UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Mike Gomes Camara

Safety Analysis of Autonomous Vehicle Systems Software

Master's Thesis (30 ECTS)

Supervisor(s): Dietmar Alfred Paul Kurt Pfahl, PhD

Tartu 2022

Safety Analysis of Autonomous Vehicle Systems Software

Abstract:

Machine learning approaches to autonomous driving systems that rely upon computer vision and deep neural networks have demonstrated encouraging results in the past. Some believe that the so-called end-to-end strategy is the only way to deploy self-driving vehicles at scale in the future. Safety is a concern as the efficacy of such neural networks is susceptible to adversarial attacks and are also highly dependent upon a RGB camera input.

Literature suggests that different attacks require equally different procedures to defend against such threats. However, there is no understanding of how adversarial defenses can improve the capacity of an end-to-end self-driving model to generalize to different lighting conditions.

This thesis project aims to understand how adversarial attacks can help a machine learning model to increase resilience and generalize better to different lighting conditions. First, we select a scaled real-world driving platform and a neural network to drive the model. Then, we designed an experiment, implemented, tested, and evaluated the results.

In conclusion, adversarial defenses did impact the capacity of a self-driving end-to-end model to generalize to different lighting conditions. Therefore, further research is suggested employing adversarial training to increase the robustness of autonomous driving models.

Keywords:
adversarial attacks, adversarial machine learning, autonomous vehicles, driverless cars, self driving cars, open source, CNN, deep learning, Raspberry Pi, behavioral cloning

Contents

1 Introduction

Adversarial machine learning attacks are minor disturbances injected into the input of a classification model, including autonomous vehicles models. Expected consequences of adversarial attacks used in self-driving cars include an increased rate of collisions. Previous research has shown that models trained with adversarial defense methods are more likely to generalize to unseen conditions and the attacks themselves. This study examines the efficacy of adversarial training methods to improve the performance of end-to-end self-driving models were tested in four conditions.

End-to-end driving is an approach to autonomous driving that has become a growing trend in autonomous vehicle research both in industry and academia[1]. Unlike modular methods that use expensive sensors, end-to-end techniques to autonomous driving rely on computer vision and machine learning to generate models that command steering and throttle [2]. However, these models are notoriously susceptible to lighting level conditions and vulnerable to adversarial images[3], which are often imperceptible to the human eye but can cause the model to misbehave. Testing self-driving models outside the simulation environment become more pivotal as fleets of autonomous vehicles that use machine learning are ubiquitous and available to the general public.

Creating strategies to defend end-to-end models and add robustness and resilience against changes in environmental conditions is paramount. Such vulnerabilities must be addressed and mitigated before we can see wider adoption of machine learning models fully controlling autonomous cars [4].

In the context of end-to-end autonomous driving, model misbehavior can lead to catastrophic consequences such as loss of life and property [?]. Research has been done on the impact of malicious attacks in autonomous-vehicle neural networks, mainly in simulation environments[5]. This research incorporates adversarial training methods to improve the model's skills in generalizing to unseen illuminance levels and adversarial attacks in a scaled real-world setting.

Thus, the main research questions of this study are as follows:

- **RQ1** - Can adversarial training methods improve the model's generalization skills to unseen lighting conditions?
- **RQ2** - What are the benefits of using adversarial training methods to train machine learning self-driving models?

The hypothesis is that models that have been trained using adversarial defense methods have better performance on unseen lighting conditions than the models trained with standard methods. Additionally, increasing the training dataset with adversarial images will increase the model's ability to generalize to unseen lighting conditions.

To answer the research questions, a set of experiments will be conducted to evaluate the effectiveness of an adversarial defense training method in improving model skills in generalizing to unseen lighting conditions in a real-world setting using a scaled autonomous car.

1.1 Motivation

Self-driving car manufacturers such as Tesla deploy autonomous driving solutions trained with deep neural networks at scale [1]. Nowadays, Tesla owners can experience rides having their car driving autonomously while still paying attention to the roads. However, in 2016, a Tesla Model S. with the Autopilot feature engaged failed to apply the brake after not noticing the white side of a tractor-trailer against a brightly lit sky, tragically killing its driver [2].

Autonomous driving technology is engineered to improve safety, and Tesla cars are among the safest in the world [3], the manufacturer claims their dataset is used to train their end-to-end driving models that control the automated driver assistance relies upon is more extensive than any other car manufacturer.

End-to-end methods are a deep neural network approach to self-driving and a growing trend in autonomous driving research because such methods are cheaper, simpler, and scalable, unlike conventional modular approaches [4]. However, such an approach requires collecting large amounts of data to train the models, with many varied examples of situations, which is impossible since one cannot control factors such as time, availability of resources, or even the weather. For instance, minimal changes in the light exposure can be enough to cause a model to misbehave, which in the context of autonomous driving could potentially lead to catastrophic consequences.

Strategies that improve the training process need to be explored. There is the need to create models that can generalize to situations not included in the dataset. Adversarial defense training methods that have been used to improve the model resilience against adversarial attacks could be applied to improve a self-driving model's ability to generalize to unseen lighting conditions.

1.2 Goals

This thesis aims to improve a model's ability to generalize to unseen lighting conditions and improve its resilience against the selected adversarial attack by adopting an adversarial machine learning defense training method.

We investigate the literature to discover methods to train and validate neural networks. Then we implement an experiment to create a self-driving agent and evaluate its capacity to generalize to unseen illumination intensities and adversarial images. Finally, we select an appropriate adversarial attack and create strategies to defend against it, including retraining the model and exposing it to perturbations while training.

As a result of the defense training, the model should generalize better, such as maintaining the baseline performance when exposed to different light conditions and correctly classifying standard input images while also becoming more resilient to a selected adversarial attack.

The thesis is organized as follows: Section 2 discusses the building blocks of an autonomous vehicle using neural networks. Then the threats against machine learning are discussed and the precautions necessary to deal with adversarial attacks. Finally, those concepts are applied in a real-world setting with a scaled autonomous car, and its performance is investigated. Section 3 will illustrate the approach taken and the detailed phases necessary to accomplish the experiment. Section 4 supplies the outcomes of each stage described in the prior section. Section 5 discusses lessons learned and the limitations of the project. Finally, Section 6 encloses our evaluation conclusions and suggests future work.

2 Background

In this section, a literature review is presented covering the use of computer vision to enable the creation of self-driving agents. We describe the use of neural networks to create machine learning models capable of driving a scaled car autonomously and the need for new methods of training that create more robust models. Additionally, the topic of adversarial machine learning attacks is covered, including the Fast Gradient Sign method, which is the adversarial technique tested in this research.

In section 2.1, several papers on autonomous vehicles were considered. The existing computer vision methods to approach vehicular autonomy were reviewed. Several works that use machine learning to generate self-driving models were considered.

In section 2.2, the background is provided on creating neural networks models, including data collection and training. A neural network type is selected for the experiment, and the motivation to use such an approach is explained.

Section 2.3 examines the vulnerability of machine learning to adversarial examples. The existent attack types are reviewed, and the strategies to mitigate such issues are analyzed. Also, research that demonstrates the use of adversarial defense training methods to improve the robustness of machine learning models is reviewed.

In section 2.4, papers that described the implementation of real-world self-driving platforms were studied. Moreover, the methods and metrics to describe safety and performance were scrutinized.

2.1 Computer Vision in Self-Driving

In 2004, DARPA (Defense Advanced Research Projects Agency) funded a competition to promote autonomous vehicles. The challenge had the prize of one million dollars for the fastest vehicle to complete a 240 km route. No one completed the course in the first edition, and the Carnegie Mellon University's car, Sandstorm, traveled the most distance, completing 11.78 km [5].

The following year's challenge edition doubled the prize and saw five vehicles completing the course. Sandstorm, this time, made to the podium as the second-best contender, and it was only eleven minutes slower than the winner car Stanley, from Stanford University, that completed the 212 km route in just below seven hours. Both Stanley and Sandstorm used a combination of complex sensors such as Lidar and GPS to control the car. The positive results of the challenge stimulated a surge in research on the topic.

It is interesting to notice that, Carnegie Mellon University team, much earlier, in 1989, had already introduced ALVINN, the first neural network-powered autonomous vehicle. The paper was ahead of its time, and it is still to this day relevant. Unlike Sandstorm and Stanley, ALVINN did not use expensive sensors to output the steering decisions for the vehicle. Instead, only one frontal camera was used to feed a convolutional neural network model that was previously trained with many images of human-generated driving data [6].

While pioneering the use of machine learning to vehicle autonomy, the processing power required to train the convolutional neural networks limited the approach from getting traction at the time. However, the meteoric development of computer capabilities enabled NVIDIA's research team to create a convolutional neural network that was able to learn to navigate a car through highways and traffic using cameras solely [7].

2.2 Neural Networks

Neural networks are a subset of machine learning, and both are a subset of artificial intelligence. As the human brain's neural networks, artificial neural networks can learn to classify data through training [8]. In the context of autonomous vehicles, this approach proposes processing sensory inputs to generate lateral and longitudinal control commands using complex mathematical models as a single machine learning task.

Machine learning is generally divided into three major categories, supervised learning, unsupervised learning, and reinforcement learning. The supervised approach uses labeled data to train a model to classify categories of unseen data automatically. On the other hand, the unsupervised approach automatically detects categories and patterns using large amounts of unlabeled data. Finally, the reinforcement learning approach involves training a model from scratch through exploration and improvement of driving policy, utilizing a rewarding mechanism that punishes bad decisions while rewarding good ones, which keep improving the model until an acceptable performance is achieved [4].

Reinforcement learning has been used to train real cars to drive in the physical world. However, this approach is more common in simulation environments. Nevertheless, impressive results have been demonstrated of vehicles trained in simulation but performed well when transferred to the physical environment [9].

In autonomous driving, the supervised learning approach is achieved via behavioral cloning, a form of imitation learning. The model mimics the human driver's behavior by mapping observations and motions. Also known as imitation learning, this approach uses the labeled data provided by the expert as input and convolutional network architecture to generate a machine learning model capable of performing.

2.2.1 Convolutional Neural Networks

Convolutions are linear operations in which a filter function is shaped to detect specific features in the input. The filter function is then applied to the entire input image resulting in a numerical value representing an activation [10].

A convolutional neural network (CNN, or ConvNet) is a deep learning approach to machine learning and is called deep because of the number of additional hidden layers added to learn from the data. Because of its benchmark efficiency, it is widely used to analyze visual imagery.

An illustration of the architecture of a neural network is displayed in Figure 1. In this example, the network takes four inputs, processes them in a hidden layer, and outputs one single machine learning model. For instance, inputs could be pictures, sounds, or any other sensor reading, and the networks can have one or many hidden layers to process and output the model.

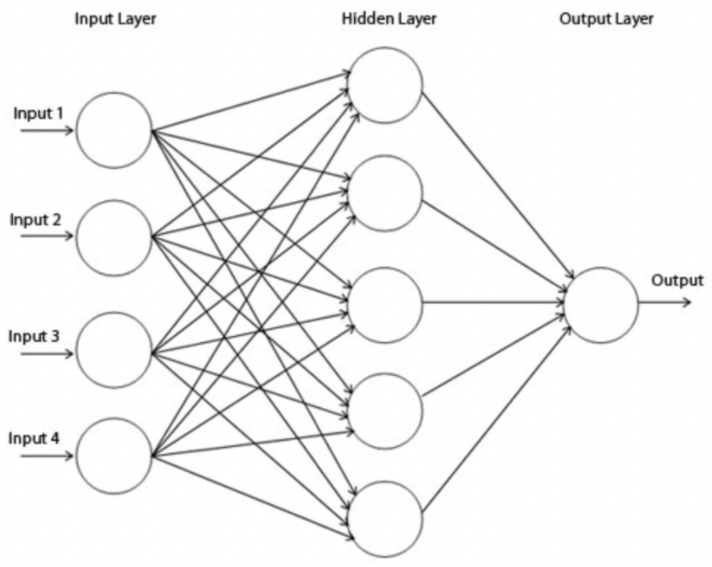


Figure 1. A simple neural network is depicted as having only one hidden layer, and such architectures are often composed of many hidden layers, making them known as "deep" neural networks. [8].

In the context of self-driving cars, the use of deep neural networks is also known as end-to-end methods, it was first used by ALVINN [6], and it is relatively simple to use while yielding good results. End-to-end methods rely on convolutional neural networks to fuse data coming from multiple sources and generate self-driving models. Current state-of-the-art CNN models outperform on the standard the NoCrash urban driving benchmark [11]. The results point to impressive success rates driving in urban environments. However, tests are run in simulation environments, and the models do not show enough generalization to be deployed in the real world, which exposes the need for more work on ways to make models generalize better to the real-world setting conditions.

2.2.2 Self-Driving Models

A machine learning self-driving model is a file that has been created to recognize patterns in a given input. Models are trained with a set of data, and after being trained, the models are expected to reason over unseen data [12]. For example, a self-driving model could have been trained to perform maneuvers based on the images captured by the car's camera. Thus, if the camera captures a left curve, then the model triggers the car left turning and so forth. Autonomous driving models generated from convolutional network architectures have received growing interest since a similar linear model architecture was successfully demonstrated by NVIDIA's paper [7]. The architecture used in the project is displayed in figure 2 uses labeled data which includes information such as the driver's activity like staying in a lane or turning weather conditions, and road type to output a self-driving model.

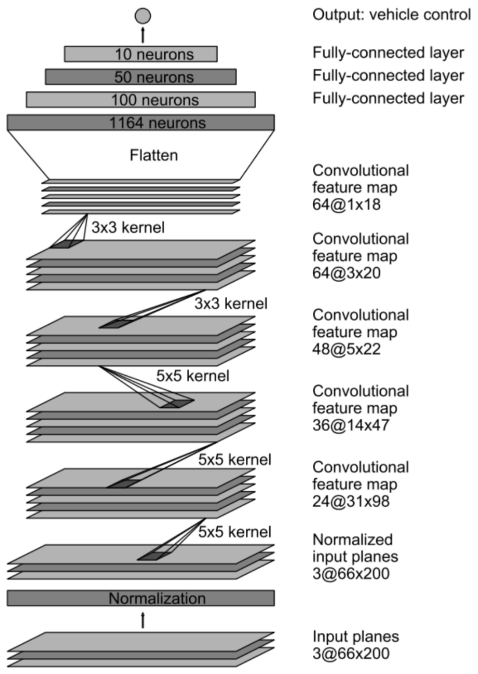


Figure 2. The NVIDIA's CNN linear network architecture was used to train a self-driving model and has about 27 million connections and 250 thousand parameters. [7]

A similar architecture is used in this project to create a self-driving model tested in a scaled vehicle.

2.3 Self-Driving Platform

2.3.1 Scaled Autonomous Car

Using a real-world scaled self-driving car to evaluate the performance of a neural network model's performance was demonstrated by Mahmoud et al. [13]. The study revealed that it is possible to improve a neural network model using image augmentation techniques. By reducing the image sizes, the response rate of the neural models increased, improving safety and the speed of the vehicle. The experiment was conducted using the Donkeycar self-driving platform. Donkeycar [14] is an open-source, easy-to-use, and well-documented Python library that can be used in association with a self-driving 1:10 scale remote control car.

The scaled autonomous vehicle used in the experiment is shown in figure 3. It comes pre-assembled with a Raspberry Pi 4, camera, remote car chassis, battery, and a sensor hat.

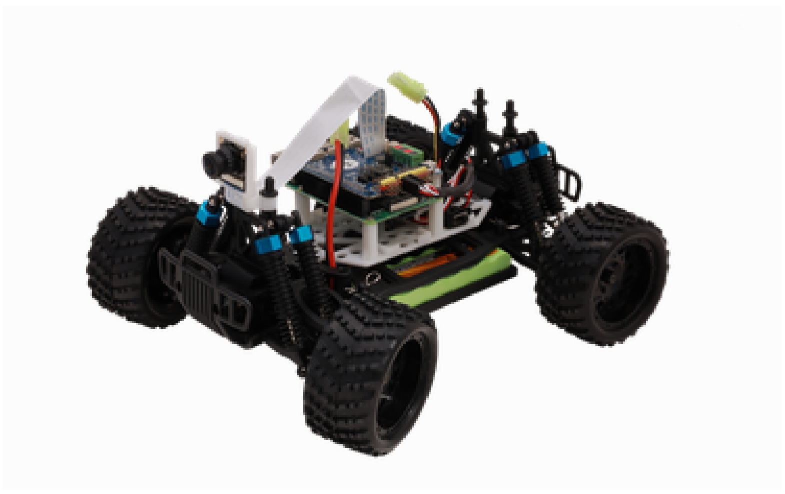


Figure 3. Donkeycar S1 platform was used in the experiment. It comes equipped with a frontal camera, the only sensor used in the experiment. The sensor hat attached to the Pi micro-controller provides an IMU (Inertial Measurement Unit) sensor, not considered in the scope of the experiment.

Donkeycar has an active community of machine learning and autonomous driving enthusiasts who keep improving the library and adding new features. Some improvements include traffic-light and stop sign detection using the external hardware Google Coral [15] to process a benchmark object detection algorithm. This will not be used in this project as the focus of the research is the improvement of the model that predicts the vehicle's navigation.

Another hardware that comes included with the car is the Inertial Measurement Unit (IMU) which is a set of inertial sensors attached to the Raspberry Pi that uses gyroscopes and accelerometers to track the movement of the device [16]. It is possible to use the IMU information for navigation purposes. However, as this study aims to improve the generalization skills to unseen lighting conditions, the only sensor used in this research is the one front camera attached to the vehicle.

The Donkeycar software was built upon open-source libraries such as OpenCV for image processing and Keras for training and running the machine learning models. Keras is a lightweight python neural network library that runs on TensorFlow, which the Donkeycar libraries use to train and run autopilot models. From creating to driving autopilot models, Donkeycar offers a variety of tools that allow fast experimentation. As shown in figure 4, creating models with DonkeyCar involves three steps a) to collect data using the car, b) to transfer the data to a host computer for cleaning and training the model, and c) transferring the autopilot model back to the car ready to be tested.

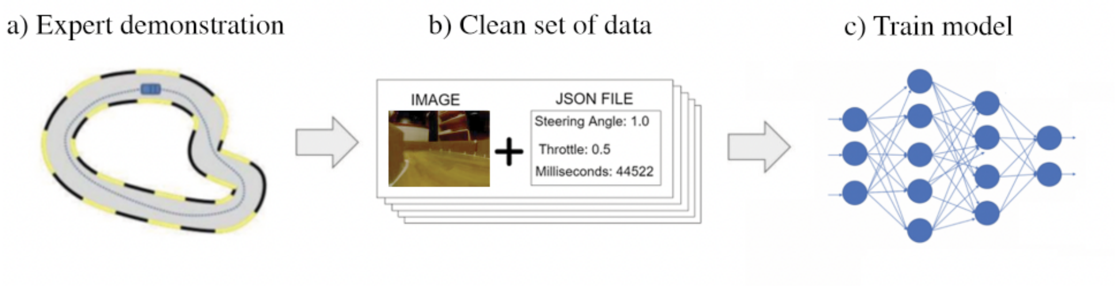


Figure 4. Donkeycar provides tools to collect and label expert demonstrations, clean inaccurate records from the dataset, and train models using deep neural networks.

The Donkeycar autopilot machine learning models are created using a form of supervised learning called behavioral cloning in which the behavior of an expert is recorded and associated with labeled data and then used to train the model, which is the output of the neural network and consists of an array of decision-making algorithms.

2.3.2 Training Neural Network Models

The process of training a neural network involves providing examples of labeled data as input and applying mathematical and statistical concepts such as feed-forward and error backpropagation that gradually reduce the error between the predicted output and the desired output across several iterations over the dataset [8]. Each cycle of complete training iterations is called an epoch, and the training session is composed of as many epochs needed until the model is no longer able to improve its prediction performance. Behind the curtains, similarly to a ranking algorithm that computes that smaller distances are voted to highest ranks, it is possible to train a neural network usin a K-NN (k nearest neighbour) algorithm accepts a set of training data and calculates the learning patterns of the data computing the distance between the in a n-dimensional space,

The Donkeycar models are trained using Keras, a few different architectures are available to choose from, and Keras Linear is the standard neural network. According to the Donkeycar documentation, neural network models trained with this architecture are robust and enable smooth steering. It also performs in low compute environments such as the Raspberry Pi 4 [17]. In this thesis, a model is trained using the Keras Linear neural network, which consists of a single output linear convolutional neural network composed of five convolution layers followed by two dense layers before the output of two dense layers with one scalar output each with linear activation for steering and throttle.

The Donkeycar libraries enable the creation of well-performing models using a relatively modest amount of data, which means that good models can be created using five to ten thousand labeled images. However, such models seem to overfit the conditions exposed during data collection, including light conditions. For instance, a model created with data collected during daytime is unlike to perform well during nighttime. In other words, the luminance levels seem to affect the model's performance considerably.

Ideally, to create a robust model skilled enough to generalize well to lighting changes, the model would use samples of multiple different lighting conditions. However, recording large amounts of data in different circumstances is not always feasible. Alternative training methods that help models perform in unseen lighting conditions need to be investigated.

This research project explores the use of adversarial defense training to improve the capacity of a machine learning model to generalize to real-world changing lighting conditions. Different strategies to improve the model robustness, including image augmentation techniques such as image cropping, are commonly used and available in the Donkeycar platform. However, such methods are not investigated in this research as the focus is solely to measure the impact of adversarial training in the overall performance of the models.

2.4 Adversarial Machine Learning Attacks

Adversarial machine learning attacks consist of methods to generate adversarial examples purposely designed to cause a machine learning model to fail in its predictions. However, despite causing great harm, the perturbations generated with the adversarial images can be subtle enough to be imperceptible to the human eye.

Adversarial attacks are organized into evasion, poisoning, and extraction attacks. Piazzesi et al. [18] explore in their study only evasion attacks because of their likelihood to compromise safety by being carried out on a self-driving agent while it is running. Such attacks can be white-box and black-box attacks. While the white-box attacks require having full access to the architecture and parameters of the model, the black-box attacks do not require knowing the model structure and architecture.

Adversarial examples have been widely used to fool machine learning models successfully. However, there is no clear understanding as to the impact of adversarial images on the machine learning model's ability to generalize to real-world ever-changing lighting conditions.

2.4.1 Adversarial Defense Method Training

Rosebrok [19] demonstrated the possibility to improve the model's ability to generalize and defend against adversarial attacks using adversarial images during the training process. The concept consists in modifying the standard training procedures adding artificially generated adversarial images. Training CNNs with mixing standard and adversarial images has improved the model's robustness against adversarial images in simulation environments.

In Figure 5 the adversarial training process is shown. Adversarial images are mixed with standard images to compose the set data used by Keras to train the neural network.

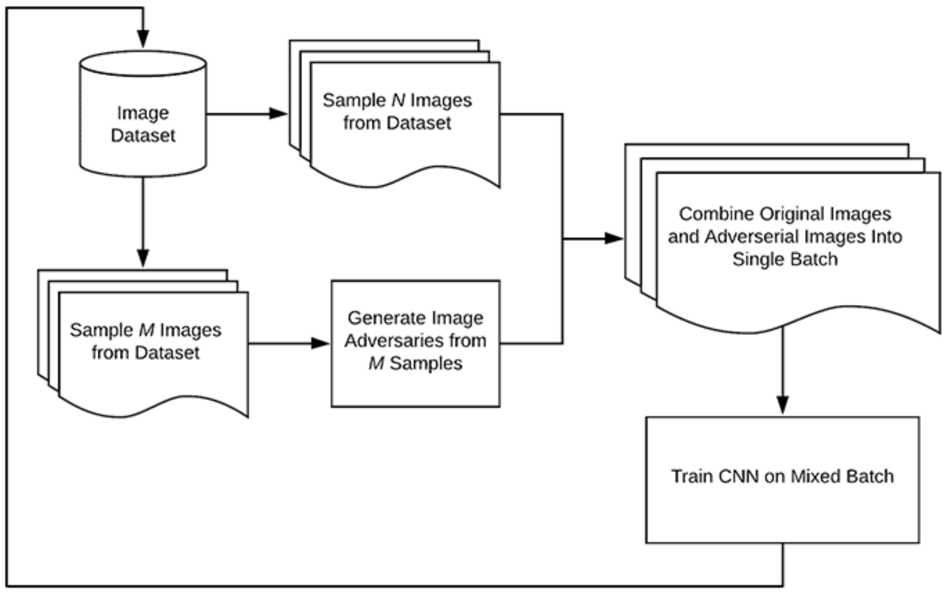


Image Dataset
Sample *N* Images from Dataset
Combine Original Images and Adverserial Images Into Single Batch
Sample *M* Images from Dataset
Generate Image Adversaries from *M* Samples
Train CNN on Mixed Batch

Figure 5. The adversarial defense training method demonstrated by Rosebrok involves generating a batch of adversarial images and combining them as a single batch used for training the model. [19]

To generate a total of N adversarial images, a model must be combined with the adversarial attack Fast Gradient Sign Method (FGSM). The FGSM [20] is a reliable way to generate adversarial examples that cause models to misclassify their input.

See in figure 6 how Goodfellow et al. first demonstrated that a discrete perturbation could be constructed by the model and added to the input causing the neural network to fail the classification task.

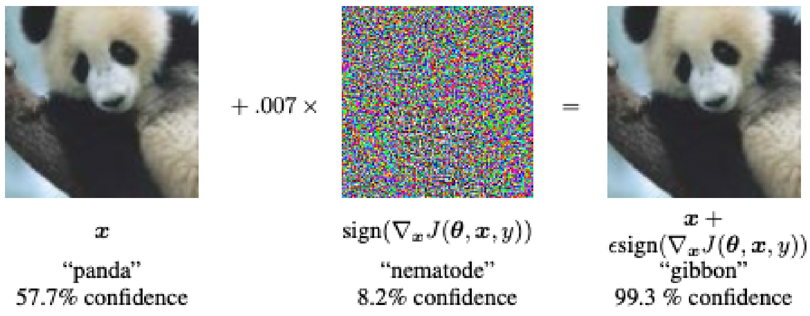

$+ .007 \times$
$=$
$\boldsymbol{x}$
"panda"
57.7% confidence
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

Figure 6. In the left-hand side of the image, a model is used to correctly classify an image of a panda in the class "panda" with 57.7% of confidence. In the middle, a gradient descent noise vector is constructed using "0.007" as the epsilon value that defines the norm of the perturbation, which is itself erroneously classified as a type of worm called a nematode. The noise vector is added to the original image, resulting in the same model misclassifying the panda as an ape species called gibbon with 99.3% confidence. (Image source: Explaining and Harnessing Adversarial Examples)[20]

The fast gradient sign method is named because it is the simplest and fastest method to construct adversarial examples. The method adds perturbations and linearizes the cost function in the gradient direction. The function proposed by Goodfellow et al. specifies that adversarial image is the results of the sum of the image $X$with the sign function, which takes as the input parameters the accurate label of X and J is the cost function that was used to train the model and $XJ()$ represents the gradient of X, and $Xadv$ is the resulted adversarial image from the original image X [21]

$$Xadv = X + sign(XJ(X, y))$$

Some of the FGSM advantages include low computational complexity and high transfer rate, making them appropriate for use with low processing power computers such as the Raspberry Pi. On the other hand, some disadvantages of the fast gradient sign method are a low attack success rate and perturbations on a single image. In contrast, there are universal image-agnostic perturbation attacks methods that fool classifiers by single adversarial perturbation to all images [21].

It is interesting to notice, that the concept of using gradient techniques differ from other image augmentation forms such as brightness or blurring techniques, an image gradient is defined as a directional change in image intensity, the gradient is a representation of the image comparison of front and background in a form of edge detection recognizing areas of high or lower constrast. At each pixel of the input image, a gradient measures the change in pixel intensity in a given direction. By estimating the direction or orientation along with the magnitude (i.e. how strong the change in direction is), we are able to detect regions of an image that look like edges. [22]

In practice, image gradients are estimated using kernels and finding the structural components of the image. For gradient estimation, the first step is to simply find and mark the north, south, east and west pixels surrounding the center pixel. The gradient magnitude is used to measure how strong the change in image intensity is. The gradient magnitude is a real-valued number that quantifies the "strength" of the change in intensity. The gradient orientation is used to determine in which direction the change in intensity is pointing. As the name suggests, the gradient orientation will give us an angle that is used to quantify the direction of the change. Applies convolutions to define orientation and magnitude of the gradient by dividing the image pixes in kernels containing values, each step of the way examines the directions and compute the difference compute veritcal and horizontal changes and define intensity, and direction of pointing.

The FGSM is considered a white-box, gradient-based, non-targeted attack. Unlike black-box attacks where the attacker has no access to the model infrastructure, white box attacks are situations when the attacker has access to the target model's parameters and architecture. As this study investigates the possibility of using adversarial defense training methods to improve the performance of the self-driving model enabling generalization to external lighting conditions in a real-world setting, having access to the model is expected.

As this thesis analyzes the impact of models running in low processing power micro-controllers, low computational complexity is crucial. The disadvantages of the method are not deterring as the low attack success rate does not out-weight the possible benefits of an improvement in a training process. Therefore, FGSM is the adversarial machine learning chosen to improve the models' performance in unseen lighting conditions.

2.4.2 Safety and Performance

Zhang et al. [23] propose in their study an end-to-end evaluation framework with a set of driving safety performance metrics. The research measures the impact of adversarial attacks on the driving safety of vision-based autonomous vehicles. The collision rate and the success of route completion rate were some of the metrics used to validate the results under the perturbation attack.

The impact of adversarial attacks in the performance of self-driving models was tested by Piazzesi et al. [18]. Their study analyzed how autonomous driving model agents deployed in a simulation environment react to several different adversarial attacks. The study results were measured by analyzing the rate of collisions in different circuits in a specific set of pre-defined conditions. The adversarial attacks significantly impacted the driving models, leading to collisions in all four test runs.

The manual recovery from collisions, also known as human interventions, together with the lap time, was also used as a measure of success in [24]. The study compared the performance of different self-driving convolutional neural network models in the same circuit using standard training techniques. In the study, the author demonstrated that the best-performing model could drive five laps without human intervention. They revealed the capability of end-to-end neural networks models as a feasible option for autonomous driving research.

All in all, the number of collisions accounted in N elapsed circuit have been used successfully used to demonstrate the capabilities of self-driving models. In this study, walls were built surrounding the track lane, and crashed against the wall will be measured and compared to evaluate the performance of the different models tested in the experiment.

3 Method

This section describes the plan and the procedures to use adversarial defense training methods to train a CNN supervised classifier autopilot using behavioral cloning with a 1:10 scale car based on the open-source platform Donkeycar.

The project aims to train deep learning neural network models to drive autonomously and then improve their generalization ability to unseen lighting conditions. The method is divided into four distinguished main stages, including data collection (expert demonstration), training the models using the built-in Donkeycar convolutional neural network, testing the models self-driving abilities in the circuit, and finally, evaluating the performance of the cars using the self-driving models.

The research project was formatted in a factorial design, with multiple independent variables being tested. The summary of the main stages and how the independent variables of experiment affect the dependent variables are specified in table 1.

Data collection	
Independent variables	**Values**
Light	low
Laps	20 and 37
Images	20 per second
Label 1 (throttle)	[0, 1]
Label 2 (steering angle)	[-1, 1]
Two datasets	small and large
Training - Four different settings	
Independent variables	**Values**
Dateset	small and large
Image data augmented by adversarial attack data	no, yes
Testing - Four different settings	
Independent variables	**Values**
Laps	2
Lights	low, high
Vision	uncorrupted, corrupted
Evaluation	
Dependent variables	**Values**
Collisions	[0...n]

Table 1. The study's independent variables were manipulated to measure the results of the dependant variables, such as the number of collisions during a two-laps performance. The throttle value ranges from 1 when at full speed to 0 when stopped. The steering angle is -1 when at 60 degrees left turn and 1 when for 60 degrees right turn.

Initially, data is collected using a frontal RGB camera attached to the scaled car while the expert drives along the circuit using a joystick, the behaviour of the expert is recorded in a form of a batch of images captured at a rate of 20 images per second, and each image contains a corresponding steering angle, throttle and timestamp. Despite the expert's best effort, collisions or near misses are inevitable, thus the data must be cleaned, and inaccurate records are removed from the recordset before it can be used in the next phase, training the neural network.

In the next stage, the cleaned data, which includes a batch of labeled images, is used to train a Donkeycar built-in convolutional neural network architecture with imitation learning as described in section 2.3.2. Testing the newly created self-driving models will follow in four distinct conditions.

The first and second testing conditions will evaluate the performance of the models when it is exposed to the same lighting conditions used during the training phase. The third and final conditions will evaluate the model while exposed to unseen higher lighting conditions. The results of the first testing phase will establish the baseline. Depending on the results, if the model fails to generalize to unseen lighting conditions, then an adversarial defense training method will be employed as an attempt to improve the model generalization skills.

During the testing phase some of the conditions include submitting the models to the task of classifying a corrupted vision which is created by generating adversarial attacks like modifications of the real-time image on the fly.It is important to salient the fact that when using the corrupted images, every single image exposed to the model for classification is corrupted.

One control condition was selected for internal validity compared with the other experimental conditions. The control conditions is defined in the testing the model M-TS, in which performance measurements are collected after exposing the model to lower lighting levels, using a standard training method described in 3.2.1 which uses the small dataset and performing without direct adversarial attacks.

The models' performance are collected in the testing phase, and then compared against the baseline to evaluate and measure success. Details of all the experiment stages are described in the following subsections.

3.1 Experimental Setup

To train an autopilot with Keras, the Donkeycar recommends collecting from five thousand images as the minimal amount to produce a good model[25].

To collect the data necessary for the experiment, the testbed, composed of a front-facing wide-angle camera, captures two sets of data collected in two different sessions, the first containing almost five thousand images and a second larger dataset containing just under ten thousand images. The need for two datasets with different sizes is to compare the results and evaluate the impact of the volume of data on the model's overall performance in two distinct scenarios.

3.1.1 Description of the Testing Conditions

The experimental results are based on the platform's performance described in the previous subsection. Five points cover a range of locations in the track to illustrate different luminance conditions. The points consist of locations captured by the platform that drives in the circuit until it cannot proceed due to a collision with the wall. In all conditions, the shape of the circuit remains unchanged. However, the light intensity is modified.

The model is trained only with the data captured in the first conditions in lower lights. Thus, the models will have their generalizations skills tested in unseen luminance conditions when exposed to the higher-light conditions. The vehicle's camera captures twenty frames per second. The dataset is stored, cleaned, and used in successive training.

An expert policy is collected in two different sessions. The first session performs 20 laps, and the second session collects 37 laps, gathering approximately 5000 images and 9000 images, respectively. All images are labeled with their corresponding steering angles and throttling values. The steering has number values within the range of [-1.0, 1.0], in which positive values correspond to turning to the right, and negative values correspond to turning to the left. Throttle values are within the range of [-1.0, 1.0], in which zero the vehicle is stopped, positive values mean forward-moving while negative values mean backward moving. One caveat is that only forward moving is considered in this study. Thus, human intervention was necessary to recover the car from collisions and recenter it in the lane.

Four different autopilot models are created. The first model is trained with a smaller dataset. The second model uses the exact same data as the first model but is created using an adversarial defense training method. The third model is trained with a larger dataset, and the final model uses the same data as the third model but is created using the same adversarial defense training method used in model two.

Defense methods are applied in the data to improve the expert policy's generalization skills to unseen lighting conditions. Therefore, all four models are tested and evaluated while exposed to four different conditions. In the first condition, the autopilot models are tested in lower lights, with the same level of illuminance captured during the data collection phase. The second condition involves using the same lower-light environment to run the model while under the same adversarial attack used for the defense training method. The model is tested in an unseen higher-lighting environment in the third condition. In the final condition, the models are evaluated in the same previous unseen higher-lighting environment but under the adversarial attack.

To test and evaluate the different models on the established conditions, an adversarial image generator must be integrated with the platform. Specific implementation settings and values will differ depending on the experiment environment. This step includes testing and performing iterations to find appropriate values.

Once the driving platform is implemented and integrated with the adversarial defense training methods, metrics must be provided to allow comparison. The metrics should quantify the autopilot model's performance and the circuit's lower and higher lighting conditions.

The final purpose is to create a proof of concept strategy for training more skillful machine learning models capable of generalization to unseen lighting conditions.

3.1.2 Self-Driving Platform

The testbed architecture is designed for training and testing behavior cloning models on car-like robots, such as the Donkeycar. The model car is equipped with a mono frontal wide-angle RGB camera, capturing 120x160 images labeled with the motor throttling and steering angle values used to classify, create, and test a self-driving agent model.

A Raspberry Pi 4 microcontroller is used to run the Keras deep learning libraries using a Tensorflow backend to trigger commands to servo-controlled steering and thrust motors. The model is trained on an indoor circuit constructed with white gaffer tape and cardboard walls. Although the Donkeycar design allows for the integration of sensors such as LiDAR and IMU, such sensors will be kept out of the project's scope, as the object of analysis is solely the input of the RGB camera system instead.

3.2 Training

The images captured by the camera are paired with corresponding labels with throttle and steering angle values, which will then be used to train a CNN model capable of steering a car in a circuit. The approach taken to train the model is behavioral cloning, also known as imitation learning, which is a form of supervised learning in which labelled data is created by recording the behavior of an expert. A combination of opensource projects is used to create the neural network models including using Keras and TensorFlow functions to convert the images to arrays, allowing to load input images using openCV and represent them as a Numpy array and then converted them to a format that is compatible with Keras and TensorFlow which represent images as tensors containing the image high, width and channels.

Deep learning is applied for training a convolutional neural network on the image dataset, which consists of the circuit and its surroundings, or everything that was captured by the camera attacked to the car. The training and validation are performed using a split of the dataset with a ratio of 0.8:0.2, and the validation occurs during a set of iterations called epochs. The goal is to train the CNN to identify to which a class a given image belongs to.

After the neural network architecture of the autopilot has been successfully trained, it classifies the input image and predicts the most suitable throttling value and steering angle. The convolutional neural network used in this project was the default Donkeycar linear model, consisting of five convolution layers, a flattening layer, and three fully-connected layers.

3.2.1 Standard Model Training

The standard behaviour cloning training approach to create a CNN model capable of steering a car in a circuit presenting a behavior similar to of an expert is a process which includes first capturing images using the opensource project OpenCV as part of the vehicle and then storing the images to a path which includes a json file containing the class ground truth labels. The images are captured on a range of 20 images per second. As demonstrated in figure 7 the standard training procedure, a sample portion of the dataset is selected for training. Donkeycar uses Keras to randomly select 80% of the images as the training set and 20% as the validation set.

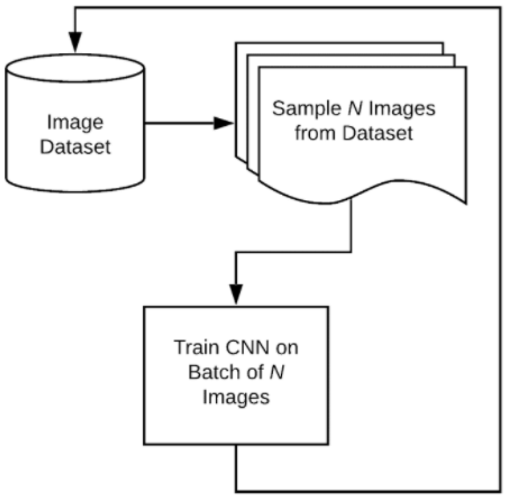


Image Dataset
Sample *N* Images from Dataset
Train CNN on Batch of *N* Images

Figure 7. The standard training, N images are sampled from a dataset to train a CNN. (Image source: The normal process of training is illustrated by [19] in his article Mixing normal images and adversarial images when training CNNs.

The final stage of the standard process training includes the use of a model fit function which takes as parameters the training set and the validation set. The covolutional network is built using Keras and TensorFlow image processing filters chained together, such as using classes to implement 2D convolutions, dropout and classifiers functions such as dense and activation.

The problem of this approach is that limits the model to perform well only under the circumstances presented in the dataset and the model usually fails in unseen conditions such as different lighting conditions, resulting in a model that perform really well in conditions similar to the original dataset but performs poorly in higher lighting conditions. A different approach using a mixed of adversarial images and normal images could be used to augment the dataset and improve the model generalization skills.

3.2.2 Adversarial Defense Training

The adversarial training defence method to train deep neural networks suggested by [19], includes mixing adversarial images to the dataset to improve the robustness of the model and generalization skills. The process is similar to the previous standard method, but slightly more sophisticated. As illustrated in figure 8 the adversarial defense method includes combining the original images and synthetically generated adversarial images into a single dataset used to train the CNN.

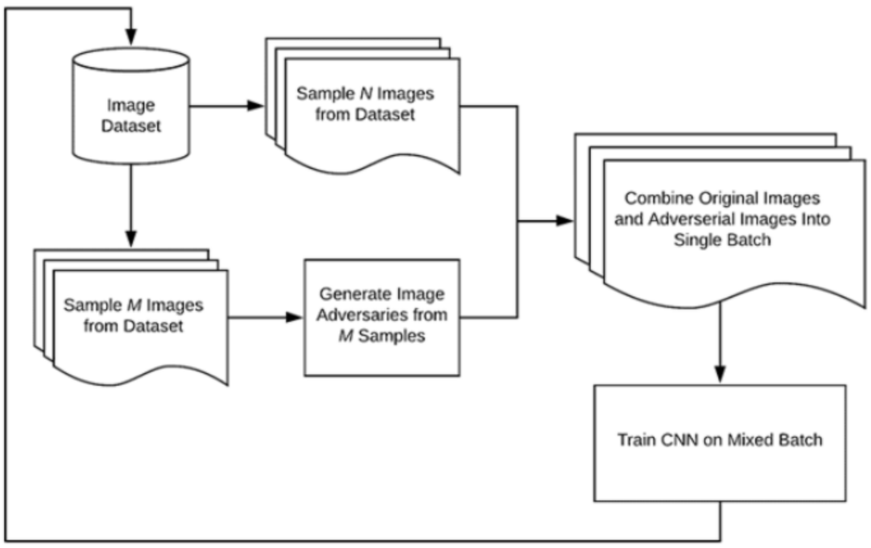


Image Dataset
Sample *N* Images from Dataset
Sample *M* Images from Dataset
Generate Image Adversaries from *M* Samples
Combine Original Images and Adverserial Images Into Single Batch
Train CNN on Mixed Batch

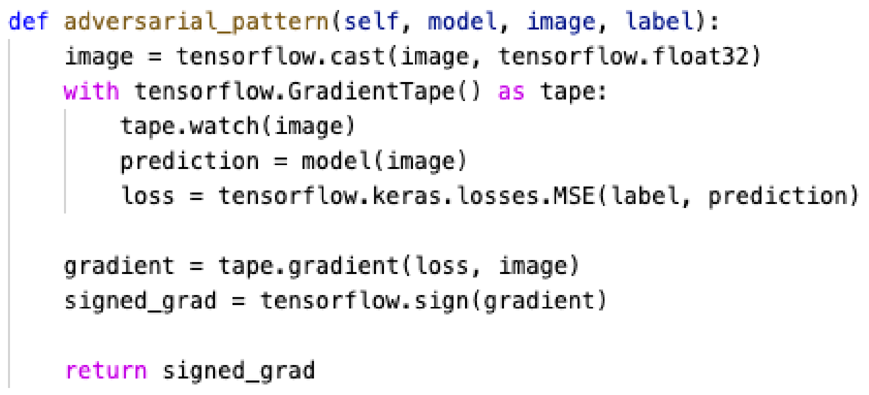
Figure 8. Adversarial defense training. (Image source: mixing normal images and adversarial images is illustrated by [19] in his article about mixing normal images and adversarial images when training CNNs.

The training pipeline itself is not changed when comparing to the standard training procedure, in which a convolutional neural network architecture is trained on sample images of a dataset. However, the dataset is composed of an equal split of normal image and adversarial images, which are batch of generated perturbed images, which were construct using the Fast Gradient Sign Method attack in addition with the CNN model itself.

Although, adversarial images aspiring a negative connotation, mostly because the vulnerability of deep learning models to them, advsarial images, can theoretically be used as an advantage to construct models capable of generalizing better to unseen conditions. Thus, the working-horse of this approach is the function to generate batches of adversarial images that could be used to create more robust models.

3.2.3 Adversarial Images Batch Generator

The adversarial defense training method is defined around the idea that the original dataset of examples can be augmented prior to training the model, without having to spend extra time collecting data with an expert. This method relies on the use of the FGSM function, as shown in figure 9 to generate an adversarial pattern that can be added to the image, the function takes as input the CNN model which was previously constructed the original dataset, a sample image and the ground truth labels to return a perturbation vector.

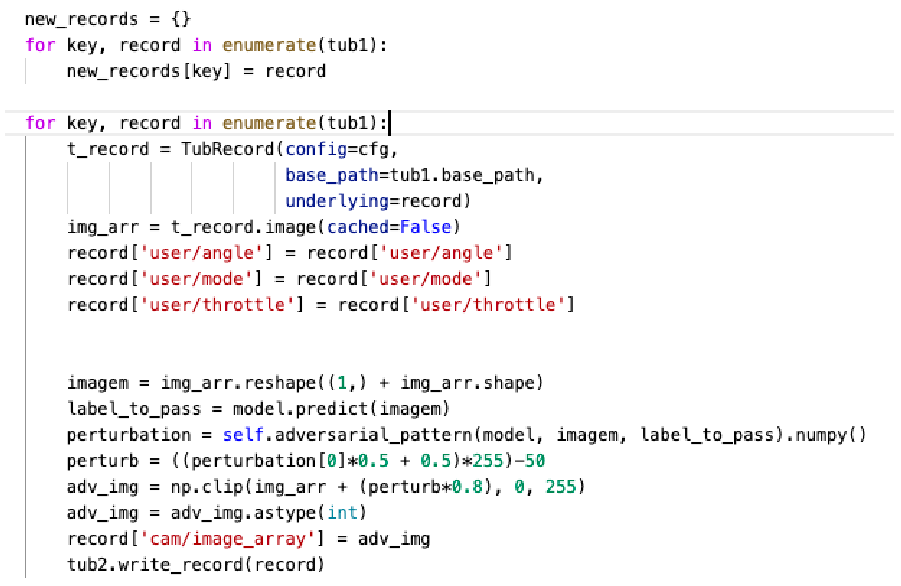


```python
def adversarial_pattern(self, model, image, label):
    image = tensorflow.cast(image, tensorflow.float32)
    with tensorflow.GradientTape() as tape:
        tape.watch(image)
        prediction = model(image)
        loss = tensorflow.keras.losses.MSE(label, prediction)

    gradient = tape.gradient(loss, image)
    signed_grad = tensorflow.sign(gradient)

    return signed_grad
```

Figure 9. The function to generate an adversarial pattern takes an instantiation of the cnn architecture we had trained on the original dataset, the image we want to contruct adverarial image)[19]

The function to generate the adversarial pattern first cast the input image to a float 32 bit data type, then monitor gradience by whatching the image and use the model to make predictions on the image and then compute the loss with respect to the original class label, passing the ground truth label and the prediction, computing the means square loss MSE function which returns a loss. Next it calculates the gradience using the loss and the image and compute the sign gradient that is returned by the function.

Having the function to generate adversarial images is only part of the process of the adversarial defense training, the next we need to use the function to generate a set of image adversaries and fine tuning the cnn including the adversarial images and the standard images to the training pipeline making it generalize better and requiring less expert demonstration time. Instead, the complete training procedure includes generate adversarial images and using them as training images. As demonstrated in figure 10 the process of creating a batch of adversarial images consists of iterating through each image available in a Donkeycar tub, which is a folder containing the images and labels generated by the expert, and making a copy of the image in a new tub, however, adding the generated adversarial perturbation.



```
new_records = {}
for key, record in enumerate(tub1):
    new_records[key] = record

for key, record in enumerate(tub1):
    t_record = TubRecord(config=cfg,
                         base_path=tub1.base_path,
                         underlying=record)
    img_arr = t_record.image(cached=False)
    record['user/angle'] = record['user/angle']
    record['user/mode'] = record['user/mode']
    record['user/throttle'] = record['user/throttle']


    imagem = img_arr.reshape((1,) + img_arr.shape)
    label_to_pass = model.predict(imagem)
    perturbation = self.adversarial_pattern(model, imagem, label_to_pass).numpy()
    perturb = ((perturbation[0]*0.5 + 0.5)*255)-50
    adv_img = np.clip(img_arr + (perturb*0.8), 0, 255)
    adv_img = adv_img.astype(int)
    record['cam/image_array'] = adv_img
    tub2.write_record(record)
```

Figure 10. batch adv. (Image source: Explaining and Harnessing Adversarial Examples)[20]

The perturbation is used to construct the adversarial image taking the signed grad and multiply by the epsilon factor, which is the gradient update to be added to the image, then convert to a numpy array from the tensorflow volume and return image adversary. Higher is the epsilon factor, also higher are the chances of the manipulation been able to be perceived by the human eye. Notice that is keeps the correct label despite the adversarial image. See in figure 11 the comparison of the standard image with an image that has been generated using the adversarial perturbation.

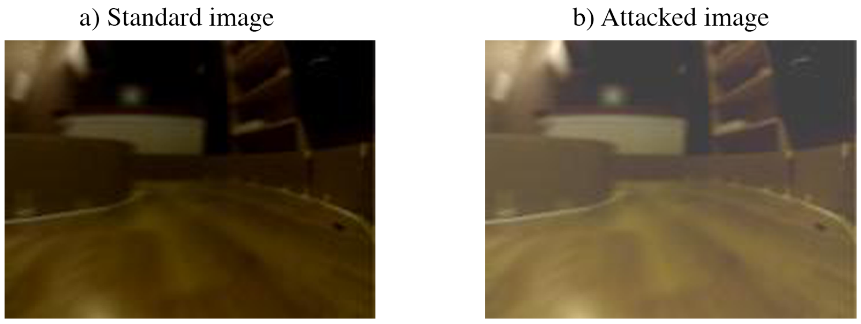


Figure 11. In the left a standard image that was captured in lower-light conditions. In the right, the exact same image has been duplicated applying the addition of the FGSM perturbation.

The function receives as a parameter the model, which we had generated previosly, a base input image, the delta which is the noise vector used for construction the adversarial image and the label. The perturbation is added to the base image with the delta vector and will clip all the pixel values to the range from 0 to 255 and convert the image from floating point data type to unsigned 8 bit integer.

At the end, a batch of the same size of the original dataset is generated composed of adversarial images, which are then mixed with the orifinal dataset and the training images will be sampled randomly from the standard images and adversarial images examples.

Limitations to this approach dependend on the model and the optimizer, to generate a robust model that is able to generalize better to adversarial conditions and have a more accurate driving it might be necessary to fine tuning the model using different epsilon factors with the FGSM to generete adversarial images.

The goal is to solve the problem of the model be overfiting to the trained conditiond, when overfitting your model, the model is close to memorise the training set, not so good at generalizing to unseen images, techiniques including data augmentations to validation and training set splits might reduce the impact of overfiting.

After all the neural network architecture of the autopilot have been successfully trained, including using standard trainign and adversarial defense training, they should be able to classify the input image and predicts the most suitable throttling value and steering angle. The convolutional neural network used to train the models in thi project was the default Donkeycar linear model, consisting of five convolution layers, a flattening layer, and three fully-connected layers. The performance of all models are recorded and evaluated in regards to their accuracy in maintaining a steering clear of collitions.

3.3 Design and Implementation

The experiments were performed on the autonomous car under sixteen different conditions to evaluate the performance of four distinct models. The goal is to demonstrate the impact of (a) lighting conditions, (b) the size of the data set, and (c) adversarial training methods on the performance of the driving algorithms in a scaled real-world setup.

3.3.1 Testing M-TS

Each testing phase creates a different model and tests its performance under four distinct conditions. In the first testing phase, a model trained with a smaller dataset consisting of images collected during 20 laps is first deployed for testing in the same lighting conditions exposed during training. The metrics values established in this first condition will define the baseline. The baseline metrics will be used to measure if the process is successful or not. In addition, to attain success, the model would need to improve its performance concerning the baseline. A model should perform at least two laps without collisions or human interventions if well trained.

Testing conditions						
Model	Laps	Images	L	LA	H	HA
M-TS	20	4750	19.6	19.6	81.2	81.2

Table 2. The model was tested in four different conditions. This model was generated using the sample data collected over 20 laps of expert demonstration in a total of 4750 images. The four conditions tested include lower lighting (L), lower lighting under malicious attack (LA), higher lighting levels (H), and higher lighting with an attack (HA). The average luminance in lower lighting is 19.6 lux, whereas the average luminance rises to 81.2 lux in higher lighting.

As shown in table 2, in the second stage of testing 1, the same model has tested again in the same lighting conditions but being under an adversarial attack. The resulting metrics will be compared against the baseline. For the adversarial attack to be considered successful, the model performance deteriorates compared to the baseline. It is expected that the model would misbehave when exposed to adversarial images.

The next stage includes testing the models in unseen conditions. The circuit is exposed to higher levels of light intensity. The model's behavior is unknown. However, it might become unstable. Hence, the need for mechanisms that safeguard against such adversities.

Finally, the model is tested in the last condition, in which the same higher levels of light exposure are maintained from stage three, but now additional adversarial attacks are also used against the model, which is expected to misbehave if the adversarial attacks become successful.

This step aims to identify whether the performance of a self-driving model is affected by unseen illumination levels or a selected adversarial attack.

3.3.2 Testing M-TL

The seconds phase of the testing phase involves creating a new model trained with a larger dataset consisting of images collected during 37 laps, twice as much as the first model. The goal is to understand if the addition of more data alone would be sufficient to the creation of a more skilled model capable of generalization to unseen illumination levels.

The average lighting levels were kept the same in each condition during all tests. However, the model tested in this staging used a more extensive dataset, as shown in table 3, while maintaining a traditional training method.

Testing conditions						
Model	Laps	Images	L	LA	H	HA
M-TL	37	8926	19.6	19.6	81.2	81.2

Table 3. The model tested in the second phase of the experiment was trained with the same standard training procedure used in the first model, but additional laps were added to the training sample. The average luminance levels are the same as the previous testing.

The model is first deployed for testing in the same lighting conditions exposed during training. If well trained, using only standard approaches, the third model should perform at least two laps without collisions or human interventions.

The second condition involves deploying the adversarial attack against the model trained with the larger dataset, initially maintaining the lighting levels. For the third condition of the testing 2 phase, the model is exposed to unseen higher lighting levels. Moreover, the higher lighting levels are maintained for the final condition, and an additional adversarial attack is deployed against the model while it is performing in the circuit.

This step aims to identify whether the performance of a self-driving model is affected by the amount of data used for the training.

3.3.3 Testing M-TSA

The third testing phase consists of the duplicate records used to train the first model M-TS, which is now used to train a new model M-TSA. However, this time an adversarial training method is used instead of the standard training method used in testing 1. The intention is to create an improved model that outperforms the standard model in generalizing to unseen lighting conditions and resilience against the adversarial attack.

As shown in table 4 the dataset has increased in size while keeping the same number of laps. This occurred because the extra added images were artificially generated using the Fast Gradient Sign Method mentioned previously.

Testing conditions						
Model	Laps	Images	L	LA	H	HA
M-TSA	20	9500	19.6	19.6	81.2	81.2

Table 4. The model tested in the third phase of the experiment was tested in the same conditions as the previous model. However, an adversarial defense training method was used, and an additional batch of adversarial images was included in the training sample dataset.

Similar to the previous stage, the new model is tested in four different conditions, the first being the standard conditions, which means the same lighting conditions that the model was trained with. The second condition of the third testing phase involves deploying an adversarial attack against the model while it is performing in the circuit, maintaining the standard low light levels used during training.

In the third condition, the model is performed under the exposure of unseen higher lighting conditions. This stage verifies if the proof of concept defensive training method improves the model's efficacy to generalize to unseen illumination levels.

Additionally, the final condition of this testing phase maintains the higher lighting levels of the previous condition to evaluate the model's performance trained with an adversarial defense method against the same adversarial attack that it has been trained with while underexposure unseen lighting intensity.

3.3.4 Testing M-TLA

The final testing phase uses the same larger dataset used in the testing 3 phase to create a model trained with adversarial defense methods. The goal is to identify the impact of creating models trained with adversarial defense methods using a larger dataset.

As shown in table 5 the model tested in this stage is created from the most extensive dataset used in the experiment, containing 17852 images, including adversarial images artificially generated. The average lighting conditions remained the same as in the previous testing phases.

Testing conditions						
Model	Laps	Images	L	LA	H	HA
M-TLA	37	17852	19.6	19.6	81.2	81.2

Table 5. The final model was created using the most extensive dataset. The sample training data contains examples of driving behavior for 37 laps. Additionally to the standard samples, adversarial images were added to the training batch. The lighting levels were consistent across all test phases for each test condition.

The same four conditions applied in the previous testing phases are also applied in the testing 4 phase. Initially, the model is tested in the same conditions that were used to collect the data, including lower lighting conditions. The following condition involves preserving the lower lighting levels and analyzing the model's performance while under attack by adversarial images.

For the last two conditions of testing 4, the model is exposed to unseen higher lighting levels, and an adversarial attack is also generated against the model to test it under the last condition of testing 4.

The goal of this phase is to understand if the model's performance is affected by the use of a larger dataset for generating models using adversarial training methods.

3.3.5 Baseline Metrics

The performance of the model in the same circunstances it was exposed during the training will define the baseline metrics. Using standard training techniques should be enought to create a model that run precise manouvers collision free in a circuit for at least two laps. It is unknows how the models will perform in unseen lighting conditions, but we expect that under the same conditions the model was trained the driving should be all correct and collisions should be zero. And the number of collisions occured when running for two laps in the circuit are the quality yardstick used to measure the performance of the models.

3.4 Evaluation

The last stage of the method is evaluating the previous phases of the experiment. The success criteria must be defined prior to measuring the results.

Adversarial defense training methods can be considered successful if:

- The performance of the model in the standard scenario is unaffected.
- If the model's performance is improved compared to the baseline.

- The model can drive at least two laps autonomously in the circuit in scenarios A and B without collision.
- The implementation of the method is straightforward and can be duplicated.
- The training method is helpful and improves the generalization of the model to unseen lighting conditions, having the same amount of initial data.

It is important to note that the evaluation will only be based on stage one of testing. If the model evaluation shows positive results, it can be tested against an adversarial attack while performing in the circuit. To evaluate success criteria in testing 2, the model must at least outperform the model that used formal training and complete two laps with fewer collisions.

4 Results

Four tests were performed to test each neural network model in four different conditions. This section analyzes the experimental setup and training methods used in the process. The results of testing the models in different conditions are divided into four main categories:

- Testing M-TS - measures the model's performance created with a more minor data set using a normal training process in standard conditions, which means only training the models using image samples of lower-lighting exposure.
- Testing M-TL - the same normal training process and standard light conditions are used to train the models tested in this stage. However, a more extensive data set is used for the training instead.
- Testing M-TSA - analyses the performance of a new model created using the same more minor data set used in testing 1, but a slightly more complex adversarial training method is incorporated.
- Testing M-TLA - the neural network model is created again, implementing the adversarial training process using the same more significant data set used in testing stage 2.

The aim is to demonstrate the navigation capabilities of the driving models and their ability to generalize to unseen lighting conditions, to demonstrate the impact of adversarial attacks on the performance of the model, and to show how adversarial training is used to improve the generalization skills of a model in a scaled real-world setup.

4.1 Experimental Setup

Each model was tested into two main conditions: when the models are performed under the same light exposure levels used during training and when the lighting conditions are changed to unseen higher levels. The models are also tested while being under attack in both lighting conditions.

Conditions	L	LA	H	HA
Lighting Levels	Low	Low	High	High
Adversarial Attack	False	True	False	True

Table 6. The four models tested in the experiment are tested in four conditions: lower-lights (L), lower-lights under attack (LA), higher-lights (H), and higher-lights under attack (HA).

The summary of all tested conditions is displayed in table 6. All the data collected in this experiment was under the lower-light levels. Therefore, the baseline is defined, measuring the model's performance in this condition. To achieve consistency and precision over the lumminance levels and prevent contamination with natural light exposure, the data was collected indoors, in a sealed laboratory room with the controlled artificial light source. As demonstrated in figure 12, the luminance levels in five critical points in the circuit are measured in two distinct conditions.

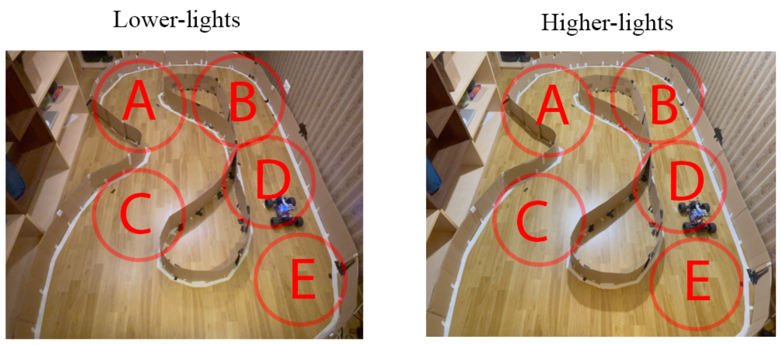


Figure 12. On the left are the standard lower-lights conditions used to collect the data set for training all models. In the right, the circuit is exposed to higher-lights unseen to the model, as no data was collected under this condition.

To measure lighting levels in the circuit, the light meter LM-3000 [26] was chosen as an in-depth article [27] comparing several light meters point to it as the one clear winner as the most accurate option. As demonstrated in table 7, different points of the circuit track had different light level measurements with levels over four times higher in the brighter conditions.

Position on the circuit	Lower-lights - L (lux)	Higher-lights - H (lux)
A	21	110
B	19	86
C	19	75
D	19	75
E	20	60
Average	19.6	81.2

Table 7. The four models tested in the experiment are tested in four conditions: lower-lights (L), lower-lights under attack (LA), higher-lights (H), and higher-lights under attack (HA).

The four models tested in this experiment were only trained using the images captured in lower-light conditions. However, two distinct training methods were applied to improve the model performance in the unseen higher-lights conditions, a standard training method and one adversarial defense training method.

4.2 Training

Four different models were trained and tested in the experiment, being two using a standard training method and the other two using an adversarial defense training method. As shown in table 8 the first model (M-TS) was trained with a smaller set of data, including data collected during 20 laps, the second model (M-TSA) used the same data as the first model. However, adversarial defense training was used instead. The third model (M-TL) was trained with a more extensive data set, including images recorded during 37 laps. The final model uses the same set of data as the third model. However, again an adversarial defense training method is applied.

Testing conditions						
Model	Laps	Images	L	LA	H	HA
M-TS	20	4750	19.6	19.6	81.2	81.2
M-TSA	20	9500	19.6	19.6	81.2	81.2
M-TL	37	8926	19.6	19.6	81.2	81.2
M-TLA	37	17852	19.6	19.6	81.2	81.2

Table 8. Four self-driving models were created in the experiment, being two trained with standard methods (M-TS and M-TL) and the other two trained with adversarial defense methods (M-TSA and M-TLA). The amount of data was increased in each model. However, M-TSA and M-TLA used artificially generated images along the images collected by the expert. The models are then tested in two different light conditions lower-lights (L) with an average luminance level of 19.6 lux and higher-lights (H) with average luminance levels of 81.2 lux. The models' performance was also measured when under exposure to an adversarial attack (LA and HA).

The first step for training a self-driving model involves having an expert demonstrate a driving policy, which is a set of labels associated with an image. In this study, two sessions were used to collect different amounts of data, figure 13 shows the steering angle labels of each one of the two sessions to collect data. Even though the second session has a more extensive set of data, the steering patterns are similar in both sessions, with most of the data containing samples of left curves with the negative values, followed by the values close to zero, which are related to when the car is driving straight. Finally, the minor portion of the data contains samples of right turns, as there was only one right curve in the circuit.

a) Smaller dataset b) Larger dataset

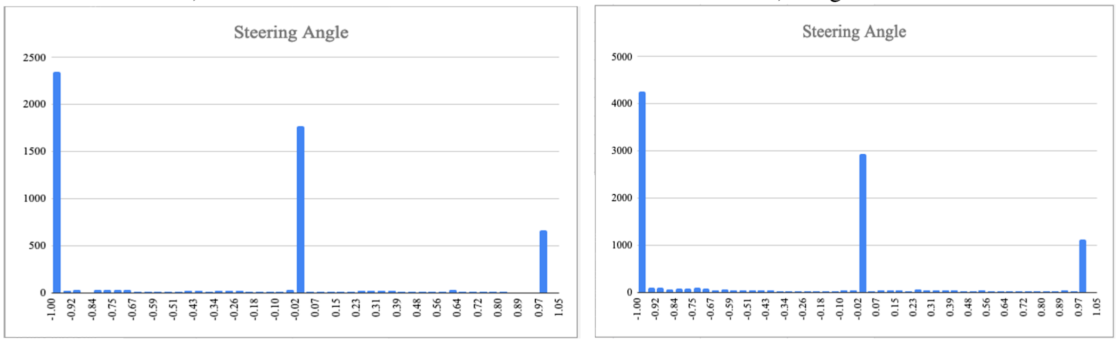


Figure 13. The steering angle labels are associated with the images in the dataset. On the left, there are approximately 5000 images in total, of which almost half of them have left steering information. Just a smaller fraction of the data contains right-turning instructions with positive values. The same driving pattern is displayed on the right side as the driving took place in the same circuit. However, more laps were driven the second time.

Besides the steering angle, the second label added to the image is the throttle used by the expert while collecting data during the two sessions. As is shown in figure 14 most of the data contain throttle values from 0.74 to 0.77, which means that the car was driven in somehow cautiously, most likely reducing the acceleration close to curves, which explain the smaller values scattered in the graph, another evidence of careful driving sessions is that the full-throttle values closer to 1.0 were never reached.

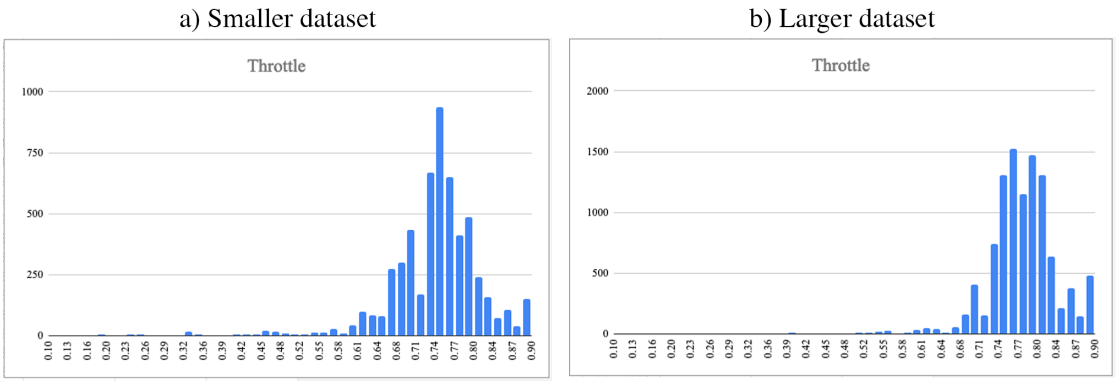


Figure 14. Both driving sessions have similar throttle patterns, oscillating from 0.74 to 0.80. However, the longer distances were traveled at higher speeds in the more extensive data set. It seems that the more significant number of laps also increased the expert's driving confidence.

Once the data is collected, the next step prior to training the models is to clean the data sets, removing any lousy driving examples, including crashes or near misses. Donkeycar library provides a tool to conveniently watch the records and easily select and remove undesired behavior. With cleaned data samples, all the necessary inputs are ready to generate the model testes in the experiment.

In total, four models were trained, as illustrated in figure 15 each model goes through a process of training across a number of epochs until no further improvement is obtained as a result of the loss function. The models trained using fewer data need more iterations to be trained, with both M-TS and M-TSA taking over 30 epochs to complete the dataset validation session. Interestingly, the larger dataset resulted in lesser epochs necessary to train the model. The model M-TL which was trained with the most significant number of images has the least amount of epochs iterations to achieve training completion.

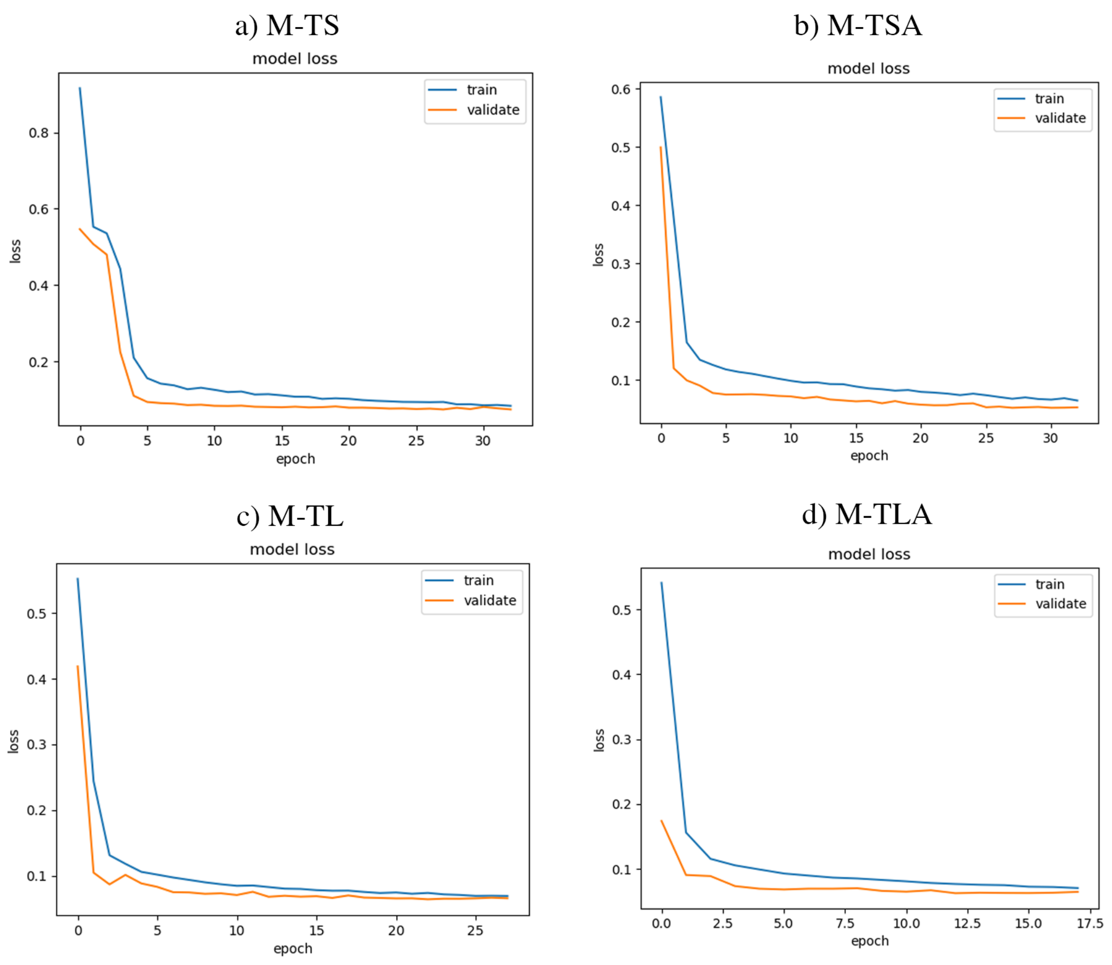


Figure 15. The four models tested presented similar learning patterns, with loss constantly falling, attaining low losses within a few epochs and then maintaining a slow, steady improvement across the remaining iterations. However, the model created with the adversarial training method (M-TLA) achieved lower loss levels earlier and finished the training with only 17 epochs.

After the models have been trained and generated, it is possible to visualize the performance of the model against the validation set, figure 16 displays the prediction of trained models, the accuracy is measured and compared to the human expert.

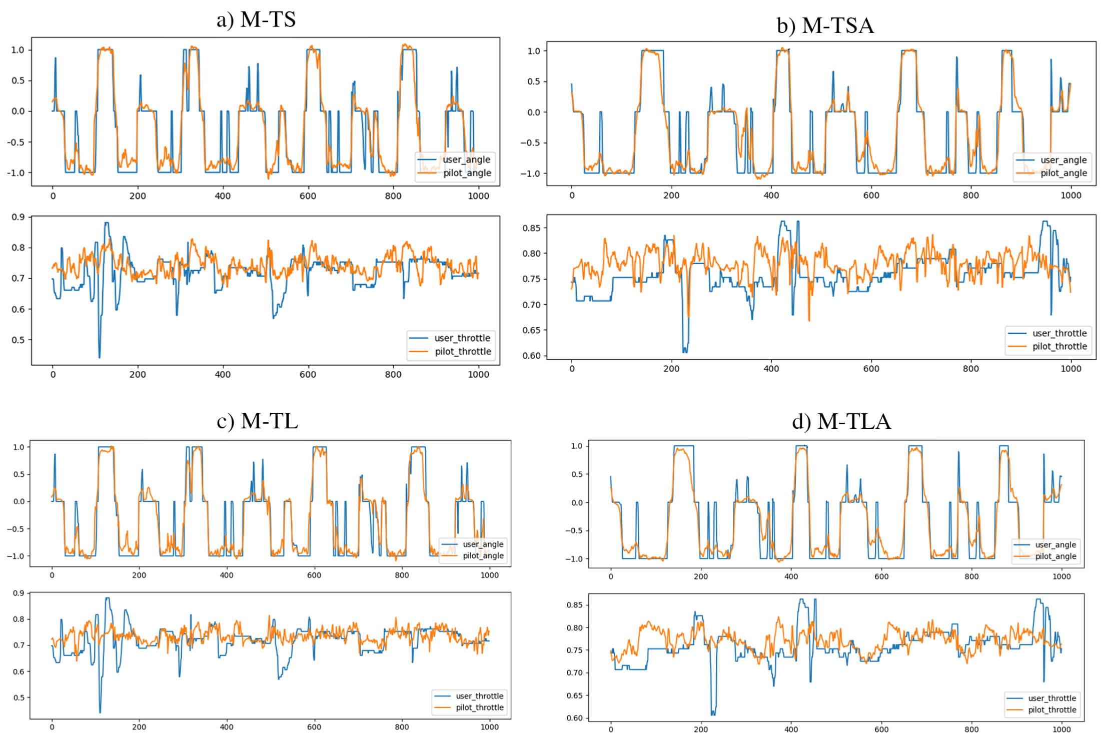


Figure 16. The steering angle and throttle prediction are compared to the human expert controls. From the above plots, it can be seen that all models present similar performance despite the different sizes of the dataset.

All models displayed similar performance when tested against the validation set. However, the test performed on the actual circuit will demonstrate the model's actual performance both in seen and unseen lighting conditions.

4.3 Design and Implementation

The four models were tested in four conditions in a real-world scaled setup to implement the experiment. The performance was evaluated, and the results are presented below. In each test stage, all models were deployed to run for a total of two laps under each of the four different conditions.

4.3.1 Testing M-TS

The testing 1 section refers to the tests performed in all conditions with a model trained with a standard training method using a smaller dataset. The results are described in table 9 which shows how the model displays a collision-free performance when running under the conditions on which it has been trained but seems to overfit for those training conditions and failed in all the other conditions.

M-TS	L	LA	H	HA
Total collisions	0	12	5	7

Table 9. There were no collisions registered in lower-light conditions (L). The most significant collisions happened when the model was under adversarial attack in lower-lights conditions (LA). The model failed to generalize to the unseen higher-lighting levels (H) and higher-lighting levels under attack (HA) colliding five and seven times respectively under these conditions.

As expected, the model had no collisions when running under lower lights, as it was the only condition to which the model had been exposed during the training. However, figure 17 shows that under the same low-lights condition, the model was susceptible to the adversarial attack, as when it was under attack, the model failed to generalize to the attack, accounting for 12 collisions.

Furthermore, the model seemed to have overfitted to the training conditions and failed to generalize to the unseen higher levels lighting conditions, colliding in total five times during two laps. Surprisingly, when in higher-lights under adversarial attack, the car had five collisions less than when it was in lower-lights with a total of seven collisions. Nevertheless, it demonstrates that different light conditions affect the model's performance, and adversarial attacks can influence its behavior.

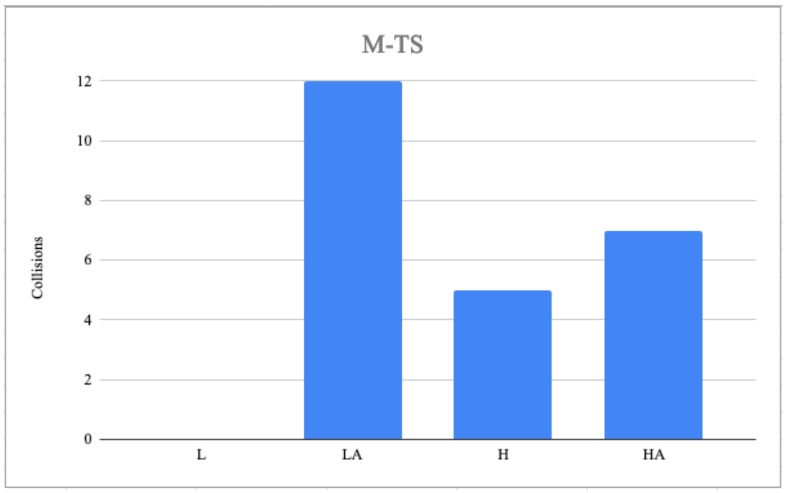


Figure 17. No collisions happen in lower-lighting levels (L), and five collisions were registered in higher lighting (H). The most significant collisions happened when the model was under adversarial attack in standard low lighting conditions (LA) with a total of 12 collisions, followed by when the mode was in higher lights under the adversarial attack (HA).

It was no great surprise having no collisions in the condition the model had been trained on, that was the expected behavior, but interestingly, two different collisions patterns were noticed in the remaining conditions. As shown in figure 18, when under the attack, the model had more left collisions, whereas when it was exposed to higher-lights without the attack, the left-collisions prevailed.

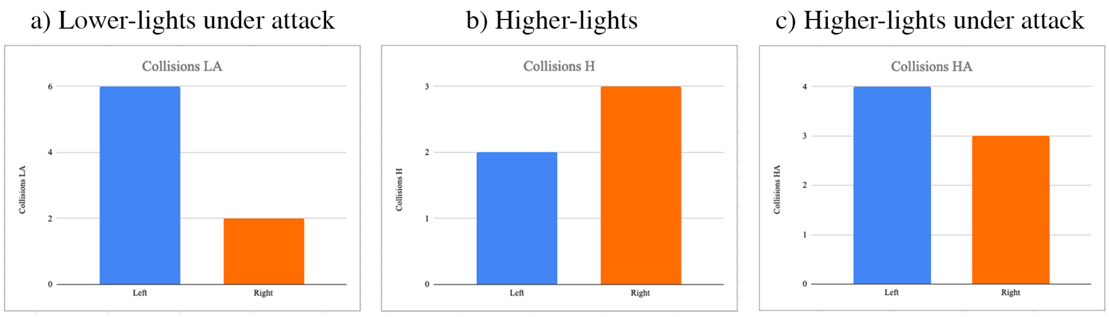


Figure 18. The model had no collisions in lower-lights, a) and b) registered a higher rate of left collisions when the model was under attack. On the other hand, in b) higher-lights without the attack, the right collisions outnumbered the left ones.

Exposing the model trained with a lower dataset to unseen light conditions was sufficient to fool it. Additionally, adding adversarial attacks when running the model made it misbehave, presenting a more significant number of collisions. Thus, indicating that the model was not skilled enough to generalize to unseen lighting conditions and the adversarial attack.

4.3.2 Testing M-TL

The second stage of the experimentation tested the hypotheses that a more extensive data sample could help the model improve its skills to generalize to both unseen lighting conditions and the adversarial attack. This model was trained with the same standard method as the previous one. However, double the amount of the data was used this time.

As demonstrated in table 10 the more extensive set of data had a minor positive impact on the performance of the model, still not in all conditions. The larger model did not generalize and had four collisions during two laps when exposed to unseen higher light conditions. The results of the condition under attack in higher-level lights again brought surprises as the model decreased its performance compared to the previous model colliding now nine times.

M-TL	L	LA	H	HA
Total collisions	0	10	4	9

Table 10. The number of collisions in each condition follows the same pattern as the previous model, with no collisions in lower-lights (L) and a higher number of collisions when under attack (LA, HA). Increasing the dataset was insufficient to prevent the model from colliding when performing in unseen higher-level conditions (H).

Although slightly reducing the overall number of collisions when compared to the first model, figure 19 reveals that the model with a larger dataset repeated the pattern followed by the previous model, with the most significant number of collisions taking place when the model was under adversarial attack and with the most significant number of collisions happen in lower lights conditions and under adversarial attack.

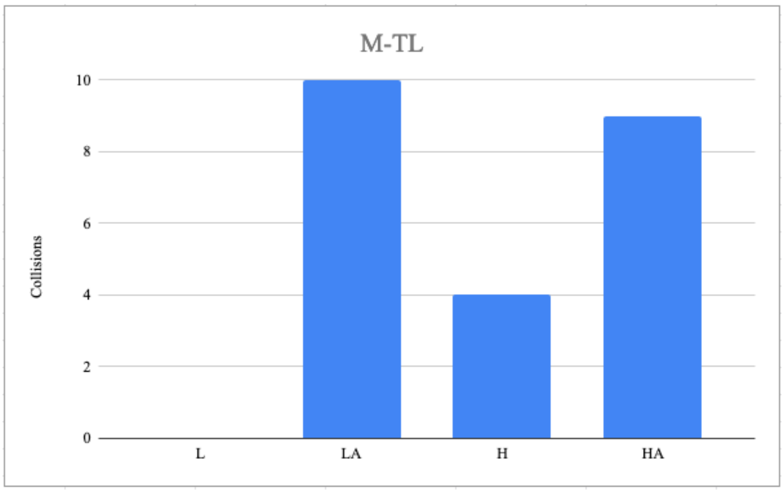


Figure 19. The model trained with a more extensive set of data still seems to overfit the training conditions maintaining a collision-free performance in lower-light conditions and colliding in unseen light conditions.

In low lights conditions, the model performed well, keeping it collision-free, which was the expected behavior considering it was the condition that the model had been exposed to during the training. However, the left and right collision rates differed from the previous test phase. As shown in figure 20, unlike the first model, a more significant number of right collisions was seen when in higher-lights under the adversarial attack. Without the attack, in higher lights, the model had one less right collision displaying the same rate of left and right collisions. Finally, the most significant change from the previous model was that when in lower-lights under attack, the model only accounts for left collisions, which could be a side effect of having more left turn samples in the dataset.

a) Lower-lights under attack   b) Higher-lights   c) Higher-lights under attack

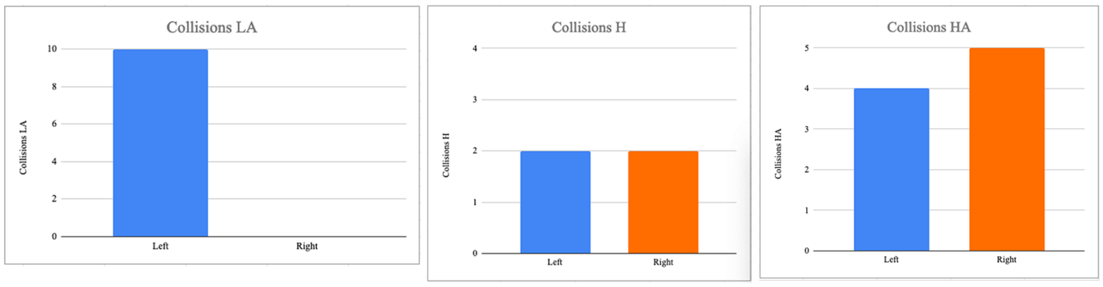


Figure 20. The rate of left and right collisions was different for each condition.

As only slightly lesser collisions were registered using the model trained with a more extensive data set, the improvement was not considered significant. Overall, the new model registered only one more minor collision than the previous model. In addition, the model's performance decreased when the model was tested in higher light conditions under attack. Therefore, using a larger dataset seemed not to have caused a substantial improvement in the model's overall performance across all conditions. The model still demonstrates the same issues as the previous model with a smaller dataset. It seems to overfit the training condition, failing to perform in unseen light conditions. Furthermore, it is equally vulnerable to adversarial attacks.

4.3.3 Testing M-TSA

For the third phase of the experiment, the possibility of improving the model's performance by applying an adversarial defense training method was tested. The model testing in this phase was trained using a combination of the smaller dataset used to train the first model used in testing 1 stage and an equal size batch of artificially generated adversarial images.

As it is shown in table 11, the adversarial training defense method helped the model to considerably improve its performance in unseen higher levels (H), enabling it to perform two laps without collisions, just like it did in lower lights (L), confirming the hypotheses that adversarial defense training methods can help to improve the model's generalizations skills.

M-TSA	L	LA	H	HA
Total collisions	0	2	0	5

Table 11. The model created using an adversarial defense method improved the overall performance. It ran for two laps collision-free in both the seen lower-lighting levels (L) and the unseen higher-lighting levels (H). However, the training method did not prevent the model performance from being affected by the adversarial attacks, reducing the number of collisions in lower-lights (LA) and higher-lights (HA) conditions.

Compared to the first model that was trained using the same dataset, the improved training method had a positive impact on the resilience against the adversarial attacks, reducing the number of collisions when under attack. As shown in figure 21, besides enabling the model to perform two laps without collision in unseen higher-level conditions, it also helped to reduce the collisions when the model was under attack in lower light (LA) and to reduce the rate of collision in higher lights under attack (HA).

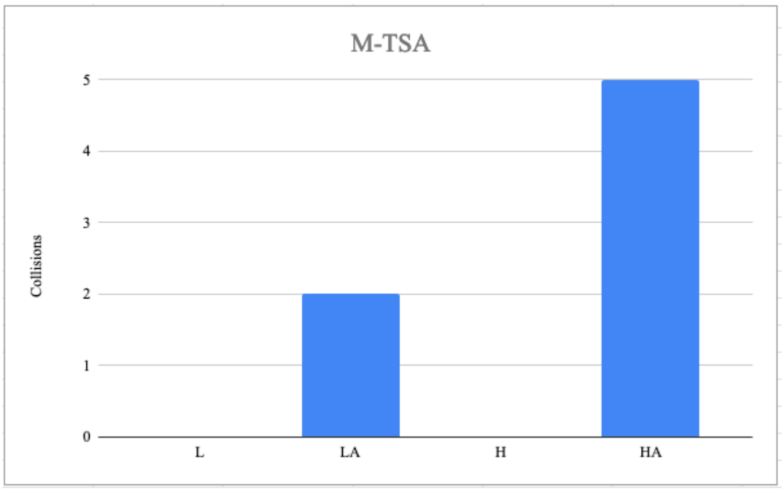


Figure 21. Training the model with an adversarial defense method helped generalize to unseen lighting conditions (H), enabling the model to perform two laps without collisions. Additionally, it has also improved the model's resilience against adversarial attacks reducing the rate of collisions when comparing the first model, which had used the same smaller dataset for training.

The efficiency of the adversarial training method against the adversarial attacks can be seen in figure 22. The number of collisions in lower-lights decreased from 12 with formal training to 2 collisions when the defense method was applied, whereas, in higher lighting, the number of collisions decreased from 7 collisions to 5 compared to the model using a standard training method.

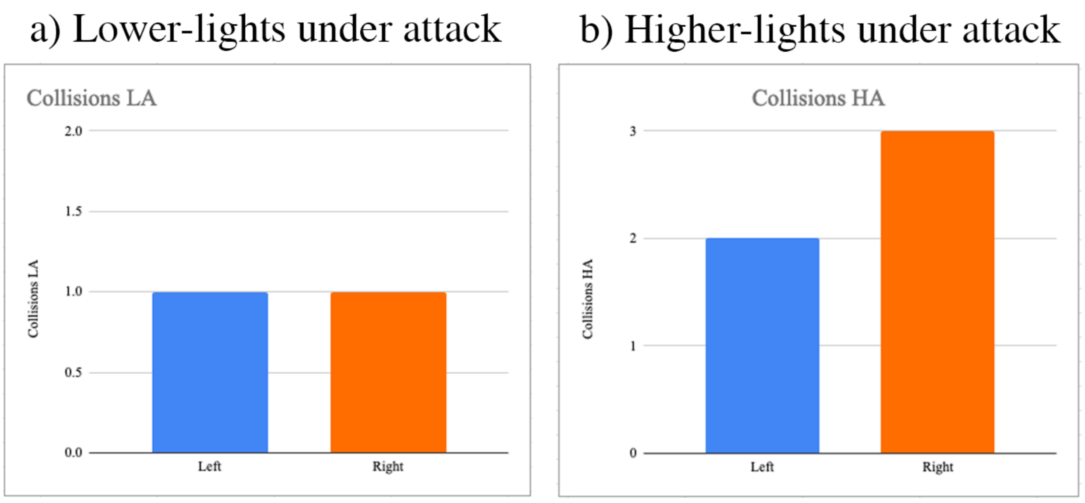


Figure 22. The distribution of left and right collisions was even in lower lighting levels, with one for each side, but the number of right collisions was slightly higher in high lighting levels.

Adding an adversarial defense mechanism to the training process improved the model's generalization skills to higher lighting levels. It also improved the model's performance when running under the adversarial attack, although not eliminating the misbehavior caused by the attacks.

4.3.4 Testing M-TLA

In the last testing stage, the same adversarial defense method used for creating the previous model was used to create a new model. However, it used the same larger dataset used in testing phase 2.

As revealed in table 12, adding the adversarial training method helped the model with a larger dataset to generalize to unseen lighting conditions, enabling the model to run two laps without collisions in the unseen higher lighting levels. However, the largest amount of training data did not improve when the model was under adversarial attacks. Despite the larger dataset, the model had one more collision in lower lighting levels than the previous model, which used the same training method but half of the number of samples. Whereas, in higher lighting conditions, when under attack, the model's performance trained with adversarial defense method using the larger dataset did not improve concerning the previous model, maintaining the rate of 5 collisions during two laps.

M-TLA	L	LA	H	HA
Total collisions	0	3	0	5

Table 12. Model M-TLA tested in stage 4 had no collisions registered in lower and higher lighting levels.

The adversarial training method efficiently improved the model generalization skills to unseen lighting conditions, but the more significant number of images used to train the model did not collaborate to increase the model's robustness against the adversarial attack. As is shown in figure 23 the model kept almost the same rate of collisions when under attack than the previous model, which had used less data for training.

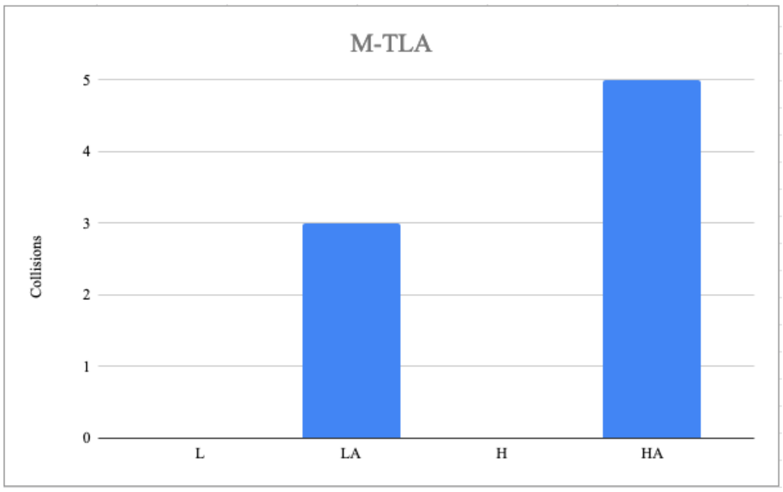


Figure 23. No collisions were registered both in lower-lighting and higher-lighting conditions. Whereas, when under attack, the model registered three collisions in lower-lights (L) and five collisions in higher-lighting conditions.

Figure 24 shows the left and right collisions rate for the model tested in phase 4 when under adversarial attack. While having one left and two right collisions in lower-lighting conditions, the model only registered left collisions in higher lighting level conditions, with a total of 5 collisions.

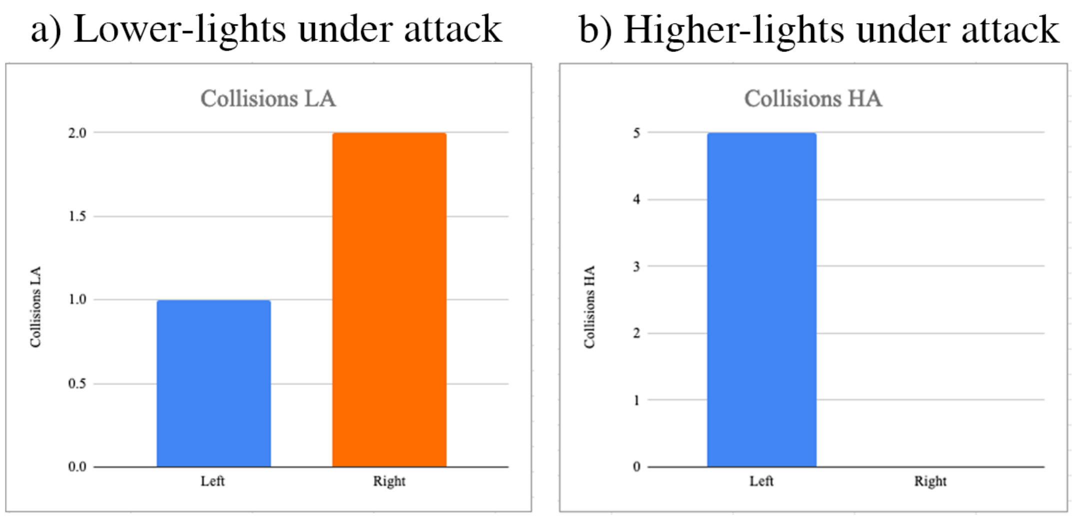


Figure 24. The larger dataset and the adversarial defense training method improved the model's performance while under attack while still having collisions. In lower-lights exposition (LA), one left and two right collisions were registered. Whereas, in higher-lighting exposure (HA), the model accounted five left collisions.

The last phase of testing, using a larger dataset to train a model using an adversarial defense method, maintained the performance in lower-lighting conditions and improved the performance in higher-light conditions, when compared to the models trained with the use of adversarial defense method, enabling the model to run two without collisions. Nevertheless, adding more data alone was not enough to improve the model when it was running under an adversarial attack.

4.4 Evaluation

Four models were tested, the first trained with standard training methods and a smaller dataset, the second model also was trained with standard training methods, but a more extensive dataset was used instead. The third and the fourth models used different adversarial defense training methods. Both methods also only differ from each other in size.

As highlighted in table 13, the results show that using adversarial defense training methods consistently helped to improve the models skill to generalize to unseen higher lighting conditions, as in both models in which such training approach was used, no collisions were registered in that condition, while also keeping a collision-free rate in the standard lower-lighting condition.

Model	Images	L	LA	H	HA
M-TS	4750	0	12	5	7
M-TL	8926	0	10	4	9
M-TSA	9500	0	2	0	5
M-TLA	17852	0	3	0	5

Table 13. The adversarial defense training methods used with the models M-TSA and M-TLA consistently helped the model improve generalization skills to unseen higher lighting levels. On the other hand, the models M-TS and M-TL, which used a standard training method, seemed to have overfitted to the training conditions and exposing the models to higher lighting conditions resulted in collisions for both models.

When driving under lower lighting exposure, which is the condition in which all images used for training the models were collected, all models drove the circuit without colliding against the walls. However, in higher lighting levels, the model trained with standard methods, registered collisions, and collecting extra data did not significantly improve the model's performance.

Furthermore, adding more samples to the data set used for training the models that used the adversarial defense training method reduced the model's resilience against adversarial attacks, see in figure 25 the comparison of the sum of all collisions that happen in each testing phase concerning the amount of data used to train the models.

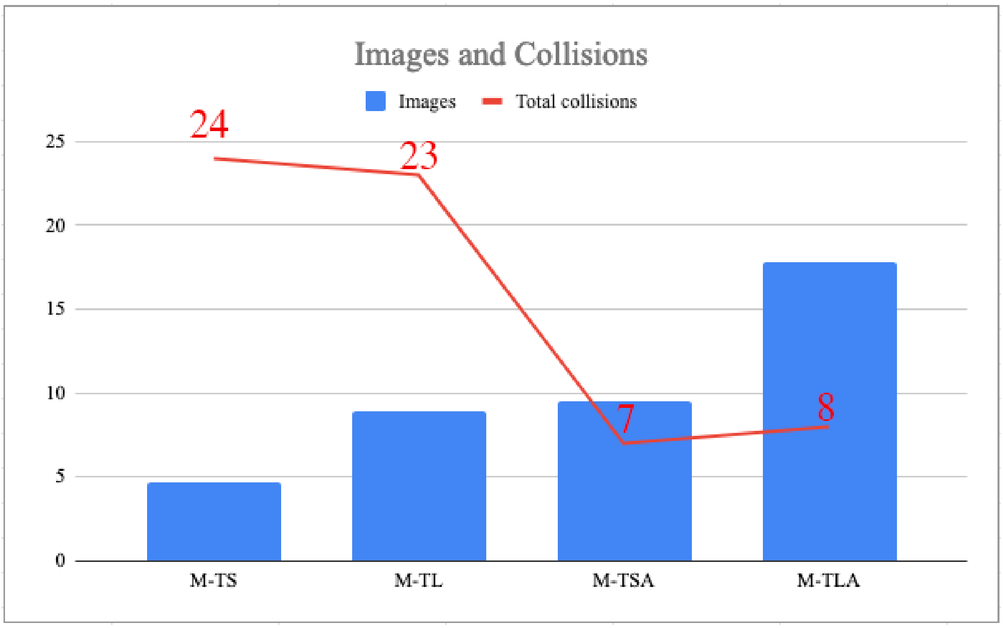


Figure 25. The expansion in the number of samples added to the model during the training only improved performance slightly from 24 collisions in M-TS to 23 collisions in M-TL. However, when using adversarial training methods, adding more samples to the dataset only worsened, as the sum of collisions increased from 7 to 8 when more data was added to the training process.

The strategy of expanding the training set with the inclusion of a batch of adversarial examples successfully overcame the issue of the model overfitting to the training conditions and failing to generalize to unseen higher-lighting level conditions. When an adversarial defense training method was used, the models performed two laps without collisions in lower and higher lighting conditions.

However, all models were vulnerable to the adversarial attacks, and surprisingly, even the models trained with the adversarial examples which were used to train itself were still susceptible to the attacks, and while the improved training method reduced the number of total collisions when under attack the models remained vulnerable in both lower and higher lighting levels.

5 Discussion

In this thesis, the neural networks model's lack of generalization skills to unseen lighting conditions was exposed and studied. Literature providing mechanisms to train self-driving models and generate batches of adversarial images was reviewed. Additionally, the use of adversarial attack methods was investigated in association with self-driving algorithms.

A hypothesis was presented proposing the possibility of using adversarial defense training methods for improving the model's generalization skills to unseen lighting levels. Specifically, the possibility of generating a batch of adversarial images and the standard images during the training process was explored.

A methodology was developed to test the generation of batches of adversarial images and to measure the impact of adding them to the training dataset. A real-world experimental setup was provided to test the performance of the adversarial defense training methods and conduct experiments in different conditions. The experiments described in the method were analyzed, and the different models' performances were compared.

Furthermore, this final section will summarize the findings, lessons learned and the limitations encountered while implementing the adversarial defense training methods to generate self-driving models.

5.1 Lessons Learned

The experiments with adversarial defense training methods indicate that it is possible to improve neural networks autonomous driving models' generalization skills to unseen lighting conditions by adding batches of artificially generated adversarial images to the training set. However, the analysis revealed that the improvement was only detected in limited conditions. The results also demonstrate that adding samples to the dataset alone was enough to improve the model performance modestly. However, models constructed with larger datasets did not stop the cars from colliding when facing unseen lighting conditions. On the other hand, the models generated from adversarial defense training methods could navigate two laps without collisions in lower and higher lighting conditions.

A possible advantage of using the adversarial training approach is that the method used to generate adversarial images could be used with different input values to generate a range of light conditions. Therefore, this approach would not require experts having to collect data for training the models as described in sections 2.3.2 across different lighting conditions, which would make the data collection process faster.

Besides the performance improvement in unseen lighting conditions, another advantage of the proposed method is that the models also become somehow more resistant to the adversarial attacks described in section 2.4.1. Despite the model being still not totally immune to the adversarial attack, it showed improvement, accounting for lesser collisions when applied adversarial training.

One disadvantage of the proposed training method is that it requires much more processing power to generate the adversarial images and train the new models with them than a normal training method.

One of the disadvantages of our method is the driver class that we use to extract internal state data during the test suite execution.

Another disadvantage of the approach taken in this project is that the attack perturbation levels were chosen manually. However, it could make it more efficient to develop mechanisms to automatically sense the lighting levels directly from the images provided in the input and generate some different levels of perturbations accordingly. Nevertheless, there is no simple way to gauge the luminance levels straight from the input image. Additionally, manually adding inappropriate perturbations levels could lead to undesired results, and the fine-tuning process could become costly.

The adversarial defense training method requires additional time for appropriate fine-tuning levels of perturbation to generate artificial adversarial images. Generating the batches and mixing them with the standard images is not a trivial task as the models must be loaded and integrated with the convolutional neural networks prior to the training sessions. In this experiment, scripts were developed to duplicate the data samples adding perturbations. However, still many manual inputs had to be decided to generate each batch.

The findings of this experiment are summarized as follow:

- A scaled real-world setup was constructed to test adversarial defense training methods with autonomous driving models.
- The adversarial defense training method uses artificially generated images to create models. Thus, replacing the need to collect data in various possibilities of lighting conditions.
- The model trained with adversarial defense methods increased generalization skills to unseen higher lighting conditions.
- Adversarial training methods improved only to a certain extent the model's performance while under the adversarial attack.
- The implementation of adversarial training method and selection of appropriate perturbations levels are rather complex tasks that may require extra time. Additionally, there are no optimal perturbation values that generate adversarial images adequate to all lighting conditions.

5.2 Limitations

Considerable limitations were encountered while experimenting with adversarial defense training methods and generating adversarial examples.

Firstly, the method was tested only on a single lighting condition change. Furthermore, the chosen conditions were simply training the baseline model in lower lighting conditions and testing in higher lighting conditions. If more levels of lighting exposure were tested, or if the training was done in higher lighting conditions and testing the generalization to lower-lighting conditions, perhaps more relevant insights could have been raised from the experiments.

Secondly, the perturbations levels used to generate adversarial samples were done manually. If such inputs were automatically generated, then the models could be possibly be tested in additional lighting conditions, which could have increased the visibility of the phenomena.

Thirdly, the dataset collected for the experiments could be considered small, and different results could have been found if an even greater set of data was tested in the experiment. Limiting the dataset to relatively small sample size was necessary to prevent performing issues in lower processing power devices such as Raspberry Pi.

Finally, some image augmentation methods such as cropping, blur, brightness, flipping, and rotation could have been applied in addition to the adversarial training methods to improve the model's performance further. If such techniques were further explored, perhaps, richer and more expressive results would have been found.

6 Conclusion and Future Work

This thesis aimed to implement a proof of concept adversarial defense training method that can be used to generate models able to generalize better to unseen lighting conditions. This goal was reached by showing that when exposed to unseen higher lighting conditions, the model trained with adversarial methods could perform just as good the levels demonstrated in the performance under the conditions used during training. This was not true for the models created using standard training methods.

The demonstrated adversarial training method has shown that time and resources can be saved while using this approach as it can reduce or altogether remove the need for expert demonstrations in different lighting conditions. Adversarial defense training methods allow machine learning engineers to deploy more robust self-driving models that generalize better to unseen lighting levels.

Adversarial training methods have also increased the performance of the models when under the adversarial attack. The results have shown that the number of collisions reduced considerably while still being affected by the attacks.

Future work is expected to use additional image augmentations to improve the model performance in unseen conditions further. Also, the generation of adversarial examples could be extended with a more diverse selection of attack techniques regarding selecting suitable adversarial attacks. Finally, a larger volume of samples might yield interesting results.

In summary, the thesis goal was achieved by implementing an adversarial defense training method and evaluating its performance in a scaled real-world setup. This should expressively reduce the engineering effort required to collect enough data to create high-performance autonomous driving models resilient to changing lighting conditions.

7 Acknowledgement

References

[1] T. Team, “Artificial intelligence autopilot,”

[2] T. Team, “A tragic loss,” 2016.

[3] T. Team, “Tesla vehicle safety report,” 2021.

[4] A. Tampuu, M. Semikin, N. Muhammad, D. Fishman, and T. Matiisen, “A survey of end-to-end driving: Architectures and training methods,” *CoRR*, vol. abs/2003.06404, 2020.

[5] “Darpa grand challenge,” 2016.

[6] D. Pomerleau, “ALVINN: an autonomous land vehicle in a neural network,” in *Advances in Neural Information Processing Systems 1, [NIPS Conference, Denver, Colorado, USA, 1988]* (D. S. Touretzky, ed.), pp. 305–313, Morgan Kaufmann, 1988.

[7] B. F. L. J. U. M. K. Z. Mariusz Bojarski, Ben Firner and D. D. Testa, “End-to-end deep learning for self-driving cars,” 2016.

[8] J. Z. Chang, “Training neural networks to pilot autonomous vehicles: Scaled self-driving car,” 2018.

[9] Q. Zhang and T. Du, “Self-driving scale car trained by deep reinforcement learning,” *CoRR*, vol. abs/1909.03467, 2019.

[10] Q. R. Eliot Cowley, Alma Jenks, “Convolutional neural network,” 2021.

[11] T. Agarwal, H. Arora, and J. Schneider, “Affordance-based reinforcement learning for urban driving,” *CoRR*, vol. abs/2101.05970, 2021.

[12] Q. R. Eliot Cowley, Alma Jenks, “What is a machine learning model?,” 2021.

[13] T. F. T. K. a. I. A. Yaqub Mahmoud, Yuichi Okuyama, “Optimizing deep-neural-network-driven autonomous race car using image scaling,” 2020.

[14] “An opensource diy self driving platform for small scale cars.,”

[15] “Build beneficial and privacy preserving ai,”

[16] G. A. Santiago and M. Favre, “Analysis and evaluation of recurrent neural networks in autonomous vehicles,” 2017.

[17] “Keras parts,” 2021.

[18] N. Piazzesi, M. Hong, and A. Ceccarelli, “Attack and fault injection in self-driving agents on the carla simulator - experience report,” in *Computer Safety, Reliability, and Security - 40th International Conference, SAFECOMP 2021, York, UK, September 8-10, 2021, Proceedings* (I. Habli, M. Sujan, and F. Bitsch, eds.), vol. 12852 of *Lecture Notes in Computer Science*, pp. 210–225, Springer, 2021.

[19] A. Rosebrock, “Mixing normal images and adversarial images when training cnns,” 2021.

[20] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.

[21] J. Zhang and C. Li, “Adversarial examples: Opportunities and challenges,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 31, no. 7, pp. 2578–2593, 2020.

[22] A. Rosebrock, “Image gradients with opencv (sobel and scharr),” 2021.

[23] J. Zhang, Y. Lou, J. Wang, K. Wu, K. Lu, and X. Jia, “Evaluating adversarial attacks on driving safety in vision-based autonomous vehicles,” *CoRR*, vol. abs/2108.02940, 2021.

[24] M. Liivak, “Sample-efficient online learning in a physical environment,” 2020.

[25] “Train an autopilot with keras,”

[26] L. I. GmbH, “Light meter lm-3000,”

[27] D. Maglia, “The best lux meter app on ios in 2021,” 2021.

Appendix

I. Glossary

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public