THE UNIVERSITY OF WESTERN AUSTRALIA

End to End Self-Driving using Convolutional Neural Networks

Self-Driving Autonomous Car Project

Vihanga Akash Thathsara Silva

Student Number: 21480143

Department of Electrical, Electronic and Computer Engineering

The University of Western Australia

Supervisor: Professor Thomas Bräunl

Submitted on the 25/05/2020

Word Count (Body): 6124

ABSTRACT

Biomimetics is solving human problems through observation and imitation of natural elements, models and systems. One of these systems is the human visual cortex, which processes the signals transmitted by the eyes and associates it with people, objects and structures that are familiar to that respective individual. While the exact method is unclear, it is theorised that the associations are made through the extraction of features within a scene, which build up in complexity, layer by layer, from colours and shapes to objects. This natural system serves as an inspiration for Convolutional Neural Networks (CNN), which also produce outputs through feature extraction, layer by layer, in increasing complexity.

This study aims to develop a robust and efficient model that uses neural networks to achieve full autonomy in a self-driving car. The primary input into the model will be visual data from cameras. The visual data is be processed using CNN and finally fused with secondary sensors to produce an output to the motor controllers and drive actuators. This model is optimised through supervised and reinforcement deep learning methods. Both of which also mimics methods used by humans to learn.

The primary outcome from this study is proof that CNN-based End to End Self-Driving method for autonomous cars is a feasible option with current technology. A secondary finding is that CNN-based model can be implemented on compact processing units that are low cost. This is critical to the mass adoption of the technique as it reduces the barrier for entry. Though the model is feasible in its current state there are still multiple improvements that can be made. Future research should focus on breaking the model into submodels focused on specific areas of autonomy problem, which could lead to higher model accuracy.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisor Professor Thomas Bräunl for his assistance and guidance throughout all stages of this research project. The time taken out his schedule is greatly appreciated.

I would also like to thank the members of the REV Project team, especially Thomas Drage and Craig Brogle for support throughout the design process and being a source of knowledge during troubleshooting.

LIST OF ACRONYMS

CNN	Convolutional Neural Networks
CV	Computer Vision
DL	Deep Learning
DNN	Deep Neural Networks
EV	Electric Vehicle
GPS	Global Positioning System
IMU	Inertial Measurement Unit
LIDAR	Light Detection and Ranging
MPH	Miles Per Hour
NN	Neural Networks
RADAR	Radio Detection and Ranging
REV	Renewable Energy Vehicle
ROS	Robot Operating System
SAE	Society of Automotive Engineers
SLAM	Simultaneous Localisation and Mapping
UWA	University of Western Australia

TABLE OF CONTENTS

TABLE OF FIGURES

1 INTRODUCTION

The autonomous driving vehicles is a fast-evolving field which is shaping and has the further potential to shape the way that people go about their day to day life. This can affect everything from how large mining companies collect and transport resources to the way in which the general population pick up groceries. Therefore, this research project has focused on testing and improving ways in which autonomous vehicles operate through the use of advances in machine learning such as Convolutional Neural Networks (CNN) to implement End to End Self-Driving. The following report will outline the research that has already been conducted in the field, how this research project was conducted and the results that came out of it.

1.1 Background

The Renewable Energy Vehicle (REV) team originally converted University of Western Australia's (UWA) Motorsport Team's Formula SAE car over to complete Electric Vehicle (EV) starting in 2010. This was to achieve the goal of promoting sustainable energy for the future. Over the years the Formula SAE project has shifted with current advances in technology, into a platform for conducting research into self-driving autonomous vehicles. This has transformed the Formula SAE car into drive-by-wire with ability to control steering, brake and motor speed outputs, which allow for full computerised operation of the vehicle this is then combined with a number of sensors to allow for inputs from the environment to aid with self- driving algorithms. The fitted sensors include dual front-facing cameras, Light Detection and Ranging (LIDAR), Global Positioning System (GPS), Inertial Measurement Unit (IMU)and wheel speed sensors, which is all currently connected to ROS running on a Jetson embedded computing board from NVIDIA.

Presently the Formula SAE is capable of autonomous operations including waypoint driving [1], LIDAR based cone detection/navigation and LIDAR based obstacle detection. There are also a number of systems that are being developed that will utilise visual technics to achieve SLAM, however, these systems are not yet complete and/or integrated into the car. All these systems have their inherent advantages, as well as disadvantages.

In 2020, the REV Project introduced the French developed EasyMile bus into the list of vehicles used for autonomous research. This bus comes with ready equipped sensors, motors and low-level driving systems. Therefore, it is a great platform to build on to of for autonomous driving systems. Currently, the bus is in the preparation stage of the build. The bus will be the primary development platform for the team's self-driving efforts.

In 1998, a paper was released on that outlined how gradient descent technics can be used to train Neural Networks (NN) to recognise handwritten letters. In the process of completing this goal one of the very first convolutional neural networks, LeNet5 [2] was created. This proved that neural networks using convolutional layers could be used effectively in visual recognition problems. While other forms of image processing have been researched and implemented on the car, the use of CNN and machine learning makes the process of creating and improving the accuracy of recognition much simpler as the only limitation is processing power and quality of training data.

1.2 Problem Statement

To develop a more robust form of object detection, it is desirable to have multiple systems and arrays of sensors that overlap each other. This can, in turn, provide a method of overcoming the limitations of each individual system and create a more robust, accurate and feature-rich solution. One type of system that can be used to overcome some of the shortcomings of a LiDAR-based for object detection and SLAM is using visual camera-based technics. This will make use of a camera streams from multiple camera sensors to map out the surrounding environment.

There are multiple methods of implementing visual Simultaneous Localisation and Mapping (SLAM), each with its own features, and limitations, however, as the higher-level goal of a self-driving system is to have the ability to sense and navigate its environment, Neural Networks could be used to achieve the system that is capable of processing input data all the way to through producing vehicle controls. This would negate the need for another individual system to process navigation and vehicle control. Therefore, the intention for this project is to research, design and build a NN, to carry out localisation and mapping whilst also carrying out pathfinding technics to navigate around in an environment.

2 LITERATURE REVIEW

This project combines and builds upon the work of several previously conducted research and this section reviews the literature from that research.

2.1 SLAM

SLAM is a process of mapping and continuously updating the map of the surrounding environment while maintaining track of the machine mapping the environment [3]. Through SLAM, it is possible for the system to operate autonomously, irrespective of the environment and surroundings it is placed, which make SLAM a critical for a system to achieve true autonomous. To generate effective mapping and localisation, equipment for sensing the environment is required, this can be from sensors such as LiDAR, Radio Detection and Ranging (RADAR), Stereo Cameras, etc. There are advantages, and disadvantages to each of these types of sensors, however, the most widely used method for SLAM is through LiDAR. Through the use of sensors and previous estimations made of the environment, it is possible to use mathematical methods such as Kalman Filter (illustrated in Figure 1) to make a forecast of the surrounding environment.

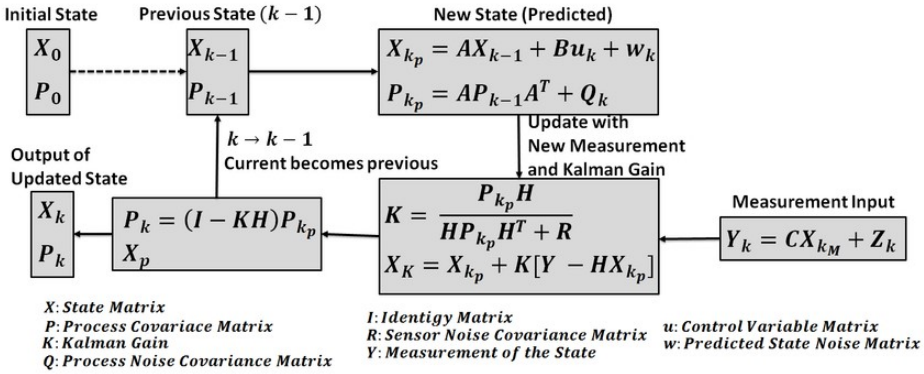

Figure 1: Graphical Layout of the Kalman Filter [4]

SLAM is applied in numerous applications and is a powerful tool for attaining situational awareness for autonomous machines. In Robot Operating System (ROS) SLAM is used in the *gmapping* package for making 2D occupancy maps [5]. Consequently, it has been made simple to preform SLAM by means of LiDAR in a 2D space.

SLAM, nevertheless, has its own intrinsic limitations, that make the system a computationally exhaustive when working with large spaces [6]. This is especially true when dealing with 3D spaces, and the computational cost required to process mapping for such spaces is not wanted as it would not be practical to carry large quantities of equipment required for such calculations in smaller form factor robots.

2.2 Convolutional Neural Networks

CNN is a form of Deep Neural Networks (DNN) that is mainly used for analysing visual data [7]. Unlike the usual full connected to neural networks, CNN makes use of pattern recognition to analyse data, starting with small patterns and using increasing convolutional layers, work to analyse bigger more complex pattern. Connections can be compared to biological processes with respect to how CNN processes imagery [8], it is theorised that human visual cortex uses a similar mechanism to creating relationships between shapes and objects that humans recognise. There is often little arithmetic required to prepare images and/or data to feed into a CNN, as during the training of the network, the network learns the filters essential to analysing the image. The difficulty of the patterns that can be recognised by a CNN is often hinged on the number of layers; the typical design of a CNN can be seen in Figure 2.

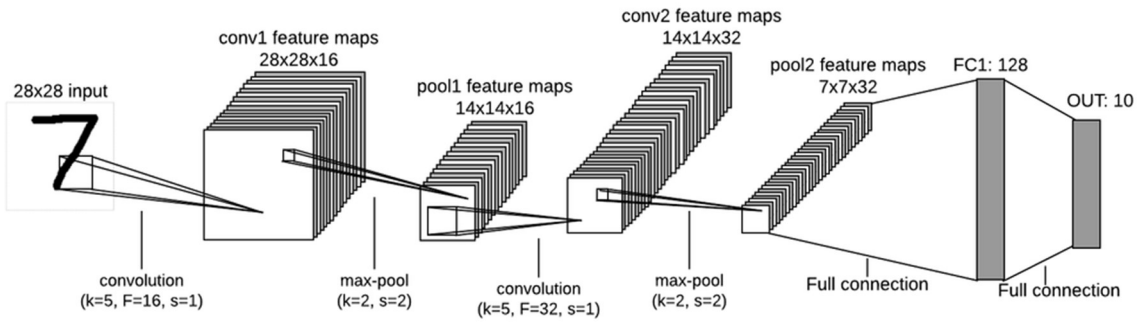

Figure 2: Typical Layout of CNN [9]

There is a sum of applications for CNN, though it is mainly used for image and video analysis, it has of recent been used widely in natural language recognition [10]. It is a commanding tool for autonomy as in contrast with old forms of handcrafting algorithms to automate actions, CNN is can be trained to on formatted sets of data and when applied correctly is capable of making more generalised decisions.

2.3 Computer Vision

There are two key categories of Computer Vision (CV) being researched and applied, these are old arithmetic based CV and neural network-based CV. The end goal of computer vision is to extract data from the setting into structured data. This data can then be used for decision making. The arithmetic approach uses filters and geometric relations to estimate and predicted the structure of the setting, however, has the arithmetic approach requires the handcrafting of filters and processes, it is very

difficult to extract from more compound patterns and information from a scene [11]. It has still advanced to a stage where facial detection, as well as other forms of object detection, is possible [12].

Neural network-based CV has established a reputation for being much simpler to develop. No handcrafting of processes is necessary, as the network learns to recognise data from the scene through structured and labelled datasets. This is where the limits are in neural network-based CV, as a model is only as good as the quality and accuracy of the dataset used for training. Therefore, it is critical to ensure that training datasets are accurate and covers many conditions. Having a dataset that extends a large variety of conditions ensures that the model learns general features when extracting data from a scene, as opposed to overfitting specific circumstances [13].

2.4 End to End Self-Driving

End to End Self-Driving is when inputs, such as camera streams and vehicle parameters are fed into a neural network, where all of the calculating (such as object detection and pathfinding) is done within the network so that the final output is control directions to steering, accelerator and brakes [14]. As image processing is required, most of the layers of the network used are CNN. Unlike some NN models, inputs are usually fed into two distinct networks that are fused together using densely connected layers, this is shown in Figure 3. The makeup of the second network is often vastly different from the image processing network as the data that is input is usually independent of the other inputs.

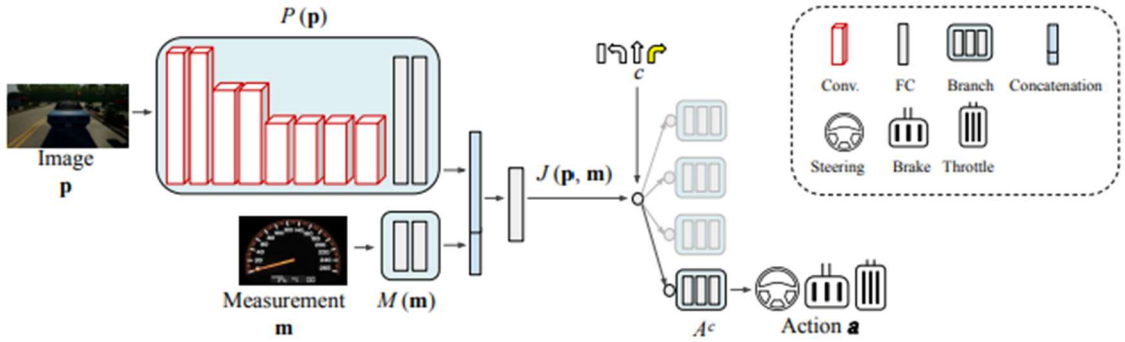

Figure 3: End to End Self Driving Example Layout [15]

While the technique from a design point of view is simple, the results are extremely depended on the training datasets. Just like all other applications of NN, wide-ranging datasets that cover a large range of situations is essential to training a network that is generalised and is not overfitted to a specific condition.

2.5 CAN-Bus

CAN-Bus is a communication bus designed in the 1980s for the purpose of communicating with controllers in a localised network [16]. There are a number of variations of the CAN-Bus that has been developed over the years. This change in speed capability and the amount of information that can be carried during one transmission. It is ideal for communicating diagnostics messages between various systems within a vehicle [17]. The network priories lower IDs over the higher IDs, this can be utilised in situations where a certain device needs priority over another. This, therefore, means the CAN-Bus is an ideal form of inter controller communication within a vehicle.

3 DESIGN PROCESS

This section of the report outlines the course of action to be taken when executing the project's design process.

3.1 Requirements

The final system needs to be able to drive a vehicle autonomously, using only the sensors that are fitted to the car without an external assistant. The term-end to end entails that the system is capable of completely controlling the car with only the sensory inputs from the vehicle and must be self-contained within the vehicle. This system also needs to primarily rely on machine learning technics to achieve this. CNN is to be the primary method of processing inputs. This system should primarily rely on cameras and lidar to drive the wheel. The outputs from the system are steering angle, throttle and brake. This form the requirements for the design

3.2 Constraints

In order to design an effective solution, the constraints for the solution first needs to be identified. This falls under multiple categories which are outlined below.

3.2.1 Safety

The safe operation of the system is critical. The system is ultimately being designed to operate on a vehicle driven on the road, where there could be objects, pedestrians and other vehicles. This gives a large amount of importance on how the system can operate safely in ordered to not cause to damage to others. Therefore, is important that these factors are taken into consideration when designing the system.

3.2.2 Hardware

The system needs to be able to run on a computing unit that is physically small enough to be contained within a vehicle. This limits the amount of processing power available to run the system. This is further constrained as this computing unit also needs to be able to run other aspects of the car responsible for other various functions. As the function of the system depends heavily on the amount of computing power available, this is the largest constraints to manage.

3.2.3 Sensors

The system is limited to using only the sensors available of the car. As the vehicle is required to work independently of its surroundings, there are limits to the sensors that are available to the system to operate the vehicle. Therefore, the system needs to be able to operate completely on the sensors available on the vehicle.

3.2.4 External

All large constraint on the progress of the design was the COVID-19 pandemic of 2020. The pandemic hindered the work carried out on the second half of the project. The social distancing measures that need to be taken to ensure the safety of the general public limited the type of work that could be conducted at the lab. This, in turn, made it difficult to collect data to use for training the end to end driving model. Therefore, the simulation was the only viable option to train and drive the model.

3.3 Design Tools

A number of design tools were used throughout the design process to prototype and finalise the model. These tools are outlined in detail below.

3.3.1 TensorFlow

TensorFlow is an open-source machine learning framework that provides features to process data from end-to-end [18]. It is a framework that includes tools from input pipelines to process data, all the way to large scale deployment of machine learning models. TensorFlow makes it simple to build, test and evaluate models while creating a layer of abstraction from the low-level arithmetic. The low-level calculations are done using C++ and CUDA to ensure that model propagation is fast. Therefore, TensorFlow is an ideal framework for machine learning.

TensorFlow was the choice for the framework for this design as it is robust, well documented and plenty of community backing is available for troubleshooting issues. The Keras API available in TensorFlow simplifies the process of building networks and streamlines making improvements. This is crucial during research as large amounts of adjustments need to be made before finalising a design. The Data libraries available in TensorFlow makes it simple to input and pre-process data to feed into a neural network. Finally, CUDA during runtime increase the speed at which models could be fit and predicted, which reduces a large portion of the time spent when evaluating model performance. These are the reasons why TensorFlow proved to be a crucial tool in the design process of the system.

3.3.2 Udacity Simulator

The Udacity simulator was built for teaching purposes, however, proves to be a low barrier for entry simulator when testing out self-driving systems [19]. There are very basic sensors available as inputs for a system, however, the main inputs into the system, such as cameras and vehicle speed/control measurements are available for use. This is, therefore, a simple starting point to get a basic model designed and optimised. For these reasons the Udacity Simulator was the starting point for developing a model for self-driving and proved to vital in optimising the image processing portion of the model.

3.3.3 Carla Simulator

CARLA is an open-source simulator, that is built specifically for the purpose of aiding the design and development of driving systems [20]. CARLA is the choice of the simulator for the entire REV Project as it is a robust and flexible simulator that can be configured to suit the needs of the team. CARLA as the capability to simulate a large array of sensor, all the way from simple vehicle speed sensors, up to LIDAR and GPS. It is also capable of simulating traffic, both vehicles and pedestrians, whilst being able to communicate with ROS. These are all critical areas of interest when developing an autonomous vehicle, are the inclusion of all of these features in a single simulator makes it a very appealing solution. Therefore, CARLA was used extensively at all stages of the design process.

3.3.4 SocketCAN

SocketCAN is a set of libraries that are included with ubuntu and a number of other distributions. SocketCAN allows for simple interfacing with CAN-Bus communications, and the available features allow for reverse engineering of CAN messages. This tool proved to be vital when attempting to

interface with CAN network on the autonomous vehicle. This allowed for diagnostics data to be read and taken into considerations when operating the vehicle.

3.4 Agile Project Management

Agile project management is a form of project management that takes an iterative approach to deliver a design/solution [21]. Agile split up the project into multiple smaller batches of work focused on delivering a stage of the project. These stages (sprints) then build up to form the whole project. When all sprints are completed correctly the goal of the project will also be achieved.

3.5 Cone Detection

Cone detection was preliminary work that was conducted to get understanding for the way in which neural networks were trained, evaluated and improved. In order to conduct the work, the agile process of management was applied. Figure 4 outlines the work conducted each week (sprint) to produce the final cone detection system.

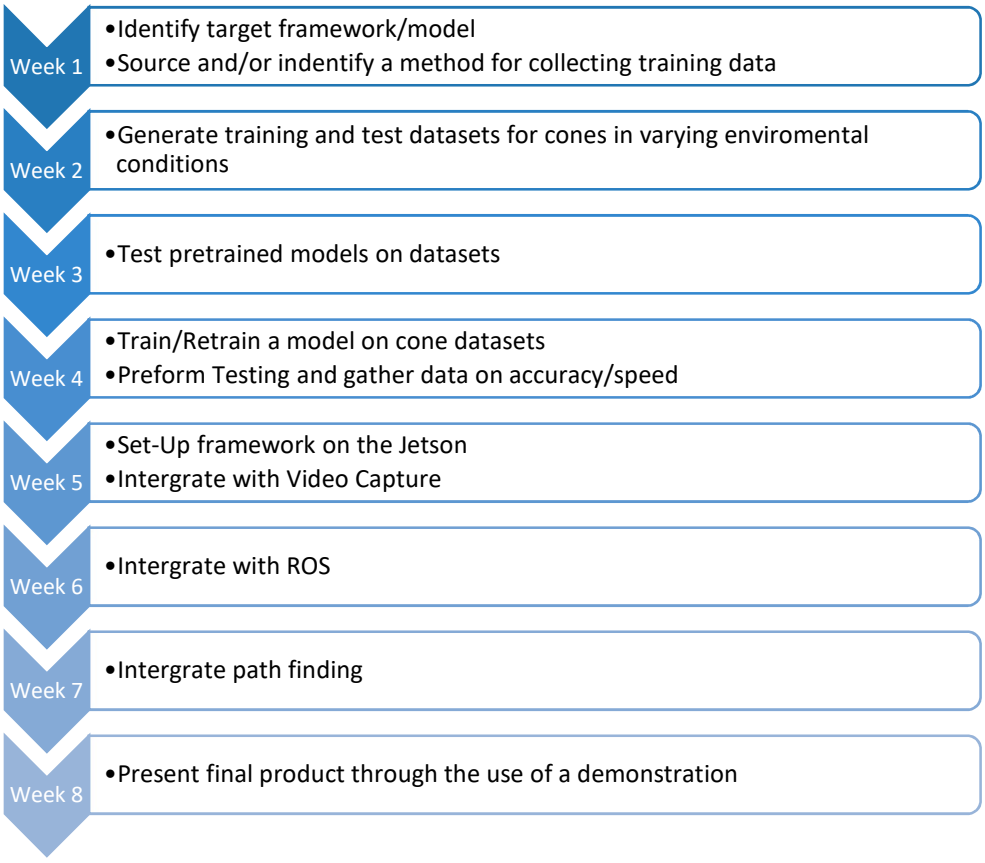

Figure 4: Sprint Project for Cone Detection

The system made use of an image detection system known as YOLO [22]. The exact model utilised is the YOLOv3-Tiny variant, as this offered a compromise between speed and accuracy. The model was trained on a dataset consisting of cones on both grass and road. These datasets were formed using video footage of the previous cone driving demonstrations of the Formulae SAE vehicle. The training was conducted with varying parameters to get an understanding of the effect each had on the outcome. An Nvidia GTX 1060 6GB edition graphics processor was used training the dataset.

The final trained system produced satisfactory results and provided valuable insight into how image processing is conducted on neural networks. The results of the YOLO cone detection system is available in Appendix 1 Figure 16 and the training data is illustrated in Figure 17.

3.6 End to End Self Driving

The End to End Self Driving system requires a large amount of work to be done over a year-long period to achieve the outcomes of the project. An agile method of project management was also adopted to carry out the work on the project. The sprints broke up the project into individual manageable sections. The agile structure use is illustrated in Figure 5.

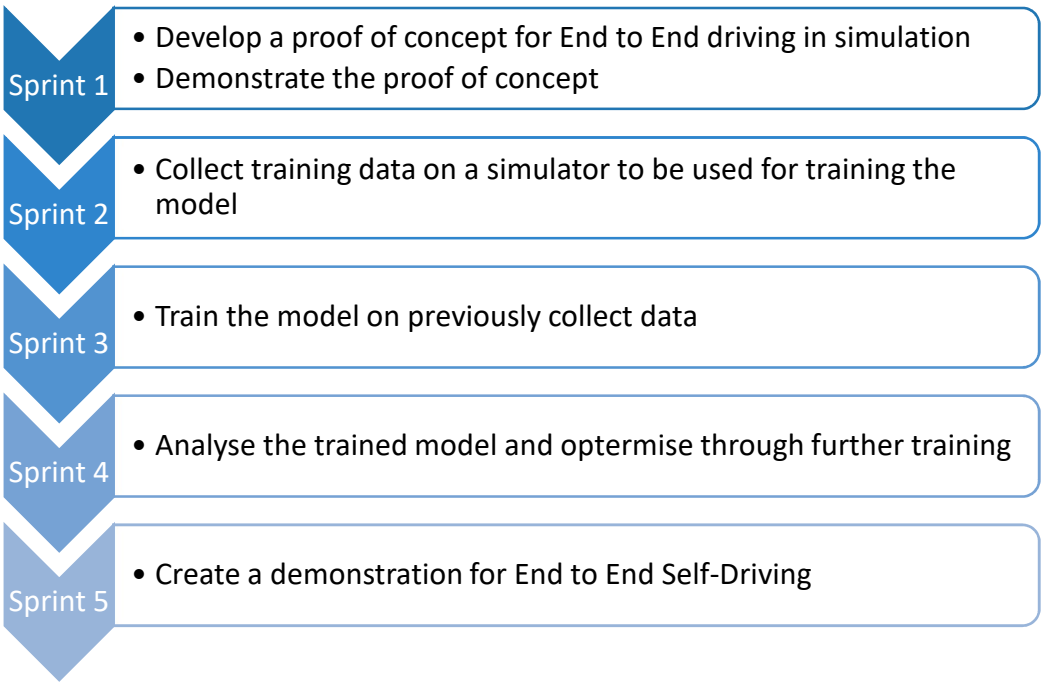

Figure 5: End to End Self-Driving Sprint Project

3.6.1 Proof of Concept

A proof of concept was first developed using a limited number of sensors to assess the feasibility of the larger model. The sensors available for the proof of concept was limited to one camera, speed, steering angle, throttle and brake. The Nvidia Self Driving model was used and adapted to create the initial model. The model was constructed using the TensorFlow framework. The Udacity Simulator was used for the data collection as well as the final driving process. This simulators simplicity made it very simple to test, train and drive models within a very short frame of time. Adaptations to the model could be assessed with very little effort. The proof of concept was successful and therefore, the model that was developed from the this was used as the basis for further development to transform into a more robust and capable model.

3.6.2 Data Collection

Due to the COVID-19 outbreak, the data collection was limited to CARLA simulator. In order to ensure a generalised model was produced it is important that all of the data collected represents a wide range

of conditions and scenarios. In order to get a variety of scenery, a number of different maps were used, these include Town01, Town02, and Town04. These maps come shipped with base CARLA simulator.

There are also a number of other factors within a map that can dictate the variation in a dataset. These factors include the traffic, weather, lighting and time of day. Traffic can be a very large factor to train for as this can often be unpredictable with lots of variation, so for this design, it was completely excluded. Weather conditions can cause reflections and hinder some sensors, therefore the dataset included both sunny and rainy weather situations. Finally, the light and time of day were also varied as the availability of light as well as the angle of light plays a large role in varying input data for a given scene.

3.6.3 Training

The training was conducted on the dataset collected on CARLA simulator. The training was initially conducted on a simple model that only input camera and speed information. This proved to be quite quick to achieve 16 frames per second during training. However, there was one major issue that arose during the training which involved excessive system memory usage. This came about from the caching function build into the TensorFlow input pipeline, which attempted to cache as much of the dataset as possible before training. This eventually led to all available system memory being used and caused the process to crash. The memory used over time before and after the fix was applied is shown in Figure 6.

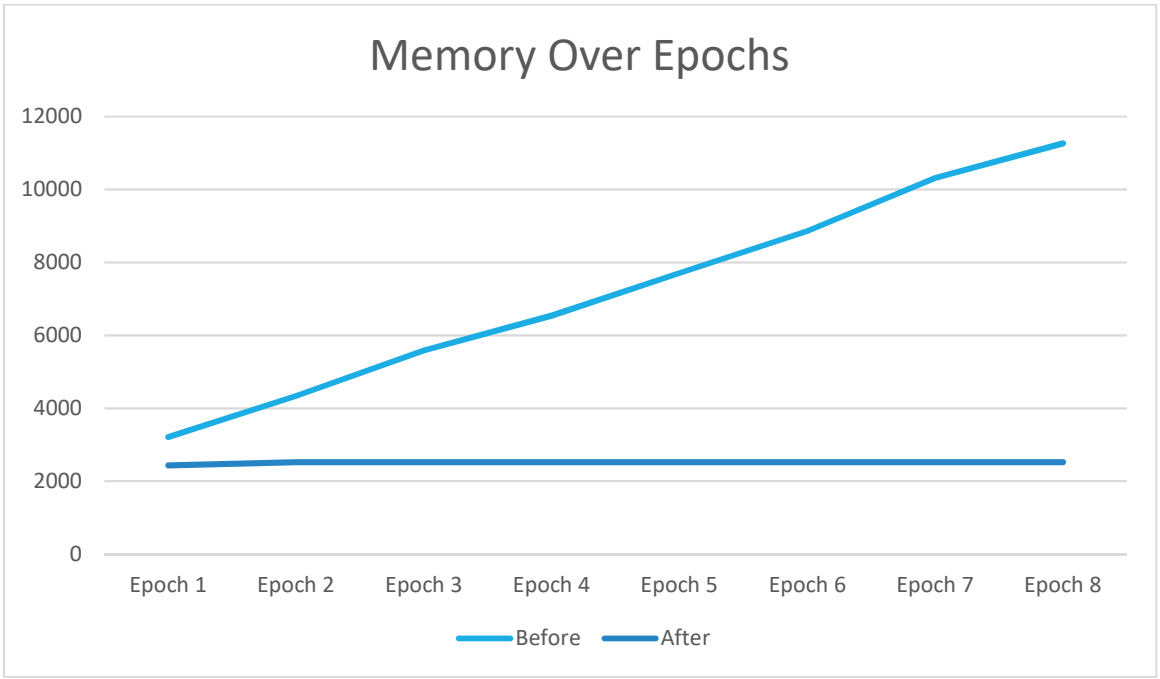

Figure 6: Memory usage before and after cache fix over epochs

3.6.4 Improvement

Once the first model was trained, further improvements were made to add more features to the model. The main additions include using Depth Range sensor data for creating a depth input into the model. This allowed the model to make predictions based on two sources of data for localisations and situational awareness. Adjustments to the layers were also made in order to optimise the performance of the model as a whole. The car measurements portion of the model did not require a high number of

connections, so these were reduced to improve speed. These changes made a major difference in the accuracy and performance of the model as a whole.

3.6.5 Driving

The driving of the model was the simplest section of the project. Driving was simple as it did not require a large amount of code to achieve. The driving script was mainly responsible for creating an input pipeline for the model. An illustration of the is pipeline is shown in Figure 7.

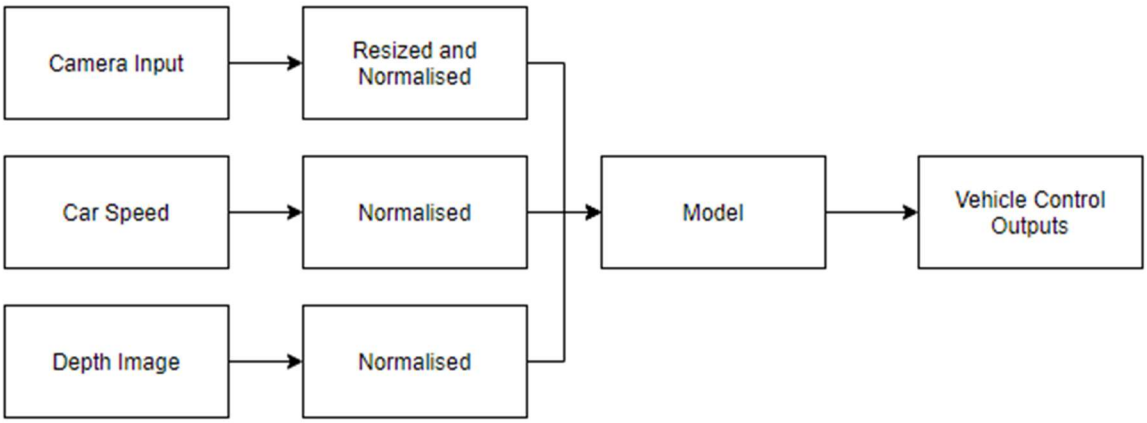

Figure 7: Input pipeline for driving

3.7 ηUWAy Preparation

The EasyMile bus that the REV Project team acquired needed to be prepared so that all of the previous and current autonomous driving systems can be implemented on it. These are outlined in this section.

3.7.1 CAN-Bus

CAN-Bus protocol is used extensively throughout the bus to transfer messages both control and diagnostics. Therefore, it was of great importance that a system for interpreting these messages was devised. The hardware component of the system was very simple to work with as all of the hardware required to interface with the CAN-Bus is already installed. The initial task was to determine the speed at which the CAN connection operated. This was determined to be 500kb/s after some trial and error. Using Ubuntu's built-in SocketCAN [23] libraries an interface with the CAN-Bus was established.

Next step was to build a ROS node to publish diagnostics messages over the network. This was aided through the use of already available nodes that could communicate with CAN-Bus using SocketCAN. Using the broadcasted message and the CANopen protocol to the messages could be decoded to gather the required information to be published to the network. This was primarily from the Safety PLC and gave important information to the state of the vehicle.

3.8 Evaluating the Design

In order to achieve a successful design, it is important to have methods for evaluating what success is in design. The aim of end to end driving is to achieve complete autonomy, therefore the primary way in which this can be measured is to analysis the accuracy of driving to what we expect from a competent human. The datasets that were collected can be split into two batches, one used purely for training and

the other can be used for evaluating the results of the model. This can then be compared to evaluate the final performance of the model. It is also important that driving examples of the model is also visually analysed to identify areas of interest that could be improved through the attention sensors, and or technics.

4 FINAL DESIGN

Once the design process was complete a final design was the result. This final design will be outlined in this section.

4.1 Inputs

There are three main inputs into the final design, these are Camera Image, Depth Image and Car Speed. This is the only inputs used by the model to make predictions for vehicle control. The camera image has a resolution of 400 by 400 and uses the 3 channels red, green and blue. A visual representation of this is shown in Appendix B. Whilst the depth image is an input resolution of 360 by 180 and uses only 1 channel which represents the distance of the object within the pixel. This is not in the traditional point cloud form that LiDAR data is usually represented in, however, point clouds can be converted into this form easily. Using a format similar to a greyscale image make it possible to use convolutional neural networks to process the depth data. Finally, the car speed input is simply a single floating-point value that represents the speed of the vehicle in MPH. All the inputs are normalised to a range between 0 and 1, for the images 1 represents 255 for a given pixel. And for speed 1 represents 30 MPH.

4.2 Structure

The final structure for the End to End Self Driving system is illustrated in Figure 8 below.

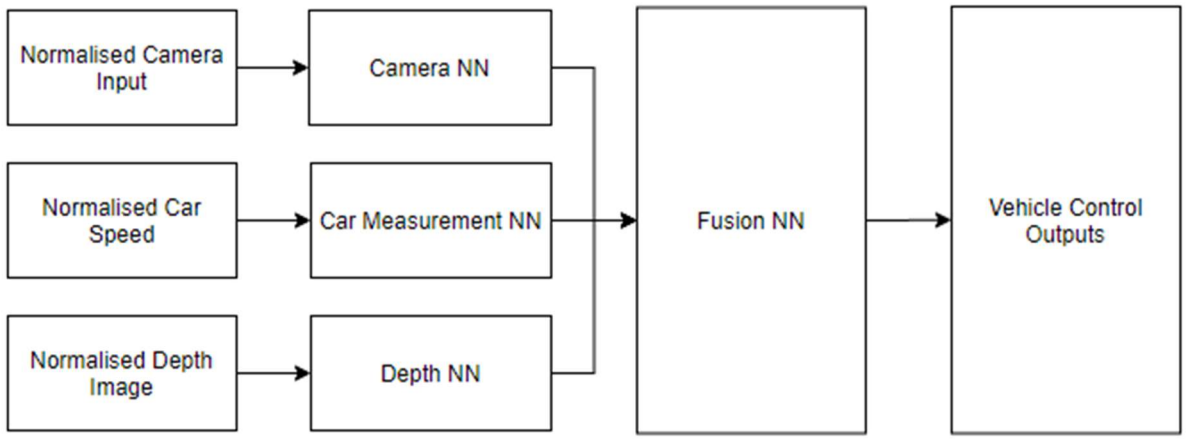

Figure 8: Final Design Structure

This structure allows for processing to be conducted by the neural network on each of the various inputs before this is finally fused together for final processing. This is an important feature as the various inputs are not similar in the data that is provided; therefore, it is important that the critical information is extracted from each of these inputs before using them to make predictions for vehicle control. The fusion network is therefore in charge of the main decision process for vehicle control and it does this by using the resulting outputs from the data processing networks.

4.2.1 Depth Image and Camera Neural Network

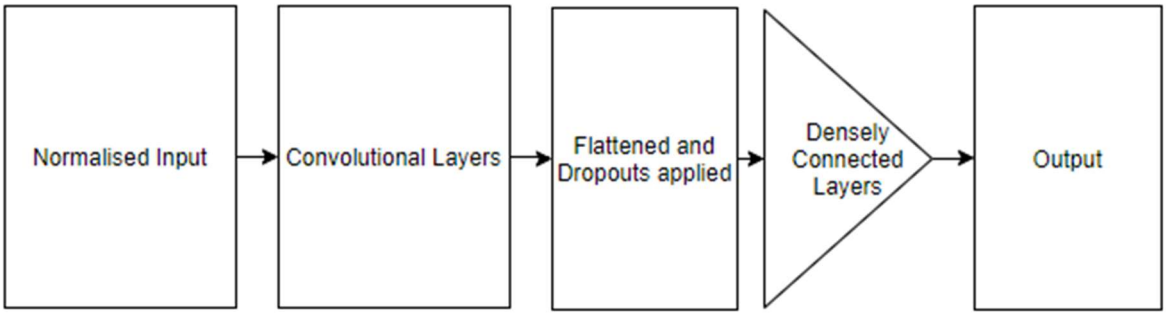

Figure 9: Depth and Camera Neural Network

As the depth and camera images are 2D, convolutional layers are used for the feature extraction. Convolutional layers apply 2D filters [7] to the input images that allow for geometric features to be extracted. This is one of the more efficient ways of processing the 2D images as other methods would rely on individual pixels rather than groups of pixels. Relying on individual pixels is inefficient as it requires more processing power, whilst giving similar results. Dropouts are then applied to connections, which are random in nature. This ensures that the network does not overfit to a specific area of the input images for making decisions. A number of densely connected layers are then used to produce the final out from this network.

4.2.2 Car Measurement Neural Network

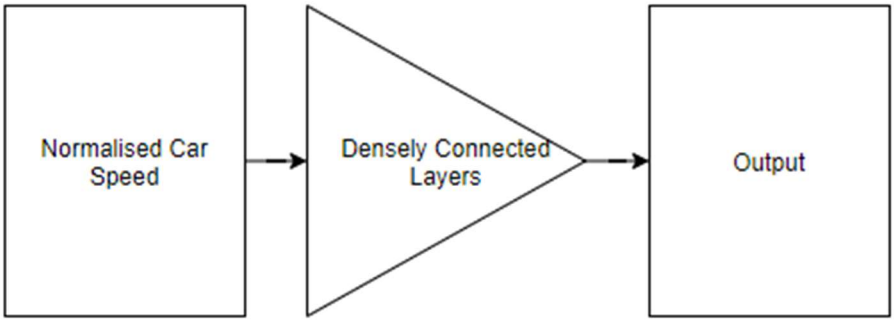

Figure 10: Car Measurement Neural Network

A simple network is used for processing car measurement inputs. The inputs to this network are simple values that are independent of the others. Therefore, a simple densely connected layer network is all that is needed to process the information.

4.3 Outputs

There are 3 outputs from the model, these include steering angle, throttle and brake. The steering angle is given by a value between -1 and 1, where -1 represent full left steering angle, 1 represents full right steering angle and 0 is 0 steering angle. The throttle output is of a range between 0 and 1, where 1 represents full throttle. Braking is similar to a range between 0 and 1, where 1 is the full brake.

The outputs are checked to ensure that there are no miss matching predications, such as full throttle and full brakes at the same time. In the case that there are such occurrences, numeric checks are done to

analyse the best action to be taken to keep driving without collisions. If the speed is too great when taking turns, the output for the accelerator is reduced to ensure the model is able to react fast enough.

5 RESULTS

This section of the report is to present the results of the tests carried out various model.

5.1 Using 20 Samples Size and 675 Epochs

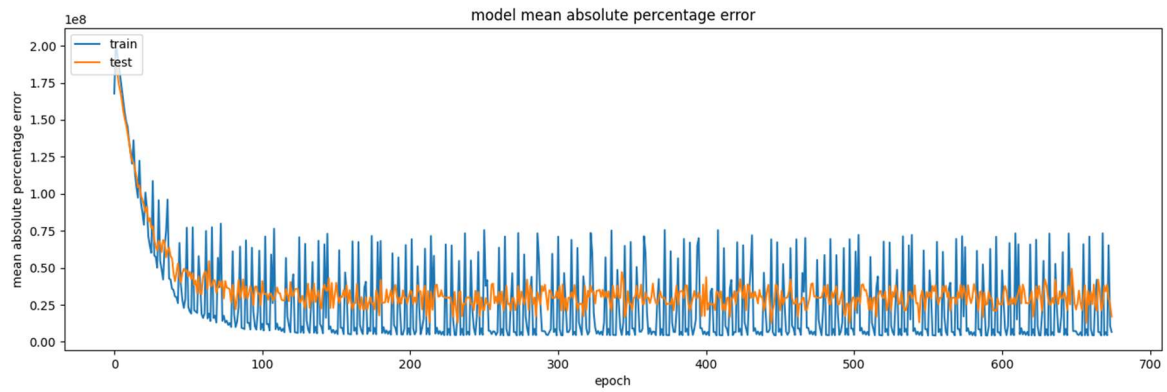

Figure 11: Percentage Error for RGB and Depth (20 Sample Size)

Figure 11 shows the results of a model trained on using a sample size of 20 and over 675 epochs. From the figure, it is clear to see that after 1000 epochs there are diminishing returns from further training. This means the model is overfitting to the dataset or reaching a local minimum. These same training parameters were used for the depth the only model and result for this can be seen at Appendix C. As these training parameters proved to be ineffective the models were discarded after training in favour for a more accurately trained model.

5.2 Using 288 Samples and 10 Epochs

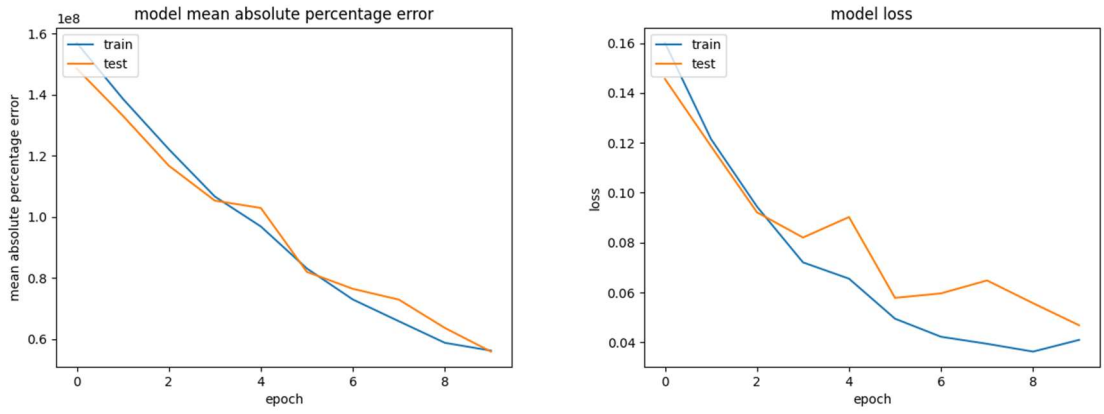

Figure 12: RGB Error and Loss (288 Samples)

Figure 12 shows the results obtained from the revised training method, using a sample size of 288 batches and 10 epochs for the camera only data. As compared to the other training parameter this is optimised quicker reaching an accuracy greater than the 20-sample method, whilst only using 10 epochs. As the loss graph still has a downwards at the 10 epochs mark it a be deduced that the model

has not overfitted the dataset. The is likely more improvements to accuracy that could be achieved if more epochs were run, however, the shape of the loss curve indicated that this would have diminishing returns after approximately 2 more epochs.

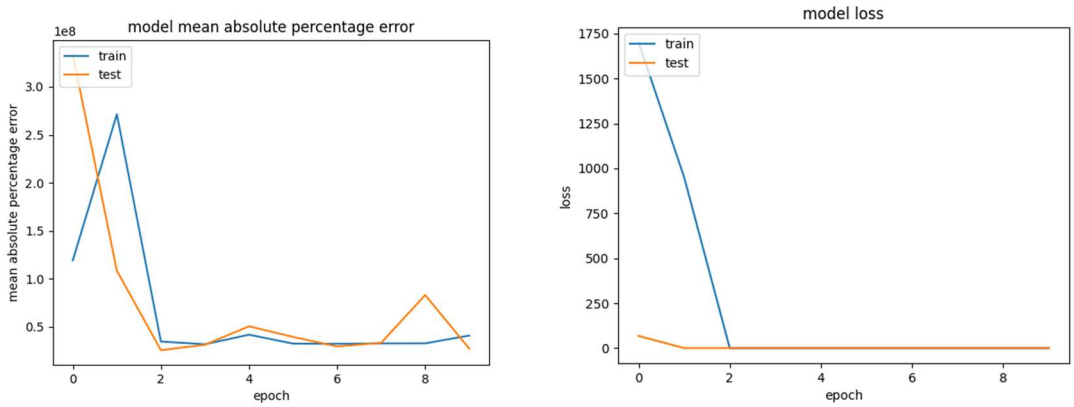

Figure 13: Depth Only Error and Loss (288 Samples)

Figure 13 shows the results obtained from the revised training method, using a sample size of 288 batches and 10 epochs for depth images only model. From this data is simple to deduce that the model reached a local minimum that it was not able to get beyond. This is likely due to the limited data that is available to the model to make predictions. So, the training has settled weights that achieve the lowest error.

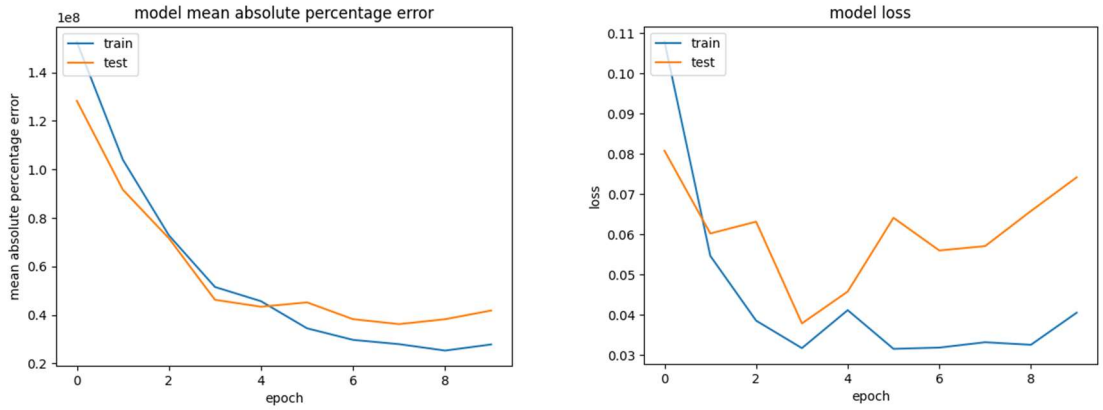

Figure 14: Depth and RGB Error and Loss (288 Samples)

Figure 14 shows the results obtained from the revised training method, using a sample size of 288 batches and 10 epochs for depth and camera model. This is a situation where the model has overfit the dataset and produced a solution that performs drastically worse when put in scenarios outside of the training dataset. Though the accuracy of the training set is similar to the camera only training, it is clear that test dataset did not perform as well, this is characteristic of a model that has overfit the training dataset. Increasing the dropouts in this situation could aid in getting a more general solution.

5.3 Speed of the Models

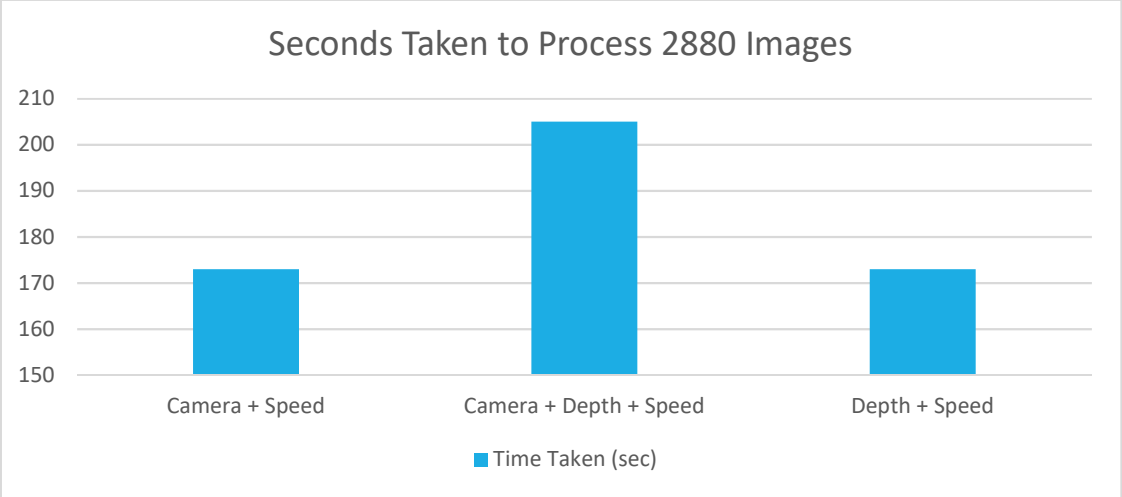

Figure 15: Time Taken to Process a Set of Images

The model's speed is vital to the real-world application of the model, Figure 15 shows the data collected for testing the speed of the models. The speed of the model was very similar for each of the three variations. Using the camera only and depth data only models managed to achieve a speed of 16 frames per second, which means that 16 predictions for steering, throttle and braking can be made per second. Whilst combining camera and depth data together achieves a performance of 14 frames per second. Each of these models is therefore capable of making adjustments to vehicle control fast enough to maintain smooth operation.

5.4 Driving

Whilst using a test dataset during training gives numerical data on the accuracy and performance of the model, it gives little insight in which specific situations the model succeeds and fails in. For gathering this information, a simple driving test was conducted in simulation. Through observation, it was apparent that the model performs very well when driving on uninterrupted roads. It is capable of keeping within the lanes and taking any bends in the road. However, the areas that the model falls short on are intersections and traffic lights. It often leads to erratic behaviour when an intersection or traffic light is reached, which results in collisions with objects causing the car to get stuck. There isn't a clear reason to why this is the case, however, it is likely due to the model not knowing which path to take.

6 DISCUSSION

End to end self-driving is an area that has a large amount of promise, and the results achieved in this area will only grow with improvements in technology and machine learning technics. The models that were designed and developed through this research are capable of driving a vehicle with limited path planning capability. Whilst other non-neural network forms of this is possible with even greater accuracy, the speed at which the development of this model is difficult to match. Therefore, with a number of more improvements, it would be possible to build a truly robust machine learning end to end self-driving model.

6.1 Improvements

There are a number of improvements that can be made to increase the accuracy of the models, these include larger datasets for training, incorporating directions to manoeuvre intersections, increasing the number of sensors used by the model and reducing the number layers of nodes in the network.

It was apparent from the test data evaluation conducted on the camera and depth model, that model was quickly able to fit and subsequently overfit the training dataset. This shows that there were areas for improvements given a larger data set was available. Therefore, increasing the size of the dataset as well as introducing further variety to the scenarios present, could yield a more accurate model.

One major limitation of the model is being able to manoeuvre intersections and traffic lights, this could be overcome with directions for which direction to take. As the model has no basis to make a decision on which direction to take at an intersection, it has had time dealing with these situations. One simple way to try and minimise this problem is to give the model a direction input. This can then be used when intersections are reached to give a general direction for the vehicle to head in. This reduces the uncertainty within the model as there will be clear instructions for which direction the vehicle needs to head. It is important that this is used in the training set also, as the connection between the direction and which way to turn cannot be established otherwise.

The model has proven that is capable of processing sensors quickly in its current state, however, in order to increase the number of sensors used by the model it is also important to reduce the number of layers/nodes used. The number of layers/nodes directly affect the time taken to process a given frame, therefore, a reduction in nodes can yield faster processing times and consequently faster predictions.

6.2 Areas for Future Work

Future work needs to be conducted on the areas outlined in the improvements section, as well as the feasibility of a model which takes in pre-processed data from other numerical based algorithms, to then make predictions for vehicle control. These are two different directions that can be taken and will come with the benefits and limitations of their own. However, due to the ever-evolving nature of the field, both of these paths should be considered.

7 CONCLUSION

The field of self-driving and neural networks are fast and ever-evolving forms of technology. Therefore, new advancements in the field are not always well tested. However, in order to stay on the cutting edge of technology, all these solutions need to be adapted and tested. This design combines aspects of an autonomous vehicle with machine learning technics to create an innovative solution to the problem of autonomy. There still a number of improvements that can be, however, in the early stages of the technic, it is already clear that is definitely a feasible solution. Therefore, ongoing research in this area is encouraged. Through incremental improvement and increased computing power, it will be possible to achieve complete autonomy with a high degree of accuracy.

8 REFERENCES

[1] T. H. Drage, "Development of a Navigation Control System for an Autonomous Formula SAE-Electric Race Car," University of Western Australia, Perth, 2013.

[2] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proccedings of the IEEE,* vol. 86, no. 11, pp. 2278-2324, 1998.

[3] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robotics & Automation Magazine,* vol. 13, no. 2, pp. 99-110, 2006.

[4] F. Merchant, T. Vatwani, A. Chattopadhyay, S. Raha, S. Nandy and R. Narayan, "Efficient Realization of Householder Transform through Algorithm-Architecture Co-design for Acceleration of QR Factorization," *IEEE Transactions on Parallel and Distributed Systems,* p. 99, 2016.

[5] ROS, "gmapping," 02 04 2019. [Online]. Available: http://wiki.ros.org/gmapping. [Accessed 11 09 2019].

[6] J. Aulinas, Y. Petillot, J. Salvi and X. Lladó, "The SLAM problem: a survey," in *Artificial Intelligence Research and Development: Proceedings of the 11th International Conference of the Catalan Association for Artificial Intelligence*, Amsterdam, 2008.

[7] "CS231n Convolutional Neural Networks for Visual Recognition," [Online]. Available: https://cs231n.github.io/convolutional-networks/. [Accessed 2019 09 10].

[8] K. Fukushima , "Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position," *Biological Cybernatics,* vol. 36, pp. 193-202, 1980.

[9] Easy-Tensorflow, "Convolutional Neural Networks (CNNs)," Easy Tensorflow, [Online]. Available: https://www.easy-tensorflow.com/tf-tutorials/convolutional-neural-nets-cnns. [Accessed 11 09 2019].

[10] N. Kalchbrenner, E. Grefenstette and P. Blunsom, "A Convolutional Neural Network for Modelling Sentences," *arXiv,* vol. 1404.2188, 2014.

[11] T. S. Huang, *Computer Vision: Evolution and Promise,* Urbana: University of Illinois at Urbana-Champaign.

[12] A. M. Bronstein, M. M. Bronstein and R. Kimmel, "Three-Dimensional Face Recognition," Israel Institute of Technology, Haifa, 2004.

[13] J. Browniee, “How to Avoid Overfitting in Deep Learning Neural Networks,” Machine Learning Mastery, 17 12 2018. [Online]. Available: https://machinelearningmastery.com/introduction-to-regularization-to-reduce-overfitting-and-improve-generalization-error/. [Accessed 10 09 2019].

[14] M. Bojarski, B. Firner, B. Flepp, L. Jackel, U. Muller, K. Zieba and D. D. Testa, “End-to-End Deep Learning for Self-Driving Cars,” NVIDIA, 17 08 2016. [Online]. Available: https://devblogs.nvidia.com/deep-learning-self-driving-cars/. [Accessed 10 09 2019].

[15] Y. Xiao, F. Codevilla, A. Gurram, O. Urfalioglu and A. M. Lopez, “Multimodal End-to-End Autonomous Driving,” *arXiv,* vol. 1906.03199, 2019.

[16] R. Li, C. Liu and F. Luo, “A design for automotive CAN bus monitoring system,” in *2008 IEEE Vehicle Power and Propulsion Conference*, Harbin, China, 2008.

[17] J. Bräuninger, R. Emig, T. Küttner and A. Loffler, “Controller Area Network for Truck and Bus Applications,” *JOURNAL OF COMMERCIAL VEHICLES,* vol. 99, no. 1, pp. 704-714, 1990.

[18] A. A. P. B. E. B. C. C. C. G. S. C. A. D. D. e. a. Martín Abadi, “TensorFlow,” Google, 2015.

[19] A. Brown, “Udacity's Self-Driving Car Simulator,” Udacity, Mountain View, 2016.

[20] G. R. F. C. A. L. V. K. Alexey Dosovitskiy, “CARLA: Open Urban Driving Simulator,” in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017.

[21] H. F. Cervone, “Understanding Agile Project Management Methods using Scrum,” *OCLC Systems & Services: International digital library perspectives,* vol. 27, no. 1, pp. 18-22, 2011.

[22] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” *arXiv,* 2018.

[23] O. Hartkopp, U. T. J. Kizka and W. Grandegger:, *SocketCAN,* Linux Kernal.

9 APPENDIX A: CONE DETECTION

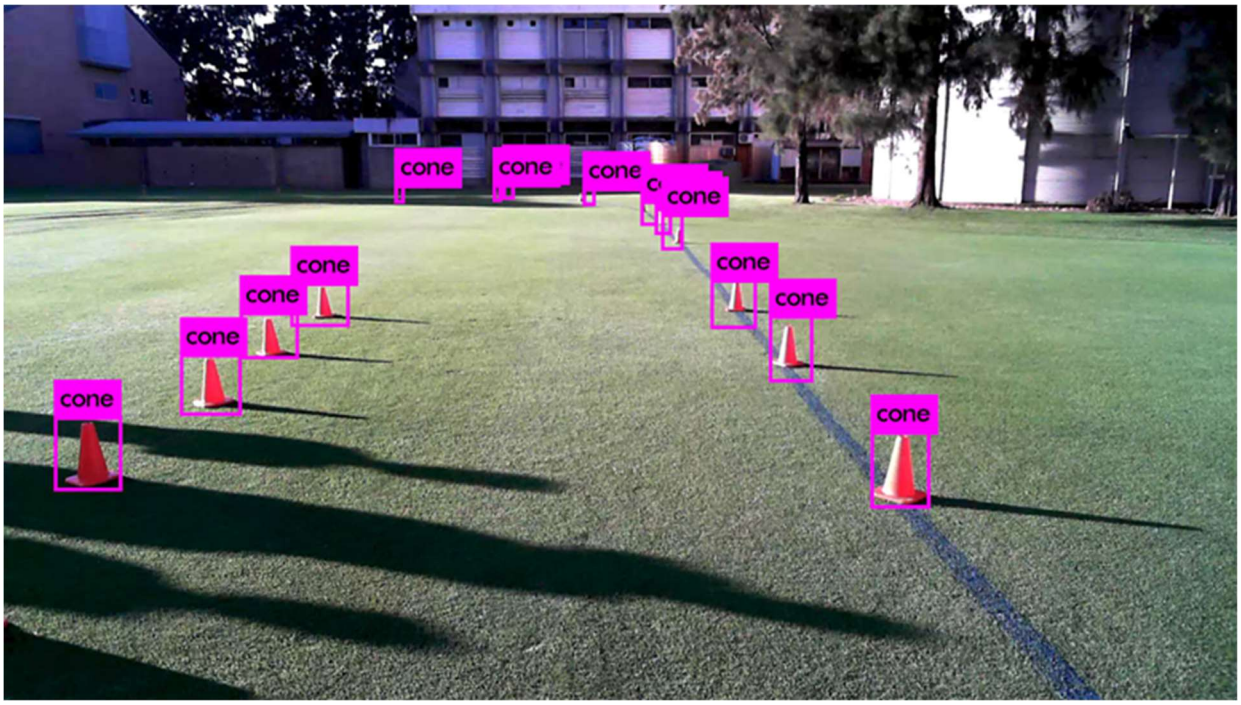

Figure 16: Results of the Cone Detection

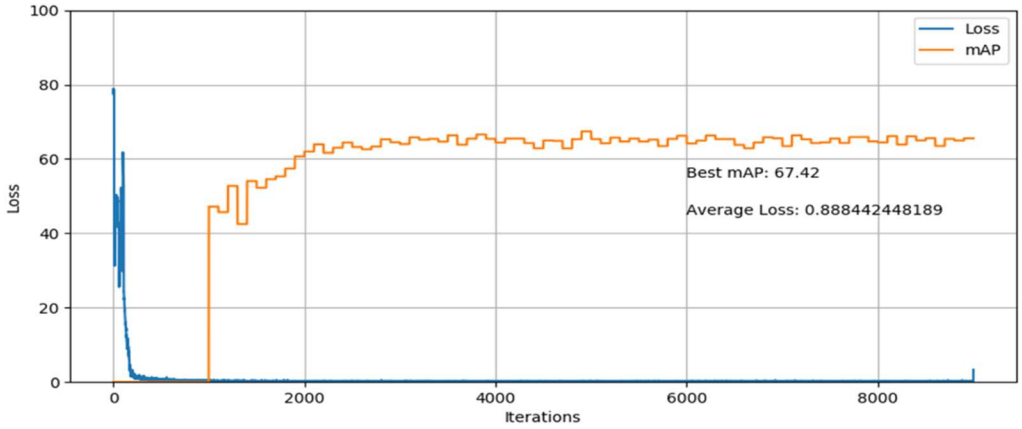

Figure 17: Training Data for Cone Detection Using YOLO

10 APPENDIX B: INPUT IMAGES

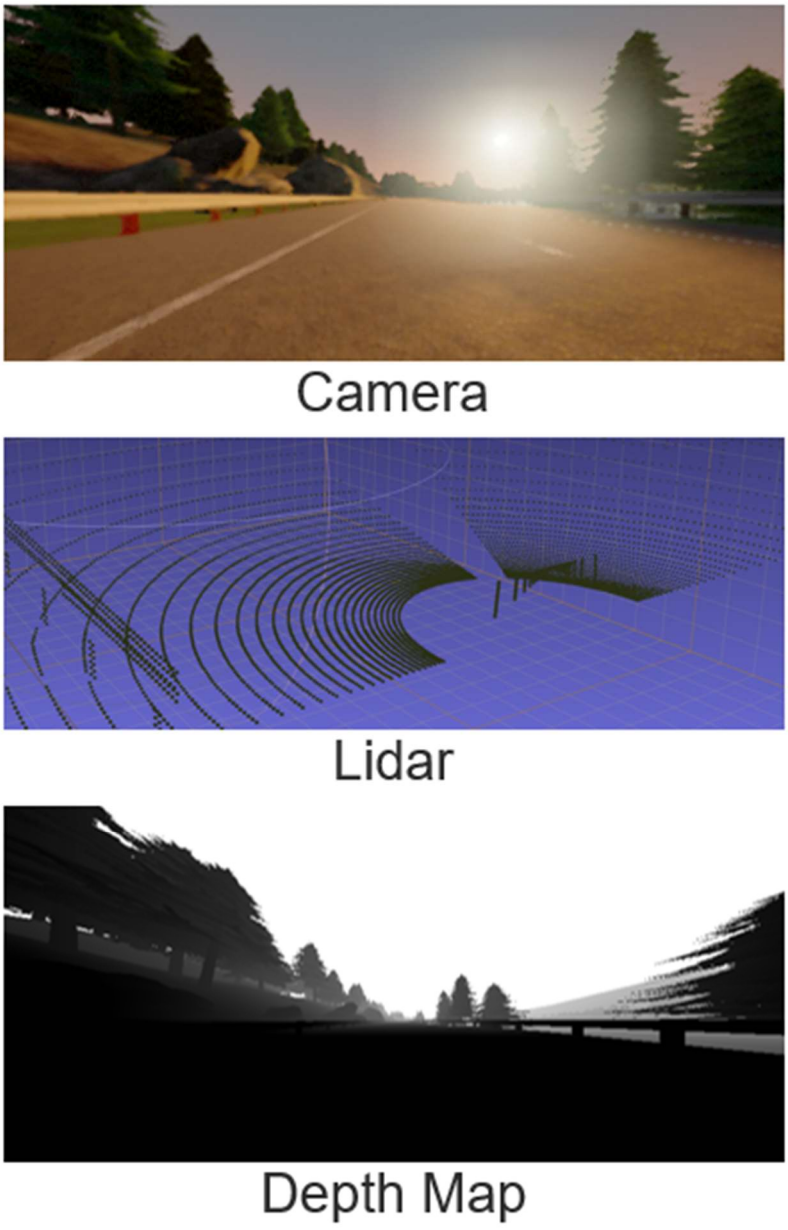

Figure 18: Input Images Displayed

11 APPENDIX C: EXTRA RESULTS

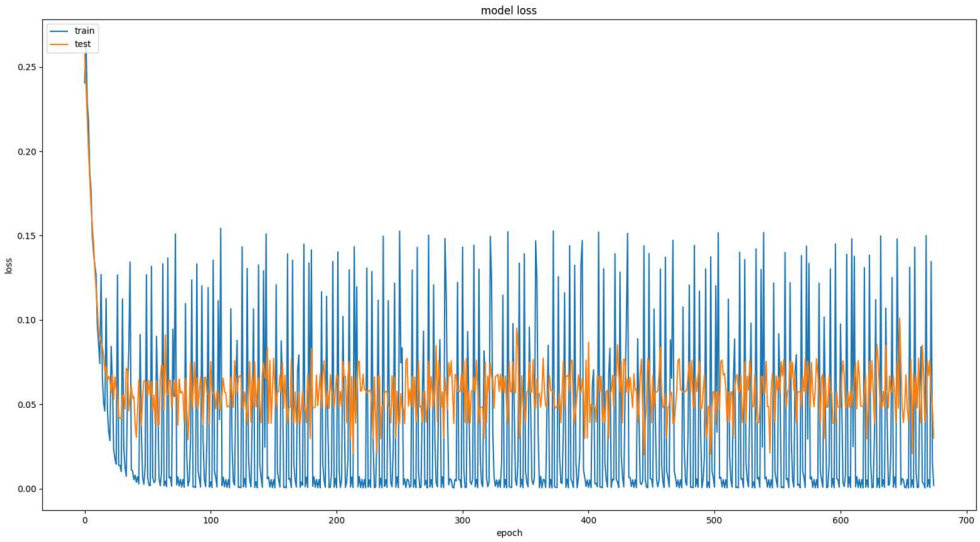

Figure 19: RGB and Depth Loss (20 Samples)

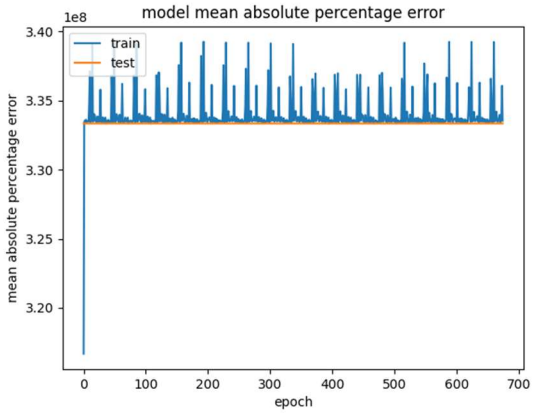

Figure 20: Depth only Percentage Error (20 Samples)

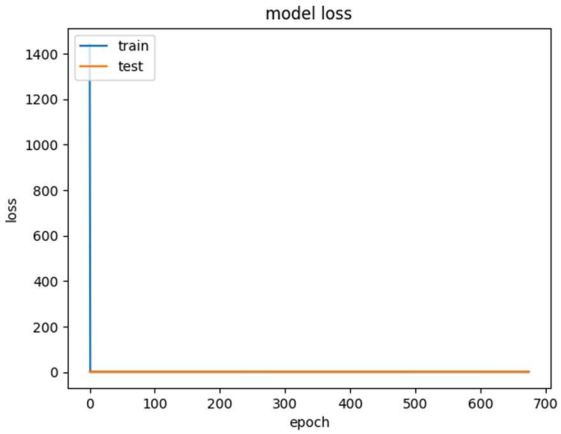

Figure 21: Depth only Loss (20 Samples)