Adverse Behavior of Self Driving Models

Mithun Bharadwaj
University of Maryland
College Park, MD 20740
mithun02@umd.edu

Saurav Kumar
University of Maryland
College Park, MD 20740
skumar17@umd.edu

Kumar Gaurav
University of Maryland
College Park, MD 20740
kgaurav@umd.edu

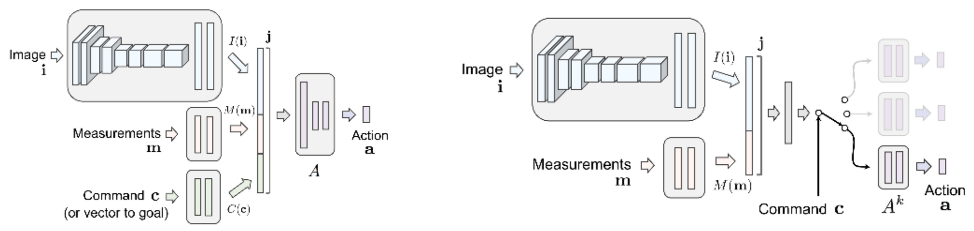
Abstract

The advent of deep learning in recent times has given rise to a range of high-stakes applications such as autonomous driving, fraud detection, medical imaging, etc. For a wide host of technical reasons, high-fidelity explanation of most AI decisions are not possible. This creates a huge space for unreliable and susceptible behavior of AI. Specifically, the autonomous driving application requires real time perception. Deep Learning for perception has been shown to be vulnerable to adversarial attacks. Usually these attacks involve carefully constructed adversarial examples through small perturbation of values at the pixel level. The preferred version of an AI system is end-to-end deep learning, where the network takes the raw input and processes it to the output of required format. For example, in an autonomous driving system, the inputs will be sensor data (camera, LIDAR, etc) and output will be the steering angles. Here we try to explore easily realizable physical adversarial attacks on such end-to-end systems that have potentially high costs. We also attempt to estimate the deviation of the vehicle from safe behavior and classify it as adverse behavior or not.

1 Introduction

While neural networks (NNs) have achieved remarkable performance and accomplished unprecedented breakthroughs in many machine learning tasks, recent studies have highlighted their lack of robustness against adversarial perturbations[1][2] Given a self driving end to end model (possibly with a deep and complicated network architecture), we are mainly interested in realistic attacks on outputs produced that occur after training, which are especially relevant for an autonomous navigation system of driver-less cars. We are perturbing the images perceived by the cameras with realistic shadows or marks on roads, which can be easily neglected by humans but may trick the system to perceive them as lane.

We systematically study a particular class of attacks, where mono-chromatic black lines of different intensities and width are projected on the road. These are unsuspicious since they are semantically inconsequential (few human drivers would be confused) and similar to common imperfections seen in routine day to day life such as tread marks, worn lane pitch, and shadows. Furthermore, we demonstrate a systematic approach for implementing such attacks so as to maximize infraction, and demonstrate actual physical impact (lane violations and crashes) over a variety of scenarios. We are using state-of-the-art end-to-end deep learning based CARLA Imitation Learning network[3] to showcase such adversarial attacks. We consider scenarios where the agent approaches a junction and has to turn right and the aforementioned projection may be perceived as lane or obstacle by the network.



(a) (b)

Figure 1: State-of-the-art end-to-end deep learning based CARLA Imitation Learning network

2 Background and Related work

End-to-end (e2e) learning models comprises of DNNs that accept raw input parameters in one end and directly calculate the desired output at the other end. Rather than explicitly decomposing a complex problem into its constituent parts and solving them separately, e2e models directly generate the output from the inputs. It is achieved by applying gradient based learning to the system as a whole. Recently, e2e models have been shown to have good performance in the domain of autonomous vehicles [4], where the forward facing camera input can be directly translated to control (steer, throttle and brake) commands.

The first imitation learning algorithm applied to autonomous driving was 30 years ago, when ALVINN system [5] used a 3-layer neural network to perform lane keeping from raw camera sensor data. With recent development in deep learning, Carla has deployed two end-to-end driving system using deep convolutional neural networks, Conditional Imitation Learning (IL) and Reinforcement Learning (RL) [3] which exhibit good lane following behaviors even in challenging environments. The IL model uses a trained network consisting of demonstrations of human driving on the simulator. In other words, the IL model tries to mimic the actions of the expert behavior on which it was trained. RL uses a trained deep network with a defined reward function, in a particular environment based on the corresponding actions [6]. However, even these networks are vulnerable to adversarial attacks. Considerable related work has been done in the field of adversarial attacks, where a robust network is tricked by a simple perturbation. We use the imitation learning network from Carla [3] for the adversarial attacks.

DNNs are highly susceptible to pixel level perturbations in images. These perturbed images are easily misread by the network [7] [8]. Perturbed images that would be easily ignored by humans may not be correctly recognized by the DNN model. More recently, attacks in the physical domain have begun to draw more attention. An example being stop sign being misread as speed limit sign when the stop sign is physically tampered with [9].

3 Approach

The approach was to vary the parameters shadow intensity or mono-chromatic intensity and width of the projected line so as to maximize the infractions. We can quantify the adverse behavior using two metrics, namely steering angle difference and infraction. Steering angle difference is the change in the steering angle between the adverse and safe behavior instances. Infraction is the intersection of the car with the opposite lane. For the baseline scenario where there is no attack, the infraction is zero and it has a positive value when we introduce the attack. The steering angle difference may not be the best indicator of the behavior of the model as it might have a non zero value even without exhibiting adverse behavior which makes finding the threshold for adverse behavior slightly more complicated. Whereas infraction is strongly correlated with the model misreading the lines as lanes and gives meaningful understanding of when system is attacked. We tried both these metrics and have provided the optimization values for the steering angle difference.

To formalize this, a few notations are introduced. Let l denote the parameters of the lines, i.e color intensity and width of the line (c, w) on the track where the pattern is placed, and the pattern is denoted by δ. L denotes the set of feasible locations at which the adversarial pattern δ can be placed. Let x_l be the state of the track with parameters l, and $x_l + \delta$ then becomes the state of the track at this same position when the pattern δ is added to it. The state of the track lines inserted at a particular position is captured by the vehicle's vision system when it comes into view; we denote the frame at

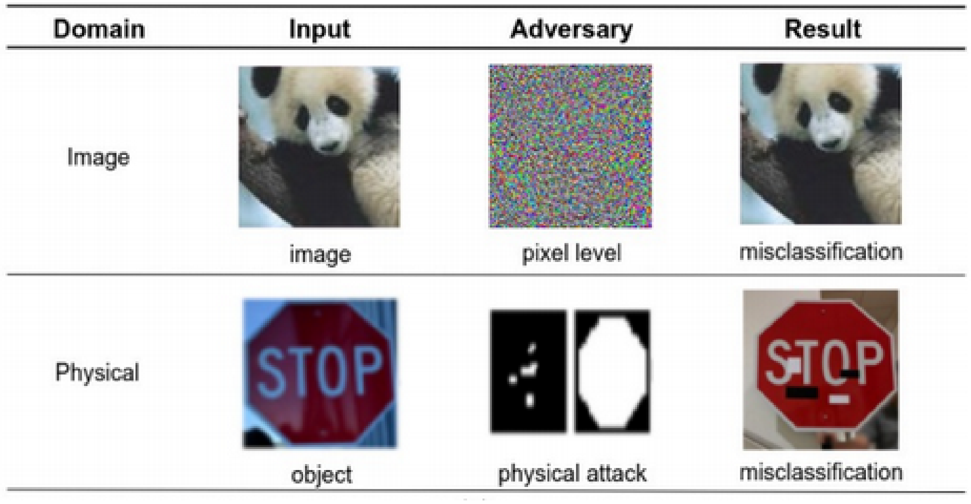


Figure 2: Existing attacks on machine learning models in the image.

which this location initially comes into view by f_l, and let Δ be the number of frames over which the track with parameters l is visible to the vehicle's vision system. Given the track state x_l with parameters l, the digital view of it in frame f is denoted by $y_f(x_l)$. Finally, we let $f_{sa}(y_f, h_f)$ denote the predicted steering angle and $f_{infraction}(y_f, h_f)$ denote the predicted infraction, given the observed digital image corresponding to frame f, and prior history of frames, h_f. Thus, our optimization equation can be given as:

$$\textbf{Adverse:} \max_{l,\delta} \sum_{\tau=0}^{\delta} f_{infraction}\left(y_{f_l+\tau}\left(x_l+\delta\right), h_{f_l+\tau}\right)$$

We wanted to experiment with different states of the imposed line and check whether the model exhibits maximum adverse behavior when the line width is close to the lane marking width. It was also important to check the color intensities at which we observe adverse behavior to get a sense of possible real world scenarios that might replicate this behavior, for example the shadows of various objects such as trees or poles on the road.

Once the behavior of the model to these modifications to the environment is observed, it's important to estimate the deviation from safe behavior and if we can expect to see adverse behavior from the model for a given state of the lines.[1]

4 Implementation

4.1 Autonomous Vehicle Simulator

Simulators have been used to test autonomous vehicles for the sake of efficiency and safety [10][11]. We ran our experiments on the CARLA [3] autonomous vehicle simulator. Built using Unreal Engine, CARLA has sufficient flexibility to create reasonably realistic simulated environments, with a robust physics engine, lifelike lighting, 3D objects including roads, buildings, traffic signs, and non-player characters including pedestrians and other vehicles

Fig 3 shows the third person view in the simulator. It allows us to acquire sensor data like the camera image for each frame (camera view), vehicle measurements (speed, steering angle and brake) and other environmental metrics that define the interaction of the vehicle with the environment in the form of infractions and collision intensity. Steering angle, throttle and brake parameters are the primary control parameters for driving the vehicle in the simulation.

[1] `https://github.com/mithun-bharadwaj/Adversarial_Attack_Self_Driving`

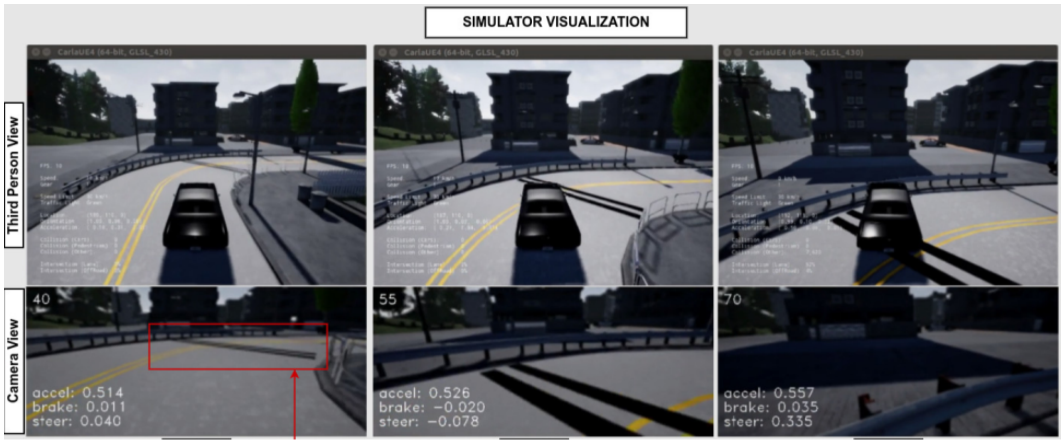


Figure 3: Visualization of the simulation in the third and first person view.

Table 1: Maximizing the adverse behavior of the model

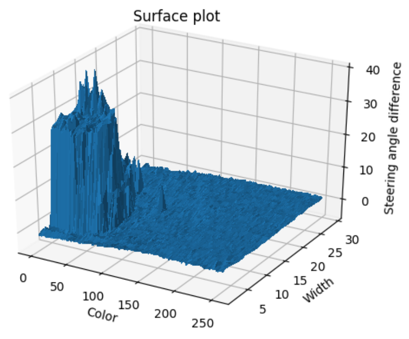
Iteration	Target (Steering angle difference)	Color Intensity	Width
1	-2.584	95.51	14.36
2	-0.8037	186.7	9.783
3	-3.592	39.78	4.028
4	24.42	14.81	13.26
5	-1.445	153.3	11.2
6	26.23	14.74	13.25
7	25.03	14.7	13.24
8	28.12	14.8	13.19
9	24.71	14.78	13.16
10	26.98	14.81	13.1

4.2 Maximizing infraction

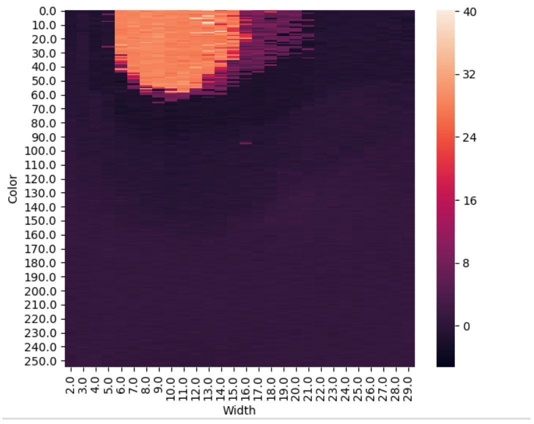
We run a baseline scenario in CARLA which allows the model to navigate around a bend without any changes to the environment We then collect some metrics such as sum of steering angles over the episode and the percentage intersection of the vehicle with the other lane (infraction) by querying the CARLA environment. Then we modify the environment with the lines and run the episode and collect the same metrics. The intuition is that when the deviation of the above mentioned metrics is high after the environment has been modified, the model is exhibiting adverse behavior. Bayesian Optimization was used for this. Table 1 shows the results of these episodes. We set the exploration stage to 5 epochs and exploitation stage to 10 epochs. The first 5 rows of the table show the random exploration phase.

4.3 Estimation of adverse behavior

The next stage was to estimate the amount of deviation from the safe behavior or baseline scenario. The data was collected by running the scenario for color intensities in the range [0, 255) in intervals of 1 and width in the range [2, 30) in intervals of 1. Our training data composed of 7140 samples. We built a four layered fully connected neural network for regression. Initial training was for 500 epochs and based on the training and validation curve we implemented early stopping at 250 epochs. A big challenge with regression was the type of data we had to fit a model to. Most of the color and width parameter values result in a value of 0 infraction and hence there is a sharp rise in the curve. Figure 4 and Figure 5 show the visualization of the data.

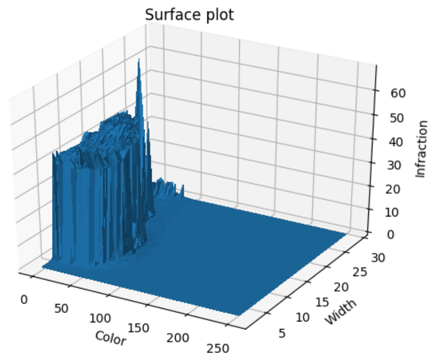


(a) Surface plot

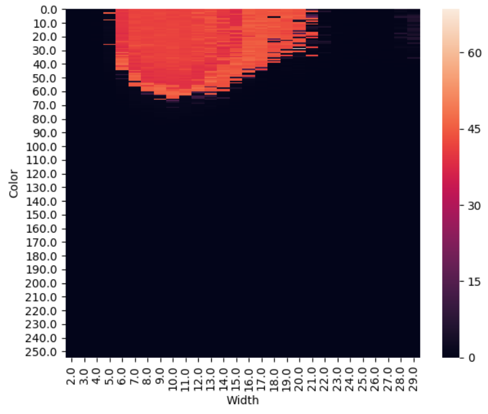


(b) Heat map

Figure 4: Variation of steering angle difference with width and color



(a) Surface plot



(b) Heat map

Figure 5: Variation of infraction with width and color

4.4 Classification

In all the scenarios that we simulated, we labelled the instances where adverse behavior was observed as class 1 and instances with safe behavior as class 0. An SVM with a polynomial kernel of degree 4 was implemented to classify the behavior of the system. With the combination of the regression and classification modules we had a sense of whether the modified environment will induce unexpected response form the system and if so, how much deviation can be expected.

5 Experiment and results

The initial training and validation results for the neural regression are shown in Figure 6. From the figure, it is clear that the the drop in the mean square error loss function is sharp for the first 100 epochs and reaches the average lowest error at around 200-250 epochs. As mentioned before, majority of the data collected consisted of zeros. When the data is randomly split into training, test and validation sets, we can expect that the validation set is completely filled with zeros if the split ratio is small. It becomes hard to draw meaningful inferences from training on such sets. We had to significantly increase the proportion of the validation set size to around 30% of the training set. We also force uniform sampling of the data in the range of input. This might be more difficult to enforce in most of the other applications where the true range of the input is not known.

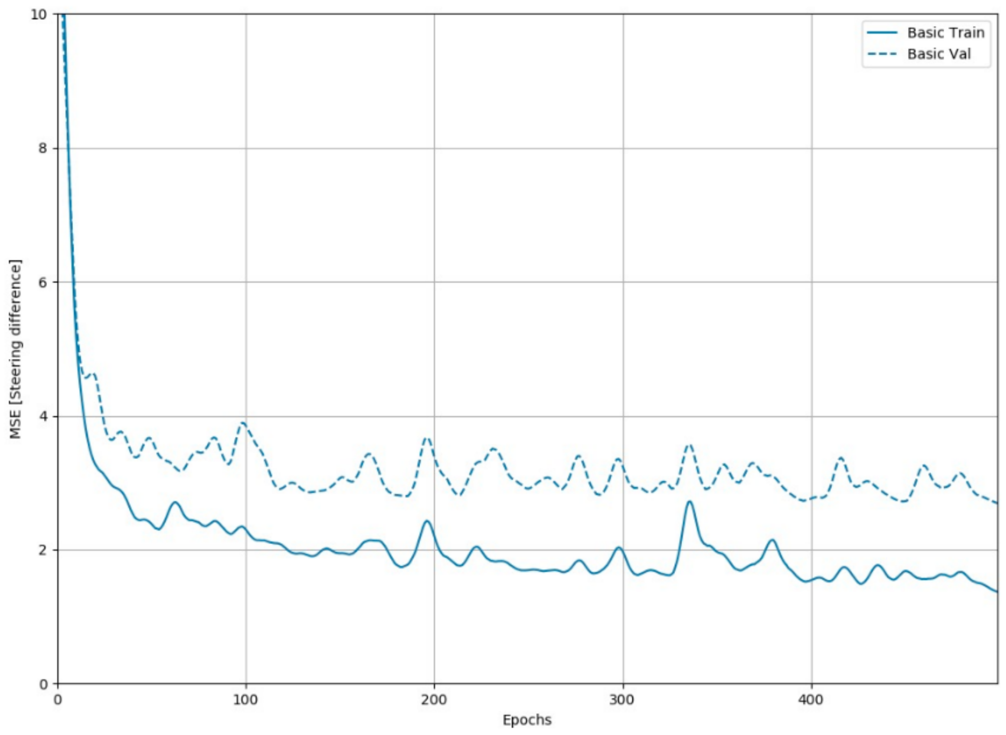
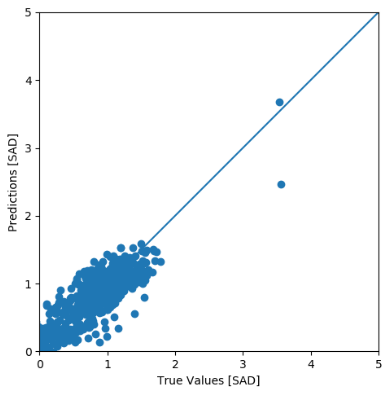
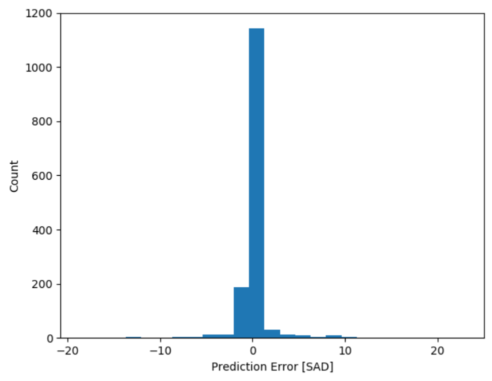


Figure 6: Training and validation curve for infraction estimation



(a) True value vs predicted value



(b) Test error histogram

Figure 7: Test results for infraction estimation

The test results can be seen in Figure 7. Figure 7a shows a plot of true value vs predicted value for the steering angle difference. The closer the point is to the $x = y$ line, the more accurate the prediction. The general distribution of the test and predicted data is along the $x = y$ line with a few outliers. Figure 7b shows a histogram of the test set error. Approximately 95.7% of our test data is within ± 0.5 error range.

As mentioned above we created a dataset for classification of infraction where the instance belongs to class 1 if it displays adverse behavior and class 0 otherwise. The results can be seen in table 2. We place special emphasis on the recall of class 1 and precision of class 0 as the evaluation metric. Recall of class 1 tells us the fraction of instances of adverse behavior that were recognized and we want this to be high. Precision of class 0 indicates the accuracy of the instances labelled as class 0. If this is low, it means a large number of instances belonging to adverse behavior have been misclassified as safe behavior, which is more dangerous than the misclassification of safe behavior as adverse.

Table 2: Infraction classification results

Class	Precision	Recall	F1-score	Support
0	0.99	0.93	0.96	1061
1	0.82	0.96	0.89	367

6 Future work

Our current work is limited to finding the perturbation in colour intensity and width such that the network is fooled to perceive the shadow as lane. The next phase would be to estimate the deviation, i.e the work of the regression and classification using images as input since that's the input to the e2e self driving model. Having this information will help make the model more aware about expecting reckless behavior. Incorporating this information in the model and retraining it will be a good way to try to defend against these attacks. We are also interested in checking the vulnerability of other end to end neural networks and compare the attacks against CARLA Imitation learning and reinforcement learning models. Knowing the vulnerabilities of these models is crucial since it's hard to logically deduce failure cases as deep learning models are still a black box at this point in time. Our future infrastructure will be built keeping these autonomous driving models in mind and it's important to make them extremely robust.

7 Conclusion

The contributions of this project are to develop a framework and simulation infrastructure to study adversarial attacks on e2e autonomous driving models. This framework can be used by other autonomous driving model to assess their robustness. We also successfully demonstrated how a simple adversarial attack is realizable such as projecting shadow-like mono-chromatic pattern which can be observed in a real world scenario, can negatively affect a supposedly robust model. We implemented a regression network to estimate the amount of deviation from the ideal scenario that can be expected for a particular modification of the environment and a classifier to detect irrational behavior.

References

[1] A. Fawzi, S.-M. Moosavi-Dezfooli, F. Codevilla, and P. Frossard. The robustness of deep networks: A geometrical perspective. *IEEE Signal Processing Magazine*, page 50–62, 2017.

[2] B. Biggio and F. Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *IEEE Signal Processing Magazine*, 2017.

[3] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy. Carla: An open urban driving simulator. *ACM Conference on Computers and Communications Security*, pages 1–9, 2018.

[4] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *CoRR*, 2016.

[5] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *NIPS*, pages 305—313, 1989.

[6] Aditya Boloor and Xin He. Simple physical adversarial attacks against end-to-end autonomous driving models. *International Conference on Embedded Software and Systems*, 2019.

[7] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. *CoRR*, 2015.

[8] Xiaoyong Yuan, Pan He, Qile Zhu, Rajendra Rana Bhat, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *CoRR*, 2017.

[9] Chawin Sitawarin and Arjun Bhagoji. Darts: Deceiving autonomous cars with toxic signs. *ACM Conference on Computers and Communications Security*, 2018.

[10] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. *CoRR*, 2017.

[11] Y. Tian, K. Pei, S. Jana, and B. Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. *IEEE/ACM 40th International Conference on Software Engineering*, pages 303–314, 2018.