of AD systems.

In terms of defending against such adversarial attacks, adversarial training can be introduced which has shown decent results for obtaining robust deep RL agents. Further, it is equally important for the creators of AD systems to protect their agents and software information confidential and not let it exploited by the attackers.

REFERENCES

- Akhtar, N. and A. Mian, “Threat of adversarial attacks on deep learning in computer vision: A survey”, *IEEE Access* **6**, 14410–14430 (2018).
- Athalye, A., L. Engstrom, A. Ilyas and K. Kwok, “Synthesizing robust adversarial examples”, in “Proceedings of the 35th International Conference on Machine Learning”, pp. 284–293 (PMLR, 2018).
- Baevski, A., Y. Zhou, A. Mohamed and M. Auli, “wav2vec 2.0: A framework for self-supervised learning of speech representations”, in “Advances in Neural Information Processing Systems”, vol. 33, pp. 12449–12460 (2020).
- Behzadan, V. and A. Munir, “Vulnerability of deep reinforcement learning to policy induction attacks”, in “International Conference on Machine Learning and Data Mining in Pattern Recognition”, pp. 262–275 (Springer, 2017).
- Bojarski, M., D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, “End to end learning for self-driving cars”, arXiv preprint arXiv:1604.07316 (2016).
- Brown, T. B., D. Mané, A. Roy, M. Abadi and J. Gilmer, “Adversarial patch”, arXiv preprint arXiv:1712.09665 (2017).
- Cao, Y., C. Xiao, D. Yang, J. Fang, R. Yang, M. Liu and B. Li, “Adversarial objects against lidar-based autonomous driving systems”, arXiv preprint arXiv:1907.05418 (2019).
- Carlini, N. and D. Wagner, “Towards evaluating the robustness of neural networks”, in “2017 IEEE Symposium on Security and Privacy (SP)”, pp. 39–57 (IEEE, 2017).
- Carlini, N. and D. Wagner, “Audio adversarial examples: Targeted attacks on speech-to-text”, in “2018 IEEE Security and Privacy Workshops (SPW)”, pp. 1–7 (IEEE, 2018).
- Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database”, in “2009 IEEE Conference on Computer Vision and Pattern Recognition”, pp. 248–255 (IEEE, 2009).
- Dong, Y., H. Su, B. Wu, Z. Li, W. Liu, T. Zhang and J. Zhu, “Efficient decision-based black-box adversarial attacks on face recognition”, in “Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition”, pp. 7714–7722 (2019).
- Eykholt, K., I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno and D. Song, “Robust physical-world attacks on deep learning visual classification”, in “Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition”, pp. 1625–1634 (2018).
- Gleave, A., M. Dennis, C. Wild, N. Kant, S. Levine and S. Russell, “Adversarial policies: Attacking deep reinforcement learning”, in “International Conference on Learning Representations”, (2020), URL <https://openreview.net/forum?id=HJgEMpVFwB>.

- Goodfellow, I., J. Shlens and C. Szegedy, “Explaining and harnessing adversarial examples”, in “International Conference on Learning Representations”, (2015).
- Ha, D. and J. Schmidhuber, “Recurrent world models facilitate policy evolution”, in “Advances in Neural Information Processing Systems”, vol. 31 (2018).
- Hendrik Metzen, J., M. Chaithanya Kumar, T. Brox and V. Fischer, “Universal adversarial perturbations against semantic image segmentation”, in “Proceedings of the IEEE International Conference on Computer Vision”, pp. 2755–2764 (2017).
- Huang, S., N. Papernot, I. Goodfellow, Y. Duan and P. Abbeel, “Adversarial attacks on neural network policies”, arXiv preprint arXiv:1702.02284 (2017).
- Huang, Y. and S.-h. Wang, “Adversarial manipulation of reinforcement learning policies in autonomous agents”, in “2018 International Joint Conference on Neural Networks (IJCNN)”, pp. 1–8 (IEEE, 2018).
- Jia, R. and P. Liang, “Adversarial examples for evaluating reading comprehension systems”, in “Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing”, pp. 2021–2031 (2017).
- Kingma, D. P. and J. Ba, “Adam: A method for stochastic optimization”, arXiv preprint arXiv:1412.6980 (2014).
- Klimov, O., “Carracing-v0”, in “<http://gym.openai.com/>”, (2016).
- Kong, Z., J. Guo, A. Li and C. Liu, “Physgan: Generating physical-world-resilient adversarial examples for autonomous driving”, in “Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition”, pp. 14254–14263 (2020).
- Kos, J. and D. Song, “Delving into adversarial attacks on deep policies”, arXiv preprint arXiv:1705.06452 (2017).
- Krizhevsky, A., I. Sutskever and G. E. Hinton, “Imagenet classification with deep convolutional neural networks”, in “Advances in Neural Information Processing Systems”, vol. 25 (2012).
- Kurakin, A., I. Goodfellow and S. Bengio, “Adversarial examples in the physical world”, arXiv preprint arXiv:1607.02533 (2016).
- Liang, X., T. Wang, L. Yang and E. Xing, “Cirl: Controllable imitative reinforcement learning for vision-based self-driving”, in “Proceedings of the European Conference on Computer Vision (ECCV)”, pp. 584–599 (2018).
- Lin, J., K. Dzeparoska, S. Q. Zhang, A. Leon-Garcia and N. Papernot, “On the robustness of cooperative multi-agent reinforcement learning”, in “2020 IEEE Security and Privacy Workshops (SPW)”, pp. 62–68 (IEEE, 2020).
- Lin, Y.-C., 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).

- Liu, A., X. Liu, J. Fan, Y. Ma, A. Zhang, H. Xie and D. Tao, “Perceptual-sensitive gan for generating adversarial patches”, in “Proceedings of the AAAI conference on artificial intelligence”, vol. 33, pp. 1028–1035 (2019).
- Liu, X., H. Yang, Z. Liu, L. Song, H. Li and Y. Chen, “Dpatch: An adversarial patch attack on object detectors”, arXiv preprint arXiv:1806.02299 (2018).
- Lu, J., H. Sibai, E. Fabry and D. Forsyth, “No need to worry about adversarial examples in object detection in autonomous vehicles”, arXiv preprint arXiv:1707.03501 (2017).
- Mnih, V., A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver and K. Kavukcuoglu, “Asynchronous methods for deep reinforcement learning”, in “Proceedings of The 33rd International Conference on Machine Learning”, (2016).
- Nguyen, A., J. Yosinski and J. Clune, “Deep neural networks are easily fooled: High confidence predictions for unrecognizable images”, in “Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition”, pp. 427–436 (2015).
- Osa, T., J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel and J. Peters, “An algorithmic perspective on imitation learning”, arXiv preprint arXiv:1811.06711 (2018).
- Papernot, N., 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 (2017).
- Pattanaik, A., Z. Tang, S. Liu, G. Bommannan and G. Chowdhary, “Robust deep reinforcement learning with adversarial attacks”, arXiv preprint arXiv:1712.03632 (2017).
- Pham, H., Z. Dai, Q. Xie, M.-T. Luong and Q. V. Le, “Meta pseudo labels”, in “IEEE Conference on Computer Vision and Pattern Recognition”, (2021).
- Qin, Y., N. Carlini, G. Cottrell, I. Goodfellow and C. Raffel, “Imperceptible, robust, and targeted adversarial examples for automatic speech recognition”, in “Proceedings of the 36th International Conference on Machine Learning”, pp. 5231–5240 (PMLR, 2019).
- Redmon, J. and A. Farhadi, “Yolo9000: better, faster, stronger”, in “Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition”, pp. 7263–7271 (2017).
- Ren, S., K. He, R. Girshick and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks”, in “Advances in Neural Information Processing Systems”, vol. 28 (2015).
- Riba, E., D. Mishkin, D. Ponsa, E. Rublee and G. Bradski, “Kornia: an open source differentiable computer vision library for pytorch”, in “Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision”, pp. 3674–3683 (2020).
- Sharif, M., S. Bhagavatula, L. Bauer and M. K. Reiter, “Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition”, in “Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security”, pp. 1528–1540 (2016).

- Sitawarin, C., A. N. Bhagoji, A. Mosenia, M. Chiang and P. Mittal, “Darts: Deceiving autonomous cars with toxic signs”, arXiv preprint arXiv:1802.06430 (2018).
- Sun, J., T. Zhang, X. Xie, L. Ma, Y. Zheng, K. Chen and Y. Liu, “Stealthy and efficient adversarial attacks against deep reinforcement learning”, in “Proceedings of the AAAI Conference on Artificial Intelligence”, vol. 34, pp. 5883–5891 (2020).
- Sutton, R. S. and A. G. Barto, *Reinforcement Learning: An Introduction* (The MIT Press, 2018), second edn., URL <http://incompleteideas.net/book/the-book-2nd.html>.
- Szegedy, C., V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, “Rethinking the inception architecture for computer vision”, in “Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition”, pp. 2818–2826 (2016).
- Szegedy, C., W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow and R. Fergus, “Intriguing properties of neural networks”, in “International Conference on Learning Representations”, (2014).
- Thys, S., W. Van Ranst and T. Goedemé, “Fooling automated surveillance cameras: adversarial patches to attack person detection”, in “Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops”, (2019).
- Tu, J., M. Ren, S. Manivasagam, M. Liang, B. Yang, R. Du, F. Cheng and R. Urtasun, “Physically realizable adversarial examples for lidar object detection”, in “Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition”, pp. 13716–13725 (2020).
- Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser and I. Polosukhin, “Attention is all you need”, in “Advances in Neural Information Processing Systems”, vol. 30 (2017).
- Wang, S., D. Jia and X. Weng, “Deep reinforcement learning for autonomous driving”, arXiv preprint arXiv:1811.11329 (2018).
- Weng, T.-W., K. D. Dvijotham, J. Uesato, K. Xiao, S. Gowal, R. Stanforth and P. Kohli, “Toward evaluating robustness of deep reinforcement learning with continuous control”, in “International Conference on Learning Representations”, (2020).
- Yang, J., A. Bolor, A. Chakrabarti, X. Zhang and Y. Vorobeychik, “Finding physical adversarial examples for autonomous driving with fast and differentiable image compositing”, arXiv preprint arXiv:2010.08844 (2020).
- Zhang, H., H. Chen, C. Xiao, B. Li, M. Liu, D. Boning and C.-J. Hsieh, “Robust deep reinforcement learning against adversarial perturbations on state observations”, in “Advances in Neural Information Processing Systems”, vol. 33, pp. 21024–21037 (2020).
- Zhang, H., H. Zhou, N. Miao and L. Li, “Generating fluent adversarial examples for natural languages”, in “Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics”, pp. 5564–5569 (2019).

APPENDIX A
VIDEO RESULTS

Since this work is applied on Spatio-temporal AD systems, it is easier to infer the results in the form of short videos which can be accessed from [here](#). Following are the preferred and not necessarily only software and operating system requirements to view the video results.

Operating System: macOS

Software: QuickTime Player