SenFI: Predicting the Failure Rates of Autonomous Vehicles due to Sensor Faults

EECE 571K: Security and Reliability in IoT - Final report

Aarti Kashyap
Student number: 19115724
Electrical and Computer Engineering, UBC
akashyap18@ece.ubc.ca

Abdul Rehman Anwer
Student number: 50342690
Electrical and Computer Engineering, UBC
abanwer@ece.ubc.ca

***Abstract*—Autonomous vehicles (AV) have seen an immense increase in popularity recently and are expected to become fully operational on normal roads in near future. The safety critical nature of autonomous driving systems (ADS) of AVs makes quantifying the resilience properties of such systems highly important. This helps in ensuring public and governmental support for their adoption. Resilience of ADS in general has been explored before but effect on reliability of ADS due to specific components such as sensors present in ADS still remains to be explored. We propose using a systematic approach to find vulnerabilities in ADS introduced due to sensors and compare its performance overhead and coverage with fuzz testing techniques.**

I. Introduction

Fully automated self driving vehicles are poised to take the roads in near future. Partial self driving systems have already been deployed and tested for over a billion miles [7], providing valuable sensor data that is used to improve the performance of automated driving systems. The adoption of autonomous driving systems (ADS) on a large scale in real world is highly dependent on their reliability. Moreover, for the acceptance of deployment of such safety critical systems by wider public, makes it necessary to understand the resilience profiles of autonomous driving systems, identify and improve weaknesses in current designs and exhaustively test these fixes to increase public confidence.

Autonomous Driving System are composed of multiple sub-systems, the major parts being the sensor arrays and the control systems. A wide range of sensors like RGB cameras, radars, LIDARs, sonar, GPS etc. are used to provide information about the environment to the control system of ADS, which usually uses a Deep Neural Network to find the best course of action and gives commands to actuators, to control the movement of the vehicle. Such a highly integrated systems introduce various concerns about the overall reliability of the system. Moreover as ADS are highly safety critical, regulations and standards impose stringent reliability requirements on their components. For example the SDC FIT rate (Failure-in-Time rate) for of SoC used for DNN inferencing specified by ISO-26262 is 10 i.e. ten failures in one billion hours of operation [10]. These strict reliability requirements, demand that a reliability analysis be performed on integrated systems and vulnerabilities or faults that can propagate, from one sub-system to other sub-systems be identified and mitigation techniques applied to make the system resilient to such faults.

Studying the autonomous driving system is hindered by resource intensive nature of training and testing solutions in real world scenarios. Dosovitskiy et al. [5] recently introduced CARLA, which is an open source simulator for training and validating autonomous driving platforms. It provides an urban environment setting from which data is captured using a flexible array of sensors and signals that are normally present in ADS, this simulated data can then be used to train and test different sub-systems of an ADS. As CARLA also provides sensor reading in addition to images captured by camera, it can be used to study the effect of a vulnerability present in a single component (for example the LIDAR sensor) on overall reliability of the whole ADS. CARLA operates in a server-client model. Server simulates the environment and generates readings for different sensors which are then provided to the client which acts as a driving agent (DA) taking different control decisions based on the provided sensor inputs. Separation of server and client functionality permits implementation of a wide range of DAs.

As per our knowledge, we are the first to attempt to find the failures caused by sensors in a self-driving car.

Section 2 introduces the related work done in the resilience checking of self-driving cars and how our work is different from the existing literature. Section 3 introduces the background related to self-driving cars. This section further talks about the different sensors which form the core components of the self-driving cars and then explains the in-depth working of LIDAR, since finding failures due to LIDAR is the core contribution of our work. Section 4 walks through the fault model, type of faults and the metrics we use in our study to evaluate autonomous driving systems. Section 5 briefly describes configuration of CARLA and driving agent we use in our study and describes the method of Fault injection we use in our study and the limitations of our chosen methods and environment.

II. Related Work

Reliability and safety of individual components in ADS have been previously studied. Li et al. [10] studied the impact of soft errors on DNN using Fault injections on hardware accelerators and proposed solutions for making DNNs more resilient. Using Fault injections for testing reliability of system has inherent disadvantage that for complex systems like DNNs a huge number of FI are required to get statistically meaningful insights, moreover the coverage that FI provides is also dependent on the input used for FI, so in case of DNN where the input has a diverse range, performing FI to get meaningful resilience profile of a DNN sub-system of an ADS can be quite resource intensive, otherwise some corner case may be missed during the analysis. Keeping these things in mind Pi et al. [12] introduced DeepXplore, which tests deep learning systems in a systematic way by finding the inputs for DL systems, that will result in mismatch in output for different DL systems that provide same functionality. In this way corner cases that are not captured by regular testing and training cycle in neural networks are found thus exposing erroneous behavior of the system without a significant overhead of finding corner cases manually. DeepXplore also finds input that maximize the fraction of neurons activated when tested, thus increasing the tests coverage of DNNs. DeepXplore can provide a good defense against vulnerabilities and erroneous behaviors in DNN part of ADS but for overall reliability of the program other components of the system also need to be analyzed.

Dosovitskiy et al. [8] developed AVFI which is a fault injector that performs resilience assessment of complete autonomous vehicle(AV) systems. AVFI inject faults in without any systematic approach in AV system which consists of CARLA simulator and a driving agent which performs the control actions. AVFI can inject hardware faults (modeling soft errors), data faults (modeling error in sensor values), timing faults and machine learning faults (errors leading to wrong prediction). Jha et al. [9] developed Kayotee which is a FI based tool to inject faults in an ADS (similar to AVFI) and then classifying errors and safety violations impacting the reliability of autonomous vehicles. Kayotee injects faults in both hardware and software components of proprietary Nvidia DriveWorks platform, in a closed looped environment and then error propagation and masking characteristics of the ADS compute stack are evaluated using a predefined ontology model. Kayotee and AVFI provides a holistic view of the system but its fault coverage is also limited, as only those errors can be studied which have been simulated using FI. As whole system is taken into consideration during holistic analysis, trying to increase the test coverage using FI can lead to state space explosion.

//TODO

Distinguish ourselves from Kayptee and AVFI.

III. Background

A. Autonomous Cars

Self-driving autonomous cars use various autonomous technologies to provide an effortless mode of transportation. In order for autonomous cars to work effectively a proper synchronization of advanced sensors collecting data from surrounding environments and advanced algorithms processing data and the vehicle in real-time.

The main components of a self-driving car are sensors (we will talk about them in more details in the next subsection), computer processors, batteries, back-up systems (eg. steering, braking etc.), rounded-shapes which maximizes the sensors coverage area and the interiors(designed for riding, not for driving).

B. Sensors

Sensors are the core components in a self-driving car. We will talk about them in a little more detail, since our research goal is to find error rate in self-driving cars caused by sensor faults.

The core sensors of a self-driving car are as follows:

1) Radar sensors: Radar sensors help the autonomous vehicles in determining the road dynamics such as detours, collisions, navigation by sending signals to the on-board processor. The on-board processor on receiving the signals from the radar sensors takes decisions such as applying brakes and/or move out of the way. The decisions made by the processor or the brain of the vehicle don't depend entirely on the radar sensors. The processor decision also depends on the gyroscopes and wheel encoders to send accurate and on time signals to it. The radar sensors are usually mounted on the bumper (two in the front and two in the back).

2) Optics: High powered cameras in conjunction with Radar sensors are another vital component of the autonomous vehicles. The camera setup on each car differs according to the manufacturer. The main goal of the camera placement is to get the surrounding views covered properly, so that the processor can take correct decisions. Some manufacturers use a single camera on the windshield [13] whereas some vehicles require multiple cameras placed equidistant from each other to get overlapped views which are sent to the processor. The processor uses the views to calculate the depth of the field and/or perform semantic segmentation.

3) GPS: Global Positioning Sensors is very similar to the navigation maps we use in our mobile phones. However, GPS in autonomous vehicles is very important since we enter the destination point and the start point of our trip. The GPS works in conjunction with LIDAR, Radar Sensors and Deep learning software [1] in order to guide the vehicle in the correct direction to reach the destination.

4) LIDAR: Laser Illuminating Detection and Ranging is the sensor on which we are focusing on in this work. LIDAR unit resembles a spinning siren light and ranges up-to 100 meters. The goal of LIDAR is to provide driver-less cars with

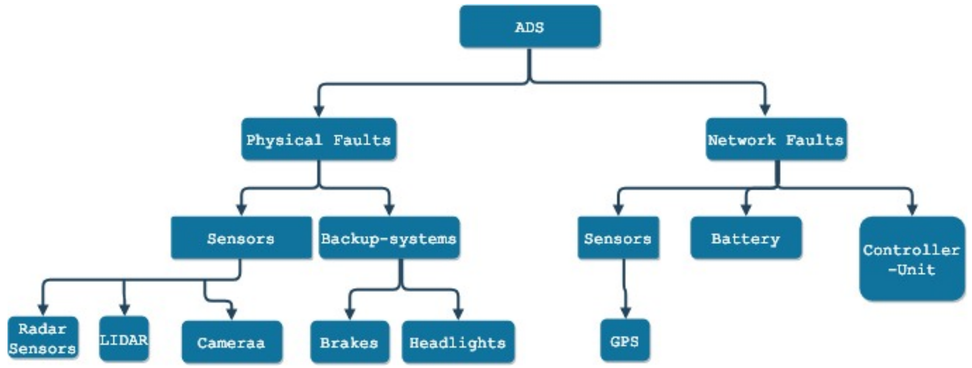


Fig. 1: This shows a general fault model of Autonomous Driving Systems. We have kept it generic enough to be considered for different Autonomous systems. However, our work is oriented towards autonomous cars.

accurate long range detection. LIDAR continuously performs 360 degree rotation in order to scan the world around. The sensor generates raw information about the world and the information is sent to the processor which digs out the relevant information in order to guide the vehicle and take decisions.

C. Camera

The two most expensive sensors in an ADS are the LIDAR and the Camera. The reason we chose Camera for our experimentation is because of the availability of test-beds in the form of CARLA. The fault-model for camera has been described in

IV. Approach

Our contribution is two-fold in this work. First, by an in depth understanding of autonomous vehicles, we device a fault model for the vehicle in general. We specifically focus on the sensor faults and design a fault-model specific to the sensors. Second, based on our fault model, we inject faults in order to obtain the failure rate of the system caused due to the sensor faults. We attempt fault injection in order to get maximum coverage of the system in an optimal manner.

A. Fault-model for ADS

Analyzing the system, we designed a fault model for autonomous driving systems. We categorized the fault occurrence in two ways: due to some physical damage or due to network issues. These two categories of faults cover almost all possible components of the systems which can lead to a failure of the system.

In Figure 1 we provide a fault-model for an ADS. However, covering the failures caused by each component is out of scope of our current paper. Also, analysing failure rates in general instead of component wise failure rates does not provide us with useful set of results. Hence we introduce a fault-model for the sensors of an ADS.

The core component we selected were the sensors used in ADS. The sensors are one of the most expensive parts of an ADS. Hence, determining the failure rates due to these sensors can help us in minimizing damage. We described different categories of sensors in Section-3. We start with an analysis of one of the sensors described above.

B. Fault-model of Camera

Camera is a very expensive sensor and hence finding the failure rates of the system, caused due to camera can help the designers understand how to mitigate the faults such that they don't lead to failures. ADS being safety critical systems, can cause a lot of harm (death in the worst case scenario) in case of a failure.

1) Hardware faults: These faults represent the soft errors that can occur in the data, that is recorded by the sensors. These faults can occur in the registers of sensor units or in communication links between sensor and controller. These faults can also occur within registers, where these values are stored in the main ADS controller. In our study we assume that the fault is always activated regardless of where the fault actually occurred. This manifests in the form of sensor data being altered before it is used by controller to make decisions. These type of faults in real life can introduce single bit-flips which is modeled by our fault model.

We created two categories of faults under this section. The first is due to the alpha-rays which causes random bit flips. The other set of faults which can occur is due to wear and tear of the sensor. Sometimes, buying cheaper sensors can lead to a faster wearing away of sensors.

2) Data Faults: These faults refers to the conditions in which the sensors give readings which are not correct representation of the environment. These faults can occur due to manufacturing defects in the sensors and various environmental conditions. Environmental conditions can refer to weather and other characteristics of the environments that are used in decision making process by the ADS.. For example, A camera mapping its surroundings can be effected by rainy weather or intense monochromatic lights, which may lead to addition of noise in the image captured by the camera sensor.

C. CARLA

CARLA is an open source simulator which simulates a dynamic world and provides a simple interface between

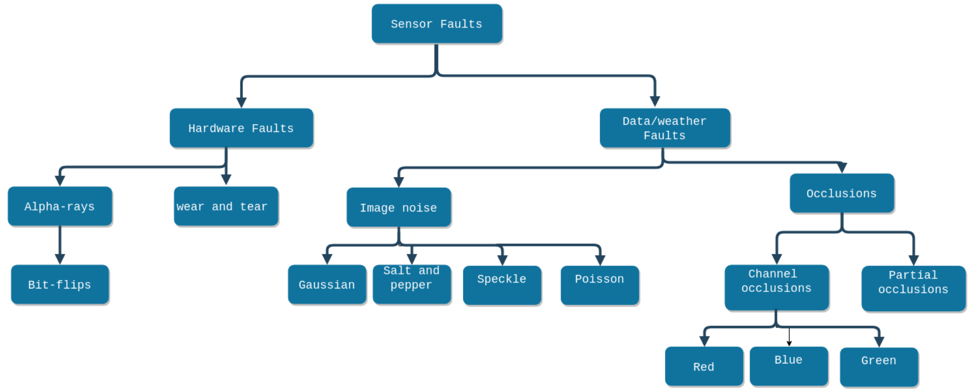


Fig. 2

environment and autonomous vehicles (AV). The simulation engine simulates a vehicle in a city environment with realistic physics. Measurements representing the environment are provided to a driving agent (DA) which controls the actions that the AV takes based on these measurements and some predefined algorithm. Figure 3 shows the high level architecture of CARLA. CARLA operates in a server-client model in which server handles the simulations, while client controls the actions of the AV(through DA) as well as simulation parameters of the overall system.

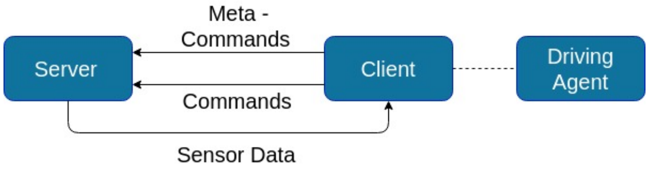


Fig. 3: High level architecture of CARLA, Server sends sensor data to the client which integrates with driving agent and returns driving commands to server.

Although CARLA provides ability to simulate complex environments, Dosovitskiy et al. [5] have shown that for a range of driving agents, only simple paths with no dynamic actors produce successful results for more than ninety percent of trials.

1) Driving Agent: As mentioned in the previous section the behavior of the vehicle in the simulation is controlled by the driving agent. Our goal is to study the resilience of ADS in CARLA, so we have chosen a driving agent developed by Codevilla et al. [4], which uses conditional imitation learning to train a recurrent deep neural network. This network is used in simulation for selecting appropriate action based on sensor. Dosovitskiy et al. [5] tested multiple driving agents and have shown that success rate of imitation learning based driving agent is highest(90%). Table I shows the results of this study for our testing conditions.

2) Environment Setup: In our study we focus on the sensor faults, so to avoid skewing of results due to any machine learning error we select a subset of paths for fault injections, in which the ADS encounters no machine learning errors and is always able to complete the predefined goal without any traffic violations in all weather conditions.

Drving agent	Training env.	Testing env.	Average
Multi-layer perceptron	82	95	88.5
Imitation Learning	89	90	89.5
Reinforcement Learning	34	16	25

TABLE I: Percentage of successful episode for different driving agents [5].

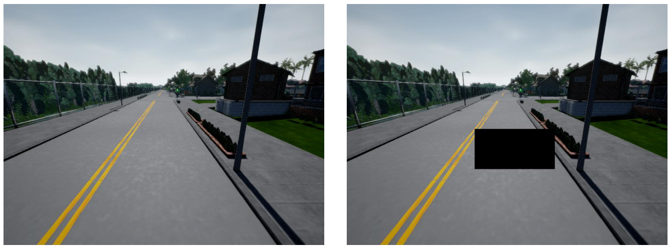
- **Path:** Length of the paths selected for our experiments ranges from 70 to 100 meters and contains one turn between the starting and the ending position.
- **Environment:** No dynamic objects (i.e. NPC vehicles, pedestrians etc.) are present in the simulation environment and six predefined weather conditions provided by CARLA are used for each path.

D. Fault Injection

In our study with random Fault injection we study each fault separately, so for a set of related experiments we only inject one type of fault. A single fault is injected, according to the fault model, in each execution run and its effect on the overall system is observed. The objective and path to navigate is constant in each trial so that its outcome can be compared with output of a golden run, in which no fault was injected.

The output of optical sensor in attached to ADS used for experiments, is a three dimensional array of size 600 x 800 x 3, which represents an RGB image of size 600 x 800. The value for each pixel ranges from 0 to 255 for all channels. The fault in the camera sensor can be simulated by altering the captured image. For example, partial occlusion fault in the experiments is simulated by selecting a random portion of the image of size 100 x 200 and setting the value of all the pixels in this portion to 0. Figure 4 shows the effect of partial occlusion on the sensor data. Details about effect of other faults are mentioned in Appendix A.

As mentioned in IV-C CARLA provides a client server architecture. We exploit this boundary between client and



(a) Sensor output (b) Sensor output after FI

Fig. 4: Effect of partial occlusion fault

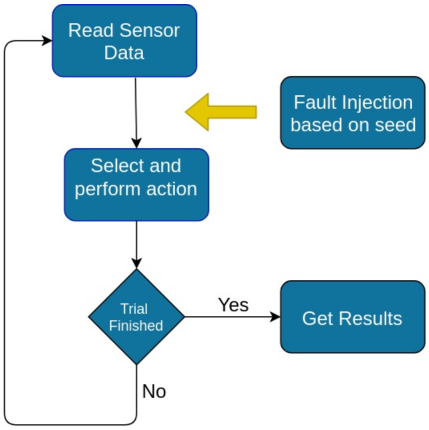


Fig. 5: Life cycle of an experiment trial on client side, Fault is injected by altering sensor readings received by client.

server for our fault injection experiments. Figure 5 provides an overview of FI method used in one trial. CARLA platform provides an ability to run multiple trials of the same experiments (same environment and goals). Each trial involves multiple steps. In each step, data is read from the sensors, which is provided to the driving agent, which based on some predefined algorithm, gives command to the AV, after the command, it is checked if the objective of the trial is achieved or a timeout has happened. If any of these condition exists, we end the current trial and the using the measurements from the server, the metrics defined in IV-D are calculated for this trial.

The sensor readings are provided in form of dictionary to driving agent from the client, which can be easily accessed in the code. We insert a code block in between reading data from server and sending it to driving agent. This code block changes the reading of the sensor under study, based on the fault under consideration.

We use the following metrics to quantify the resilience of ADS which are inline with previous work in this domain [8].

1) Success percentage: This metric represents the fraction of total trials in which the ADS was successful in completing its objective i.e. navigate successfully from starting position to end point with in specified time.

2) Success percentage without violations: This metric captures the traffic violations that occur during each trial. This is a more stricter criteria which records trials with no violations. The violations that we consider for calculating this metric are lane violation, curb violations and traffic law violations (reasons for not considering collisions with dynamic objects are explained in IV-C).

The two metrics used, provide and opportunity to study the effects of faults on the overall functioning of AV as well as the lower level deviations from normal behavior, which may not lead to outright failure but violate safety properties.

V. Evaluation

In this section we present the fault model for the sensors. We further evaluate the failure rates caused due to the faults in the sensor. We address the following research questions:

1) **RQ1** What all types of faults can exist in advanced and core sensors of Autonomous vehicles such as the optical sensor.?
2) **RQ2** What faults do we consider for the autonomous driving systems?
3) **RQ3** What subset of the above faults cause the most amount of system damage i.e system failure?
4) **RQ4** What are the overheads associated with Fault Injection in CARLA?

We answer the above research questions.

RQ1: What all types of faults exist in advance and core sensors in an autonomous vehicle.

There have been lot of studies which look at the individual sensors. These studies also performed an in-depth analysis of the faults which can be present in sensors such as high-powered cameras [3], [2]. However, none of the previous work categorizes the fault model into hardware faults and data faults as shown in fig. 2. We not only constructed a fault model for the high-powered image sensors, but we chose the faults which are most likely to occur in case of a self driving car. The previous work by Bresolin et al. [2] performs a fault diagnosis of hybrid systems. However their fault model is limited, as they only consider the hardware faults.

We chose to start our work with optical sensors because of many reasons. Due to the unstable nature of LIDARs a lot of industrial focus is being shifted towards cameras. A recent work says that soon cheap cameras would replace LIDARs [11]. Hence visualization of a 3D world can be constructed using a pair of cameras. In order to do so, it becomes an absolute necessity to understand the faults which can be a part of high-powered cameras as well as low powered cameras.

As shown in Fig. 2 we consider hardware faults and the data faults. The two major categories we consider in our system for data faults are the image noise and the occlusions. Image noise is the random variation of brightness or color information in images. The effects of image noises are diverse and can vary from causing minimal damage to the image to a level which makes it difficult to determine the subject in the image. The different types of image noises can occur due to heating up of the camera and the internal circuits. The image noise can also occur due to temperature variations in the environment. Though there are many existing filters to mitigate these noises, our work is to understand which types of noises despite the numerous filters present cause failure of the system.

The second types of data faults which we explore are the occlusions. Occlusion means that there is something we want to see, but can't due to some property of our sensor setup or some event. The way of dealing with occlusions varies depending on the way it was caused. If we are developing a system which tracks objects (people, cars, ...) then occlusion occurs if an object you are tracking is hidden (occluded) by another object. Like two persons walking past each other, or a car that drives under a bridge. The problem in this case is what we do when an object disappears and reappears again.

RQ2: What faults do we consider for the autonomous driving systems and why?

In order to validate our results and understand we limit our experimentation model. Based on our judgment and understanding, we find experimenting with data faults more compelling as compared to hardware faults. Our reason are well-justified and we will validate them in the next set of RQs.

A. Limitations of CARLA

The degree to which CARLA models the real world autonomous driving systems and their interaction with the environment, is dependent on the simulation parameters and driving agents used for controlling the ADS. In real world autonomous driving systems an array of sensors is used, which captures a representation of the environment. This captured data is used by the controller to make driving decisions. CARLA provides the capability to simulate important sensors like cameras and LIDARs but some important sensors like Radar can't be simulated. Moreover the developed driving agents do not incorporate all the sensors available in CARLA. For example, the three driving agents studied by Dosovitskiy et al. [5] do not use LIDAR and rely only on camera output to take driving decisions. Using only camera sensors is not a realistic modeling of real world driving systems.

There are no open source driving agents currently available that use sensors other than camera so in our study the driving agent we chose also uses only camera sensors. As our focus is on faults in camera sensors, this provides us good insight into reliability of ADS due to these faults. Presence of other sensor in the system may lead to masking of these due to other sensors, which is a more realistic representation of a real world system. Studying the effect sensor faults in presence of other sensors, is the next logical step after studying the reliability of ADS due to individual sensor faults. CARLA helped us simulate the optical sensors, in order to understand the failure rates due to image noises and occlusions. We look into different types of occlusions as shows in Fig 2. We test for different noises such as Gaussian noise, salt and pepper noise, speckle noise and Poisson noise. The different types of image noises represent different types of probabilistic faults in an image. For example Gaussian noise distributes the noise uniformly across the entire image. Every image noise has a different probabilistic model. Hence, according to Goodfellow et al. [6] we can see that sometimes introducing noise can fool the neural network by giving better results than before.

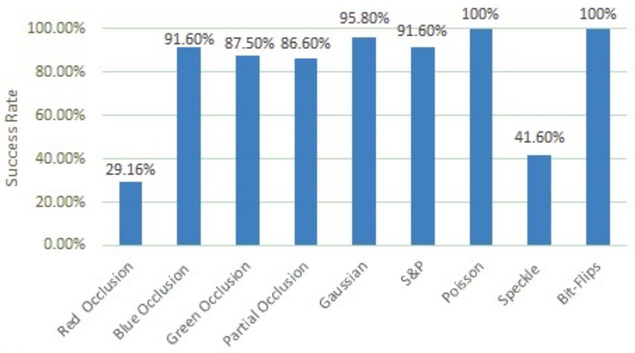


Fig. 6: Percentage of trials in which ADS was able to complete specified goals in presence of faults.

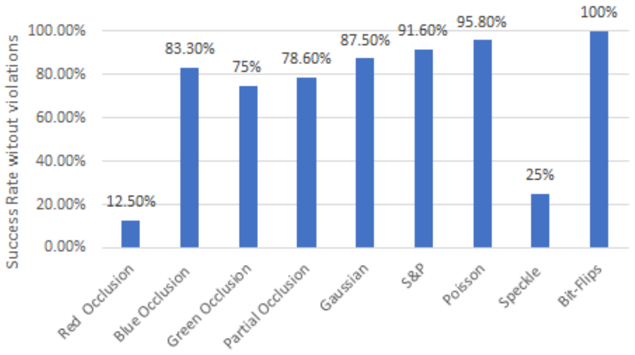


Fig. 7: Percentage of trials in which ADS was able to complete specified goals, in presence of faults, without committing any traffic violations.

We try to understand if all image noises indeed cause the failure of the system. Hence we focus our experimentation in understanding image noise and occlusions.

RQ3: What subset of the above faults cause the most amount of system damage i.e System failure?

Figure 6 shows the success rate of ADS after injection of faults. It can be seen that hardware faults, which are represented as single bit flips in our experiments, do not cause any failures. The output data of image sensor consists of approximately 11.5 million bits, so a single random bit flip event is highly unlikely to cause a perturbation that leads to system failure. Figure 7 shows the fraction of trials in which the ADS committed any traffic violation, this graph shows that there were no traffic violations when single bit flip faults were injected in the image sensor data. These results validate our

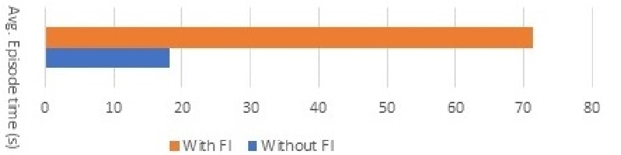


Fig. 8: Average Overhead of injection of data faults in seconds.

approach of focusing on data faults.

For channel occlusion faults, it can be seen that Red channel occlusion leads to highest number of failures as the success rate is only ~30%. Green and Blue channel occlusions only result in approximately 10% decrease in success rate. Similar trends can be seen from Figure 7 in which Red channel occlusions lead to significantly more traffic violations than other channel occlusions. This shows that the driving agent relies more on red channel then other channels in sensor data. The partial occlusion faults also cause a significant number of failures in the system, with approximately 15% of trials resulting in failures.

For image noises it can be observed from Figure 6 and Figure 7 different noises have varying effects on the success of system behavior. Speckle noise results in the highest number of failures and the success rate in case of speckle noise injection is only ~40%. On the other hand injection of Poison noise, results in no failures although Poisson noise causes some traffic violations but none of these violations lead to system failure.

RQ4: What are the overheads associated with Fault Injection in CARLA?

From previous RQ we know that channel occlusion faults and noise addition faults result in most failures of the system so from now on we focus on these faults. For finding the metrics mentioned in Figure 6 and Figure 7 we use exhaustive Fault injection. For injecting noise in the sensor data all the pixels in the image have to be updated in every iteration, similarly for channel occlusion faults all the pixels of a particular channel have to be updated in every iteration. The image produced by the camera consist of around 1.4 millions pixels. Updating such a high number values in every iteration of the episode has a huge overhead. Figure 8 shows the average time taken by a single episode of the experiments with and without data fault injections. It can be clearly seen that injecting data faults leads to almost 4 times more time spent in each episode. This overhead and thousands of FI experiments required for exhaustive FI makes this method of FI unscalable.

We analyzed the failures due to the data faults for any information that can lead to a technique which provides the same coverage as exhaustive FI with less overhead. An interesting fact that emerged from this study was that about 80% of the failures due to data faults were concentrated near such locations in the environment which were critical. These critical locations include intersections, traffic lights and traffic signs. Figure 9 shows a subset of these locations on the map that we use for our experiments. We also note that the pattern of parameters of commands, that the controller uses to control the action of the ADS, changes when these critical locations are detected. We combine these two observations to theorize that injecting fault on these locations can lead to all the failures that we observed at these locations during exhaustive fault injections.

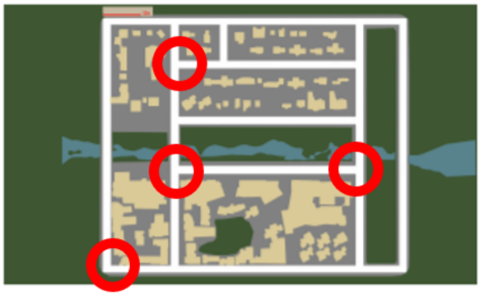


Fig. 9: Critical locations in our testing environment, at which most failures are observed.

Testing the proposed hypothesis reveals that a single data fault injected at critical location does not lead to system failures. We theorize that a single fault is not able to cause large enough deviation in the behavior of ADS to cause a system failure and the Neural network used in the driving agent is able to overcome this fault. With this observation in mind, we injected faults in iterations preceding these critical locations and observed that by increasing the number of iterations in which faults are injected the coverage of FI can be increased, but the drawback is that for the same level of coverage as exhaustive FI, almost same number faults have to be injected which defeats the purpose of the proposed technique.

VI. Discussion

In order to perform the first set of experimentation, we initially chose LIDAR. However, simulating a LIDAR sensor is difficult since, the number of states in LIDAR are not finite. It is continuously rotating and obtaining new results and hence changing states. Hence, we decided to change our initial sensor and chose a camera instead.

The first fault-injection methodology though provides a good coverage incurs a lot of overhead. In order to optimize the injection process, we inject faults after every 10 frames. However, we observe that this reduces the coverage though providing us with better overhead. Hence, this becomes a question of trade-off between overhead and coverage.

VII. Future work

We intend to setup our test environment in a distributed environment and observe the failure rates in that context. In that case, we don't look at one car individually but we have multiple cars who are supposed to work with each other.

VIII. Conclusion

References

[1] NVIDIA BLOG. Beyond gps: How hd maps will show self-driving cars the way. https://blogs.nvidia.com/blog/2016/04/05/self-driving-cars-2/.

[2] Davide Bresolin. Fault diagnosis of hybrid systems: An onboard camera model. volume 8, pages 714–719, 08 2012.

[3] L. M. Capisani, A. Ferrara, and P. Pisu. Sliding mode observers for vision-based fault detection, isolation and identification in robot manipulators. In *Proceedings of the 2010 American Control Conference*, pages 4540–4545, June 2010.

[4] Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *International Conference on Robotics and Automation (ICRA)*, 2018.

[5] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[6] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv e-prints*, December 2014.

[7] Tesla Inc. Vehicle safety report (Q4-2018). https://www.tesla.com/en_CA/VehicleSafetyReport.

[8] Saurabh Jha, Subho S. Banerjee, James Cyriac, Zbigniew T. Kalbarczyk, and Ravishankar Iyer. Avfi: Fault injection for autonomous vehicles. pages 55–56, 06 2018.

[9] Saurabh Jha, Timothy Tsai, Siva Hari, Michael Sullivan, Zbigniew Kalbarczyk, Stephen W. Keckler, and Ravishankar K. Iyer. Kayotee: A Fault Injection-based System to Assess the Safety and Reliability of Autonomous Vehicles to Faults and Errors. In *International Workshop on Automotive Reliability and Test (ART), 2018*. IEEE, 2018.

[10] Guanpeng Li, Siva Kumar Sastry Hari, Michael Sullivan, Timothy Tsai, Karthik Pattabiraman, Jeol Emer, and Stephen Keckler. Understanding Error Propagation in Deep Learning Neural Network (DNN) Accelerators and Applications. In *International Conference for High Performance Computing, Networking, Storage, and Analysis (SC), 2017*. ACM, 2017.

[11] Andrew Liszewski. Elon musk was right: Cheap cameras could replace lidar on self-driving cars, researchers find = "https://gizmodo.com/elon-musk-was-right-cheap-cameras-could-replace-lidar-1834266742".

[12] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. DeepXplore: Automated Whitebox Testing of Deep Learning Systems. In *Symposium on Operating Systems Principles (SOSP), 2017*, pages 1–18. ACM, 2017.

[13] MIT Technology Review. One camera is all this self-driving car needs. https://www.technologyreview.com/s/539841/one-camera-is-all-this-self-driving-car-needs/.

Appendix

A. Fault representation in CARLA

1) Bit Flips: A random pixel is selected from all the channels of sensor data and one bit from that pixel is flipped to zero or one depending on its previous state.

2) Partial Occlusion: A random 100 x 200 contiguous rectangle is chosen in the sensor data and value of pixels, for all the channels in this regions, are set to 0 thus emulating the occlusion of the view captured by the image sensor. The location of rectangle to be occluded within the image, is selected at the start of episode and the selected portion of the sensor data is blacked out throughout the episode.

3) Channel Occlusion: The value of all the pixels of the channel to be occluded, are set to 0 in the channel data thus emulating the condition where the capability of the sensor to differentiate between intensities of the channel is absent.

4) Gaussian Noise: A three dimensional array, with dimensions similar to the sensor output, is created which follows a normal distribution with mean of zero and variance of 0.1. This array is added to the original sensor data thus emulating presence of a Gaussian noise in sensor data.

5) Salt-and-pepper Noise: Random pixels are selected from all channels of the sensor output with a probability of 0.4%. The value of half of these selected pixels is set to 0 and the value of other half is set to the maximum pixel value (255) thus emulating per channel Salt-and-pepper noise.

6) Poisson Noise: Number of unique pixels in all channels of the sensor data along with sensor data is used as an expectation interval for the Poisson distribution. The resulting samples from this Poisson distributions are replaced int he original sensor data thus emulating Poisson noise.

7) Speckle Noise: A three dimensional array, with dimensions similar to the sensor output, is created which follows standard normal distribution. Each individual pixel of this matrix is multiplied with the corresponding pixel in sensor data and the result is added to sensor data, thus emulating the multiplicative nature of speckle noise.