# Deep Transfer Learning for Intelligent Autonomous Vehicles

Simran Deep Singh

A thesis submitted to Auckland University of Technology in partial fulfilment of the requirements for the degree of Master of Computer and Information Sciences (MCIS)

2020

School of Engineering, Computer and Mathematical Sciences

# Abstract

Autonomous driving has become a very interesting research problem for the deep learning domain. While Intelligent Autonomous Vehicles (IAVs) have developed significantly over the last 10 years, there are still unresolved issues concerning how to transfer knowledge from one driving environment to another. In particular, there is hardly anything known about how to get IAVs trained for driving on one side of the road (e.g., left-hand side in New Zealand and Japan) to right-hand side (e.g., the USA and China). This research describes how a deep learning IAV lane-positioning model can predict the steering angle based on continuous left-hand drive images and velocity inputs for 50 minutes of simulated driving (over 32,000 images) using convolutional neural networks (CNNs). We then examine freezing weights at different layers for successful transfer to right-hand simulated driving (10 minutes and over 7,000 images) and find that the best layers to freeze lie closest to the output layer. By visualizing the effects of weights at different levels, we report that the model shows signs of extracting increasingly relevant features at the higher levels that may help to explain how human drivers transfer knowledge about how to drive on one side of the road to the other. The overall contribution of this thesis is showing how a deep learning IAV model can adhere to lane-positioning by predicting the steering angle and can also transfer knowledge from left hand to right hand drive simulated driving.

**Keywords**— Image Recognition, Deep Learning, Steering Angle Prediction, Self-Driving Cars, Autonomous Vehicles, Convolutional Neural Networks, Deep Learning, Transfer Learning, Polder Blindness.

# Table of Contents

# List of Figures

# List of Tables

# List of Abbreviation

| | |
|---|---|
| CNN | Convolutional Neural Network |
| ANN | Artificial Neural Network |
| SGD | Stochastic Gradient Decent |
| MSE | Mean Squared Error |
| RNN | Recurrent Neural Network |
| RMSE | Root Mean Squared Error |
| IAV | Intelligent Autonomous Vehicles |
| LiDAR | Light Detection and Ranging |
| RADAR | Radio Detection and Ranging |
| fps | frames per second |
| SAE | Society of Automotive Engineers |
| DARPA | Defense Advanced Research Projects Agency |
| XOR | exclusive-OR |
| MLP | Multilayer Perceptron |
| ReLU | Rectified Linear Unit |
| PReLU | Parametric Rectified Linear Unit |

# Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

Signature:  Date: 18 May 2020

# Acknowledgment

This 90-point thesis was completed as the part of the Master of Computer and Information Sciences (MCIS) course at the School of Computer and Mathematical Sciences (SCMS) in the Faculty of Design and Creative Technologies (DCT) at the Auckland University of Technology (AUT) in New Zealand.

First and foremost, I would like to thank my supervisor, Dr. Ajit Narayanan, for sharing their knowledge and encouraging me to achieve the best. This thesis would not have been made possible without their help and guidance.

I would also like to thank my family and friends providing support during this period. Their words of encouragement ensured I gave it my all during this time.

I would so like to thank the teams behind CARLA and RoadRunner for providing the tools that helped this research. They helped ensure the tools functions well and kept open a channel of communication during this time.

Simran Deep Singh

Auckland, New Zealand

May 2020

# Chapter 1
# Introduction

*The purpose of this chapter is to introduce the background and motivation of this thesis. The research questions explored in this thesis are presented with a rationale and analysis. The goal of this thesis is covered next in which we link our research questions and methodology with current understanding. Here, we also outline the research contribution of the thesis. Lastly, the structure of this thesis is outlined.*

## 1.1 Background and Motivation

The New Zealand road code, like most road codes or highway codes internationally, can be summed up in a few easy to remember points that drivers should perform on a regular basis. These include speed awareness, blind-spot check, mirror check, lane positioning, hazard detection, braking distance and road condition awareness. These highlight the level of complexity that a task such as driving involves, for machines and humans alike. Unfortunately, humans are not immune to distractions and this diversion of focus results in many accidents, some of which are fatal. The goal of IAVs is to not only lower the number of accidents but also show that machines are capable of learning a task as complex as driving. However, there are tasks that humans can currently do better than IAVs, such as transfer knowledge of how to drive on the left-hand side of the road to the right-hand side. To effectively create machine learning models capable of steering a vehicle, we must first understand how humans would navigate their environment. This will not only, in such a situation, help us in better decision-making during model creation and training, but also give us insight to design and develop our models.

### 1.1.1 Human Behaviour

A human driver learning to drive follows a long training process, whether it be formal or informal training. There are steps that must be taken during each phase of driver training before they can become confident. A hierarchical description, Figure *1*, put forward by Keskinen, 1994, states four levels of driver behaviour (Keskinen, 1994). These levels show a progression of skills and how well each must be learnt before moving on. The initial skill listed is 'Vehicle manoeuvring', which includes, controlling speed, direction and positioning. The NZ road code also includes these points as they make-up the foundation for driving a vehicle. The ability to learn the foundations, or lower-levels, of driver behaviour may be dependent on the goals of the driver (Hatakka, Keskinen, Gregersen, Glad, & Hernetkoski, 2002). The highest levels on the hierarchy focus on the driver's 'Goals for life and skills for living', which may directly impact on the lower levels. If a driver does not have much use for driving, they may not expand resources

for mastering vehicle manoeuvring. This may also have some impact of the decision-making of the human driver.


Figure 1- Four levels of driver behaviour taken from Keskinen, 1994.

To identify the cause of crashes which involve a human driver, it is important to look at the decision-making process of humans. By identifying the factors that humans rely on to manoeuvre a vehicle, we can identify how a machine should perform this task as well. Research done by Evans, 1991, found that new drivers who display risky behaviour during driving are only doing so for developmental purposes. This may show that such drivers do not understand the associated risks of carrying out actions but do learn the impact (Evans, 1991). It was also found that, compared to experienced drivers, new drivers tend to move their eyes a lot and focus on features that do not provide value to the act of driving (Barjonet, 2001). As mentioned before, the learning itself is measured on the motivation of the user and their goals in life. When manoeuvring a vehicle, a human driver must, effectively identify various objects in their view and decide to steer the vehicle. These objects could include obstructions, road signs, layout or destination. In the driver behaviour hierarchy (Keskinen, 1994; Hatakka, Keskinen, Gregersen, Glad, & Hernetkoski, 2002), this is the third level. The ability to plan, navigate, set goals and provide context may ensure that a driver can accurately perform the act of manoeuvring a vehicle. Prior planning allows the driver to focus on the task at hand and attentively gather information about the environment, vehicle and the road. When it comes to gathering data around them, humans also tend to transmit their data to the other drivers on the road. Drivers do this to ensure they can predict other drivers and be predictable to allow for safe driving conditions. These levels of driver behaviour are general and specific to local conditions, such as whether to driver on the left-hand side or the right-hand side of the road.

Similar to machine learning models, humans are constantly learning to make these skills better through learning a variety of situations and road layouts. In many cases humans need to be able to transfer skills learnt in one scenario to another. In these cases, knowledge retention is very important as this knowledge may help in learning other scenarios quicker. A human driver can learn their environment through sight, hearing, smell, or combination of these (Bucchi, Sangiorgi, & Vignali, 2012). The quality of learning also depends on the goals of the human as well as the need to observe and experiment. When a human driver changes their environment, such as from left-hand side to right-hand side, they will often explore the qualities, rules and conditions of driving. How they perform in a new environment is related to their performance in their own environment. When learning to drive, the human must practice and automise basic vehicle handling and manoeuvres (Hatakka, Keskinen, Gregersen, Glad, & Hernetkoski, 2002). Otherwise, the driver will become exhausted from the constant focus on the road and the environment. This may decrease the ability to predict and be predictable to other drivers on the road. An experienced driver in a new environment displays similar mannerisms as a new driver in the same environment. Due to the constant focus, sometimes on irrelevant features (Barjonet, 2001), dealing with sudden emergencies while driving can become highly problematic. The more experienced driver will be able to adapt much quicker than a new driver.

Learning and practising a task, with proper future goals, will ensure that a human can get good at driving. Hatakka, 2002, has suggested that to become efficient in the task of driving, one must be able to automate basic manoeuvres (Hatakka, Keskinen, Gregersen, Glad, & Hernetkoski, 2002). The benefit of this is that the human driver does not need to continuously focus on the road and actively process information. This automation of task occurs in the later stages of learning when the human is an expert, and they know and understand the vehicle, environment and task very well. The expert must then go back to a novice and actively focus on the task if they enter unfamiliar territory (Michon, 1985). The expert, however, can leverage the previous knowledge and transfer their skills to become autonomous again, fairly quickly. Autonomy can also set in due to environmental features, such as polder blindness (SWOV, 2012). The term ‘polder blindness’ refers to the instance when the driver has reduced alertness on

straight roads, due to low traffic, wide flat landscape, or low visual obstructions (Koornstra, 1989; SWOV, 2012).. This can occur if the environment is of the above description and given the driver is an expert, they can transition into an automatic state. While in this state, the driver is not actively perceiving and processing information about their environment or their vehicle.

Based on the psychology of how humans initially learn to drive, we know that practice is crucial to this field. The motivation and goals of humans guide the strength of learning the foundation skills in driving. We can see that perception plays an important role in the task of driving as it is primarily how humans learn about their environment, among using other senses. Apart from that, experienced drivers focus on features on the road that are most probably related to controlling the vehicle effectively (Crundall & Underwood, 1998). Initially, novice drivers lack this focus and make many mistakes in an effort to learn and develop this crucial skill. We can understand that failing is important for the development of self. We can also understand that a machine learning model would work best when using a camera-based input to mimic perception, but also other inputs to supplement vision.

### 1.1.2 Machine Learning Background

The area of image processing has shown significant growth over these past years with the development of faster hardware and sophisticated algorithms. In turn, the development of autonomous vehicles explored many areas such as computer vision, path finding or sensory input (LiDAR, RADAR). Since the inception of self-driving cars, Campbell et al., 2010 reports that they are now able to assess their environment, detect and classify objects and apply some path finding (Campbell, Egerstedt, How, & Murray, 2010). For a trained human, these tasks may seem trivial, but for a machine, performing these tasks requires lots of data and computing power. When humans start learning how to drive a vehicle, they use their previous knowledge, gained through experience, of what a road is, what road signs are or roles of the driver etc. They also are taught by experienced drivers, reading the road code and passing multiple tests. Emphasis is also

put on practice and drivers learn to drive in an isolated environment before taking to an open road.

New Zealand drivers are required to drive on the left side of the road, whereas some other countries require drivers to drive on the right side. This, along with countries having different road signs, road markings, and road codes means the driver must adjust in different geographies. Humans can transfer their learning from one geography to another without having to re-learn the task. For a more experienced driver this maybe easier than someone new to the task of driving in general. This can be attributed to experience and knowledge gained from learning previous, similar tasks.

Machines require lots of data to complete this task. Here, the type of input data is very important and must be carefully curated by the human to ensure the models learn enough to generalise. There do exist some assumptions regarding our data and the method of collection. We use a self-driving car simulator to collect simulated data through the use of an RGB camera sensor and a speed sensor. We also have a steering angle sensor to collect the steering angle associated with the other sensor inputs. We assume that the link between simulated inputs versus real-data is strong. In fact, there may be some scenarios that don't exist in real world datasets which can be simulated to provide more information to our model. The data being captured by our sensors is also assumed to be representative of the current actions being done by the simulated vehicle. The sensor input is also collected from within the simulation. As we are using a well-known open sourced simulator, we assume that the assets, sensors, and collection methods are proven to be stable.

This data can then be fed into a machine learning model to allow the model to train on the data. The goal of the training process is to ensure the model learns to predict a steering angle, or other autonomous driving related task. The task of predicting the steering angle, given an image, and sensor data as inputs, requires the model learning the ground truth steering angle. The ground truth is the actual steering angle as reported by our simulator. The model must learn the features present in the training data and then give an output steering angle. The processing of images can

be done with the use of deep neural networks, specifically convolution neural networks as they work best with image data. An image of the road contains many possible features, such as road markings, barriers or road signs, which may contribute to the learning and steering angle prediction process. On the other hand, many features may not contribute to our goal and therefore must be handled in an appropriate manner.

Apart from large amounts of data, deep learning requires sufficient computation power to process the data. The training process is the phase where the model is learning the hidden patterns within the training data and reducing its error rate. Depending on the size of the model, type of input data and the complexity of the model, this phase can take anywhere from a few hours to weeks. This phase can be sped up by adding more hardware which would incur extra costs. Apart from extra hardware, software development practices must be applied properly to ensure minimum overheads during the training and deployment phase. To speed up the training process, fewer data can also be given, which in turn may result in a less generalised model and a higher error rate. That said, we plan to investigate the effects of variable sampling rates on straight roads vs bending roads. This is due to the phenomenon known as polder blindness. We can quantify the variable sampling rate on straight roads by measuring the error of various rates versus equal sampling of straight roads and bends.

Currently deep learning models consume a lot of resources to train and maintain, especially given the unpredictable nature of humans. As autonomous vehicles are tested on populated, urban roads, there always seem to exist some event that the model has not been trained for. For each new task, new and relevant data must be collected, annotated, fed into the model which will then spend some time training on that specific task. Some methods, such as transfer learning, exist to reduce the cost of training a new model. Transfer learning helps by using knowledge from one task to perform another task. It can help in the deep learning pipeline by allowing the mode to learn faster or avoid collecting large amounts of annotated data.

### 1.1.3 Motivation

My motivation for this project is to identify an architecture that can predict steering angles, but also attempts to learn the same features that human drivers view as important. Despite increased knowledge of how human drivers learn to drive, no computational models exist concerning how human drivers, or IAVs, transfer knowledge from left-hand side to right-hand side driving. This will give us insight to the steps needed to solidify our current understanding of deep neural networks for autonomous vehicles. What also motivates me is the idea of glancing inside the black box for deep neural networks in an effort to understand the decision-making process. A human can simply be asked to provide a reason for their behaviour whereas a neural network makes this process challenging. Interpreting deep learning models can have major benefits in the industry by removing a barrier of entry for black box models.

Apart from that, I aim to explore the benefits of the human driver going into a 'passenger' mode to let the brain 'control' the car with little active focus. This phenomenon may occur due to the human driver being confident in their abilities to drive a vehicle, especially learning the driving foundations (Hatakka, Keskinen, Gregersen, Glad, & Hernetkoski, 2002) enough to automise the basic manoeuvres. This would show that some trigger maybe present in the brain that when significant change occurs in the environment, give the control back to the driver. To do so, I attempt to uncover this through experimenting with different sampling rates and identify the benefits it may or may not have. The human brain may transition into this state to either conserve energy, or the human may already be tired. This would allow us to experimentally measure if a machine learning model is capable of such change.

As mentioned before, one of the levels in the driver behaviour hierarchy is the mastery of traffic situations and scenarios. This would allow the driver to attain confidence and composure when faced with an unknown scenario. The driver will be able to tap into their prior knowledge of similar events and will only need little adjustment when faced with a new challenge. This is of course based on the experience of the driver but is an import skill that the human brain possesses. Such as, driving on the left-hand side and transferring knowledge to the right-hand side. We can simulate this behaviour using transfer learning, discussed in depth later. The idea

behind this is that we can take some learning from a different task and perform another similarly related task with little practise.

## 1.2 Thesis Objectives

In this section, we shine light on the objective of this thesis, state our research goal and how we plan to achieve it. In this thesis we introduce experiment-based evidence for multi-input deep learning models for steering angle prediction. We also experiment with variable sampling rates to measure computational expense of our deep learning approach. Transfer Learning is also experimented with identifying the feasibility of using a left-hand driving model for right-hand driving applications. This will help training intelligent autonomous vehicle models in many different environments without having to start from scratch.

To achieve our first goal, we research and develop a machine learning architecture that will be suitable for learning steering angles from multiple inputs. As autonomous driving is still in its infancy, this model will provide input into multi-input models and how they can be used for autonomous vehicles. To measure success of this step we will be tracking the error rate. We will also be looking to see how generalised this model is, as it will then be used in transfer learning on a right-hand side driving dataset. This will be done by visualising trained filters and understand the learning.

Secondly, we will apply programming logic to create a variable sampling rate for straight roads. We will experiment with various sampling rates and to measure the success of this step we will track the error rate and the model execution time. As error rate is more important in this case, there will be some leeway on this to allow model execution time more freedom to be significant. Meaning, if model execution time shows considerable change and the error remains relatively unchanged, we can measure the statistical significance of the change.

Finally, we will use the final model developed in a transfer learning task. We will apply different adjustments to the model, such as freezing all or some layers, and adding extra layers. To measure the success of this task, we will track the error rate, and evaluate if the error rate is

lower or similar to the original. We will also evaluate if a lower or similar error rate can be achieved in fewer epochs than the original model.

## 1.3 Research Questions

Autonomous vehicles are seen as one of the greatest challenges for deep learning algorithms to accomplish. But, as mentioned before, they are very costly to train, and complexity increases as testing reveals cases for which the model has not been trained for. When moving from a simulated environment, real or virtual, to human populated environments, random noise such, as bad drivers or hidden objects is added. Ensuring that we keep a human focus in this thesis, we decide to tackle some research questions that we identified as the required foundation for driving by psychologists. These questions tackle the first and second level of the driver behaviour hierarchy and will provide great input to the research community as well.

The goal for this thesis is to experiment with end-to-end deep learning by creating a multi-input model capable of predicting the steering angle on a left-hand drive dataset. Then that model will be transferred to a dataset of right-hand side driving to identify how learning is transferred. To reduce the computationally expense of the model, a variable sampling rate will be applied to simulate polder blindness on straight roads. Formally, the research questions presented in this thesis are:

**Research Question 1:** Which architecture will best train a CNN that can predict steering angles from a given image?

To achieve our goal, we will apply supervised deep learning techniques with a CNN model. This model should be able to predict steering angles, within minimal error, and show evidence that it is extracting high quality features from the images to base its output on. This will be examined by extracting feature maps and filters trained by the model to identify which regions get activated and how crucial they are to the learning process, from a human's perspective.

**Research Question 2:** Does a variable sampling rate on straight roads have any effect on the model without compromising on the error?

As a CNN takes images as input, video files must be sampled, in our case at 10 frames per second (fps), to produce a collection of images. We will then experiment with a variable sampling rate between straight roads and bending roads to see if the model changes the error rate. The sampling rate for bending roads will be at a constant 10 fps, only the straight road will be sampled.

**Research Question 3:** Can the final model, trained on left-hand drive data, be transferred over to a right-hand side driving scenario? What error rate does this achieve?

For humans that travel to other countries, getting used to the different road conditions such as driving side, road markings, rules can be confusing initially. To succeed at driving, they leverage the skills they have learnt previously, from driving, to quickly adapt to the new environment. Can our model effectively learn to output reasonably accurate steering angles based on previous learning as well. This process of taking the skills learnt from one domain or task and transferring them to a new domain or task is known as transfer learning. It is possible that if our model successfully transfer knowledge, we may be able to shed some light on how experienced human drivers manage to cope with driving on the other side of the road.

## 1.4 Structure of This Thesis

This section details the structure of this thesis:

In Chapter 2, we carry-out a literature review of the field of machine learning, image recognition using deep learning methods, transfer learning and autonomous vehicles. Specifically, we identify initial self-driving vehicle models, and currently how image processing algorithms have been used in this field and the current understanding. Additionally, we explore the use of multi-input algorithms specific to the task of autonomous vehicles.

In Chapter 3, links to our research methodologies are highlighted in relation to our research questions. Here we state the process we take to achieve our goals whilst providing evidence for the steps chosen. We also look at the research scenario, data collection parameters, implementation of our experiments and how they will be evaluated.

Chapter 4 discusses the results of our experiments and how they solve our research questions. Here we look at any biases or limitations that hindered the performance of our experiments.

In Chapter 5, we critically analyse our results and discuss their relation to each respective research question. We visualise and analyse the trained filters, feature maps and saliency maps to better understand the model. We aim to interpret the black box to understand if they match with the psychologist's perspective on human driving. We also make clear if our research goal has been achieved, given the outcome.

In Chapter 6, we conclude our research and thesis and provide more input to each research question or the research goal. We also look at any future work that can be done to extend this research and gain insight into deep transfer learning for intelligent autonomous vehicles.

# Chapter 2
# Literature Review

*The purpose of this chapter is to shed light on the origins of machine learning algorithms and how they have transformed to be used to develop autonomous vehicles. Initially, we look at traditional machine learning methods that have been used in image processing. We then look at how autonomous vehicles were conceptualised, and which methods were initially used. We then present current literature regarding state-of-the-art image processing methods and how multi-input autonomous vehicle algorithms are designed today.*

## 2.1 Introduction

In the recent years, image recognition has been seen as a very important part of the machine learning field to research. Image recognition involves giving a model some input images and ensure the model can recognise them accurately through various techniques. These techniques may not be machine learning based, such as edge detection, contour detection etc, but form a foundation for this field. The process to classify images follows a supervised learning approach, as this allows the human to set the ground truth value for the labels. The machine learning model scans the images to learn patterns for the object in question. Supervised learning is the process of training a classifier, some nonlinear function, using annotated data, where each input is labelled by one or multiple subject level experts. The nonlinear function takes as input some n-dimensional array. The output of this nonlinear function is some value derived by computing the nonlinear function. As an example, to recognise a dog in an image, the image will get converted to an array of pixel values. The output of the network will be a value signifying a dog. The supervised learning method requires lots of time and money to produce a quality labelled dataset. Given this fact, supervised learning is still the most common technique used to train machine learning models.

Supervised learning has shown many great feats in various fields especially image recognition. On some benchmarks, image recognition models have exceeded human accuracy, such as 99.0% accuracy on the CIFAR-10 dataset (Huang, et al., 2018). With the development of sophisticated algorithms and faster processors, machine learning is now being implemented in many tasks, especially autonomous driving. Society of Automotive Engineers (SAE) International, in their 2014 report, put forward a 6 - level system of driving automation. These signify the level of driving automation based on who monitors the driving environment, how much assistance is provided by the user, and if the user needs to assist at all (SAE International, 2014). These levels are determined by how much responsibility is given to either the driver or the autonomous system. If full responsibility falls on the autonomous system, then it would be classified as a level 5 autonomous vehicle and level 0 if full responsibility falls on the human, like traditional automobiles. Currently it is believed that the semi-autonomous vehicles produced

by organisations such as Tesla and Waymo fall under level 2 -3 of the above scale. Given our experimental results, we will also be able to classify our steering angle prediction model and transfer learning model into one of these levels. The metrics we will use to judge which category our model falls into is the mean square error and the output visualisations of our model. Our goal is to minimise the error to show our model is capable of predicting steering angle as accurately as possible. The visualisations will show us that the model is learning some features that a human also learns when they are driving. In the following sections we look at the psychology of driving from a human perspective, the past work done on self-driving cars, neural network implementations and finally deep learning implementations. We also go over the structure of neural networks, including layers, methods, metrics, activation functions etc.

## 2.2 Psychology of Driving

We initially looked at high level human driving behaviour and the method of learning. We found that the goals and motivation of a human learner influence how good of a driver they become in the future (Hatakka, Keskinen, Gregersen, Glad, & Hernetkoski, 2002). The more effort put on training the foundations of driving, the driver can become more confident in their skills and learn to automate common mannerisms. To understand how a driver behaves while driving, we need to look at how they learn and how they become an experienced driver. The process of driver learning can shed great insight on how self-driving vehicle models can be taught to perform similar to a human.

Perception is an important sense, and the driver behaviour has strong connections to perception (Bucchi, Sangiorgi, & Vignali, 2012). As a learner driver, one must focus on understanding which areas of the environment to focus on at what time. This ability is built from experience, but the strength of the ability is dependant of the motivations of the learner. Van der Molen, 1988, states that observing the physical environment has some influence from past experiences (Van der Molen & B"otticher, 1988). In many cases a learner driver has seen many people drive and know some high-level details of basic driving. These could be, staying in one's own lane, being courteous and giving way especially to road hazards etc. Bucchi, 2012, also

argues that perception is not only about the ability to observe and learn from the environment through the eyes, but a combination of the other senses. Using a combination of these senses can allow the brain to make a more complex judgement about what to do in a given scenario. However, new drivers do lack the required experienced based decision-making as more experienced driver. Identifying which decision to make given the information of the road and vehicle environment can have consequences that could be fatal. To ensure a human makes the right decision they need to prioritise which features on the roads pose the most risk or provide the most information to make an informed decision. It was found that new drivers would be putting focus on either all features on the road or on the features that provided little to no input for decision making (Barjonet, 2001). On the other hand, more experienced drivers were focusing, in most cases, on the right parts in the environment. As newer drivers tend to make more mistakes, we know that this will help them learn and narrow down which features are important for decision-making. Now, if an experienced driver was moved into an environment where they have never driven before, it can be likely they will also focus on features not important to decision-making. This maybe because as it is a new environment, such as another country, they need some time to learn the differences and adapt. This process will be much more comprehensive and faster than a completely new driver, however.

The idea of newer drivers learning is thought to be focus on trial and error rather than a 'classroom' based learning environment. The more mistakes drivers make early on, the more scenarios, environments, vehicles they are exposed to, the faster they can learn (Bucchi, Sangiorgi, & Vignali, 2012). The mistakes made serve to fulfil the development of driving needs rather than a sign of bad driving (Hatakka, Keskinen, Gregersen, Glad, & Hernetkoski, 2002). As time goes on and more practise is put into this task, there will be improvements. By using perception, the human driver will start to learn each time they practise. The human will start to learn the basic requirements of manoeuvring the vehicle and how to behave in various traffic situations. The human mind should also learn to automatically detect and rely on features, on the road, that provide the most input for future decisions. The mind will start learning not only the obvious features present, but also the subtle detail. This could include, distance of braking, the

feel and control of the vehicle etc. When a driver sits in a different vehicle, the first few times they may brake slight harder or slower due to them not knowing the optimal pressure to put. Apart from that, basic mannerisms should also be automated by the brain, in an effort to not actively process information. If the human driver does not learn enough to automate certain actions while driving, the brain will be doing a lot of active work, which will become tiring. This step maybe dependant on the future goals of the human. If one does not have a requirement for driving in their future, the lack of experience will not allow the brain to transition into an automated state. Another phenomenon is polder blindness, in which the driving will transition into passenger mode and not have an active focus on driving (SWOV, 2012). This will usually happen when the environment does not appear to be changing, i.e. open flat roads with no traffic and a long journey ahead. This could show that, apart from learning and mastering the basics of driving, your brain also starts to learn when active processing and focus is required.

The task of learning to drive, alongside how to act in different situations, how to control a car etc is quite dependant on memory. With this, the human knows what decision works best in each situation. Apart from that, memory allows humans to learn the actions of other drivers on the road. By trying to predict the actions of other drivers, the human learns to decrease their reaction time by being slightly prepared. This is helped by the fact that, for the sake of safety, the drivers want to be as predictable as possible with the use of signals, lights etc. The action of predicting other drivers comes once the human themselves have learnt the basics of driving a vehicle. This is because the driver can focus less on the minor driving tasks and more on the journey, goal and planning instead (Hatakka, Keskinen, Gregersen, Glad, & Hernetkoski, 2002). The prediction of other drivers is not only required for safety, but also for planning ahead. If a human driver sees a car pulling out of a driveway, they know from experience that they can either stop and give way or keep going, hoping the other driver would stop. Once the driver has made their mind for the next step of the journey, predicting the steering angle should be an automated task. The driver does not have to actively think about how much to turn the steering wheel as the many hours of practise has given the brain enough knowledge. This also applies to the scenario when the driver transitions to a passenger mode and allows the brain to take care of basic tasks.

However, there are some instances where even experienced drivers will have certain issues driving actively. A sudden change, or ongoing change in the environment forces active focus, i.e. city driving or surprise hazard, or a change of environment with new markings or even opposite driving side. In the latter case, experienced drivers can adjust much more quickly due to their extensive memory and experience. However, in the case of focus and sudden hazard introduction, younger, newer drivers can adjust quicker than experienced drivers, due to their reaction time (Evans, 1991). This goes back to the finding of newer driving focusing on the right features in their field of view. Even though the reaction time is quicker, the decision made that follows the reaction may not be the best one for the given scenario.

In the larger picture of artificial intelligence, the psychology of how humans perform tasks is quite crucial. From the above studies conducted by psychologists, we know that perception is quite crucial to identifying features on the road. Other senses play an important role too, but they may be supplementing vision which the human driver largely relies on. Later, we will explore how purely vision based self-driving car models compare to multi-modal networks, which rely on a variety of other sensors. Perception is not just limited to identifying and understanding features on the road but also the vehicle and the capabilities and limitations in different environments. A human driver maybe confident in their vehicle for certain environments, but maybe have some trouble in someone else's vehicle. This may show the need for a machine learning model that understands the limitations of their vehicles and adjusts as required.

When looking at how human drivers learn, we know that before they start driving, they have some experience looking at others driving and know the basic road rules. They then spend time learning in a simulated environment, i.e. a parking lot or a test course, from a trained, experienced human. This maybe because, directly training on active roads maybe too overwhelming with the number of variables to keep track of. Once they become confident enough, they move to an uncontrolled environment where they can make and learn from mistakes. They learn to identify features on the road that are crucial of decision-making and manoeuvring the vehicle. This is the behaviour we expect from our machine learning model initially. The deeper the model goes the more generalised features it will learn. From the model's perspective, these features will be the

most important for predicting steering angles. A confident human driver can automate the regular, minor tasks that don't require active focus. The brain can then adjust to focus on predicting and planning based on the field of view. The transfer of focus is also to reduce the active computation needed to keep focused on minor tasks. We expect that this skill maybe crucial for reducing the time and money it takes to train neural networks on large tasks.

Mastering various traffic situations is crucial for a human to be an excellent driver. By being exposed to and remembering from past experience how to behave in different situations will allow the driver to be much better at what they do. To a driver to be able to do this consistently and learn from previous unseen situations, requires them to have sufficient practise. The more practise they have the better they will be equipped with handling new, surprising, unseen scenarios. The same goes for machine learning models. Having a model that is trained in the USA, will be able to perform well in that environment. The limitation being that, if the vehicle was to be shipped into another environment, they may require training from the ground up. Instead, transfer learning can be used to identify if the model is generalised enough to be used in another task that belongs to a similar domain. Our goal should be to create a model which can learn to be generalised enough that if the driving orientation was changed, it can still output steering angles. Not only that, our model should also be able to learn from the new data quicker whilst maintaining or beating earlier error. This is one of the benefits of transfer learning but also because an experienced human driver in a new environment can adjust very quickly, as compared to a novice.

## 2.3 Self-Driving Vehicles

As mentioned earlier, currently self-driving vehicles rely heavily on deep learning algorithms, which, in all fairness, have shown outstanding results. To get to this point, earlier self-driving did not have any automatic feature extractors or extremely efficient hardware to develop. They used traditional algorithms, image processing, and human annotated features to ensure the model is fed accurate data.

In the case of Thorpe et al., 1987, the task of creating a method for road following involved explicitly designing every part of their model. This includes creating the edge detection filters, categorising pixels as road, grass etc (Thorpe, Hebert, Kanade, & Shafer, 1988). Apart from the extensive pre-processing and training process, due to limitations, the model could not be tested over a certain speed. They would also have to ensure that their test data fell under certain specific parameters before being fed to the model, for best results. Given the number of obstacles on the road, and that maybe encountered later, this whole process must be repeated.

The approach to solving the limited computational issue, during this time, is the use of parallel processing or using sub-models or sub-systems to delegate and carry out tasks efficiently. The method of road following, however, still remain quite similar, by extracting boundaries from a given image. During training, the manual annotation is done for these regions. Apart from that, Kuan et al., 1988, apply the use of segmentation, where they segment road and non-road locations in an image. However, a 'hard-coded' colour transform is applied to the training images to show the difference between the different environments (Kuan, 1988). This is very limited as in many driving scenarios the road and surrounding environment could be quite similar, i.e. rural gravel, dust roads. This technique may cause issues, especially since the model was trained for a military vehicle which maybe travel off-road. To speed up the execution of their model, to handle faster speeds, they split the model into multiple sub-systems. The perception and detection work in parallel to the control system. This method is still not as efficient as an end-to-end deep learning model that is currently used.

Other image processing models made use of Kalman filters and perspective projection (Dickmanns & Zapp, 1987). Kalman filters are used when the source of data is either very noisy, has multiple sources or is prone to error. The filters provide an optimal estimation of the source, in this case road curvature from images, by combining multiple data-points. Their goal was to create a method of identifying road curvature by correlating contours from visual images. This would mean that they assumed the distribution of the testing data contained input features that correlated with the training data. Finding contours that were correlated with other features on the road is highly unlikely in non-urban environments. The other limitation to their model was the

fact they were limited to highway driving with well-maintained roads and visible lane markings. This would show that their method was not able to filter the noise that exists. However, their method was capable of processing inputs and making decisions at very high speeds, which was good for that time.

Some of the real push for autonomous vehicles came from the DARPA yearly challenges. DARPA, (Defense Advanced Research Projects Agency) a US military research agency, hosts these challenges for people to enter and create the best possible autonomous vehicle for a variety of test courses. The techniques used in these challenges range from traditional image processing to current deep learning based.

RASCAL, an autonomous vehicle created for this challenge, utilized a similar approach to Dickmans et al, 1987. They too used Kalman filters to pre-process the input data, but relied on a more hard-coded method, using waypoints, for road following. Multiple cameras were used to identify road boundaries, including any obstacles that may be present. Their model was a combination of cameras and sensors which formed a tight-knit workflow (Behringer, Kubinger, Herzner, & Fehlberg, 2005). However, there seems to be no weight against which type of input would work best for the task. Humans, in a sense, rely on perception as the major source, but are supplemented by other senses. Their approach could have weighted some input to be preferred over another.

Unlike complex approaches, simpler approaches may yield fruitful results too. Bebel et al., 2004, create a model that simply avoided obstacles, and predicted the steering angle to the next waypoint, supplied by DARPA. Due to the time restricted put in the challenge, they decided to employ software development practises to ensure their vehicle finished on time. The idea was, to slowly and methodically avoid obstacles, as accurately as possible, and when no obstacles were present, to speed up (Bebel, Howard, & Patel, 2004). This approach may seem familiar to the polder blindness effect, where the human would be more focused during times of high-environment activity, as less vigilant in low activity regions. Specifically, their model had two inputs, data from a laser to detect obstacles, and GPS location to navigate to waypoints. An

algorithm would determine the best path to proceed to the next waypoint. Their model is very small relative to others yet does the task of steering the vehicle. Interestingly, we have found no models yet that have been developed for transferring knowledge on how to drive on one side of the road to the other. This lack of transfer applies that every IAV must have a machine learning model constructed for left-hand side and right-hand side driving.

## 2.4 Artificial Neural Networks

The most basic unit of an artificial neural network is known as a Perceptron. It takes inspiration from the characteristics of the biological neuron cell. It was modelled in 1958 as an introduction to biological recognition, generalisation, recall and thinking (Rosenblatt, 1958). The perceptron's role was to 'learn' from training examples and the produce decision rules based on those examples. This was known as the perceptron convergence theorem. In this theorem, the perceptron converges to the correct classification, if data or training examples are linearly separable, and the learning rate is sufficiently small. The learning rate is comparable to the step size when moving along the error gradient. The goal of adjusting the learning rate to find the global error minimum whilst avoiding local minima. If the step size is too small then convergence may take too long to occur and if it is too big, then it may overstep the global minimum (Fang, Luo, & Tang, 2005).

The architecture of the perceptron consists of $\boldsymbol{n}$ input parameters, where each input links with a weight value $\boldsymbol{w_i}$. Each corresponding weight and input are multiplied and the sum of all is added to a bias $\boldsymbol{b}$ and passed into a threshold function to get the output value. This can also be written as:

$$z = b + \sum_{i=1}^{n} x_i w_i \tag{2.1}$$

If the threshold function is a linear threshold function, the output can be written as $y = f(z)$ where *f(z)* is the output of Eqn. (2.1). There now exist other threshold functions,

commonly known as activation functions. They have now become popular for their use within neural networks. More on this later in the sections.

The Perceptron consists of a single unit which in turn produces limited results, i.e. a single decision hyper-plane. This introduced the XOR problem where the perceptron was unable to learn the exclusive-OR (XOR) function. This problem is solved by adding hidden layers as it is not possible to solve this problem with a single perceptron. By adding hidden layers, more decision hyper-planes can be created, which allows us to solve any boolean function including the XOR problem. The trade-off must be made between the complication of the model versus the speed of convergence (Yanling, Bimin, & Zhanrong, 2002). The more hidden layers present in a model, the more computationally expensive the model.

Multilayer Perceptron (MLP), essentially a perceptron with hidden layers, has been used in the field of image recognition. Various fields such as object recognition, facial recognition, fingerprint recognition, have seen close to 100% accuracy when using MLPs. Unlike traditional machine learning algorithms, MLPs don't make any assumptions about the statistical distribution of the data, and they adapt to different input data with ease. MLPs were also shown to be less computationally expensive than other deep learning approaches, whilst providing comparable results, due to the simplicity of the algorithm (Caleanu, 2000; Lyons, Budynek, & Akamatsu, 1999; Mercimek, Gulez, & Mumcu, 2005). Object recognition is a task that involves a set of labelled training data within a certain number of classes. The goal of the applied algorithm is to take as input the image and output the class value. The more classes that are added, the higher the complexity, given the amount of overlap within those classes. This is dependent on the nature and similarity of those classes, as well as the distributions. If we take as example the ImageNet dataset, which contains 14 million images across 1000 classes, we can see that the image recognition problem gets quite complex.

The task of autonomous driving is also a very complex task as the number of possible scenarios, features to train, and computational upkeep is very high. Yet, in the early days of Artificial Neural Networks (ANN), autonomous driving was put to the test. ALVINN, a very

shallow 1-hidden layer ANN, was responsible for 45 directional outputs based on a single image input. Interestingly, their model learned to weight each input source independently of each other to identify which source, at a given time, held importance over the other. Some drawbacks of this model were the ability to learn conditions it has not been trained for. The lack of computational resources also reduced the ability of the model to train as it drives (Pomerleau, 1989). As compared to other road following models, ALVINN showed comparable performance, especially being one of the first ANNs to tackle this task. This was mainly due to the automatic feature classification of road boundaries done by the ANN. In other road following methods, roads, grass, gravel and other terrains would have to be classified by the pixel value individually before being combined. Other features, such as colours and textures would also follow similar classification techniques (Thorpe, Hebert, Kanade, & Shafer, 1988; Kuan, 1988). This increases the manpower behind such algorithms and as such does not scale appropriately.

The method of learning used by ALVINN is far from manual classification, as done by previous road following networks. ALVINN uses backpropagation to ensure that the network learns as much information from the input images, in a bid to keep the output error rate low. Backpropagation is the process of adjusting the neural network weights so that the output of network comes as close as possible to the ground truth values. More specifically, the adjustments are made to the weights at each hidden-layer (Werbos, 1990). The convergence speed of the backpropagation algorithm can be very slow due to the amount of computation that has to be carried out. However, there do exists some ways from which this process can be sped up. Selecting an appropriate learning rate will help to adjust the weights in a manner that will avoid local minima and converge faster.

As mentioned above, after the inputs are passed into the perceptron, the resulting output is then passed into an activation function. The activation function then produces an output which is used by the backpropagation algorithm to compare the difference and if needed, adjust the weights. There are multiple activation functions that can be used for different machine learning tasks. The sigmoid function was commonly used in the hidden layers of neural networks as it produces an output in range (0,1). Its popularity dropped when the saturation problem was

discovered where if the argument is either extremely positive or negative, it becomes insensitive to small changes.

In recent times, the activation function Rectified Linear Unit (ReLU) has grown in popularity. This is due to ReLU solving the vanishing gradient and exploding gradient problem. This problem is when the gradients of the hidden layers either become very small or very large and the weights do not change. The ReLU function can be defined as $f(x) = \max\{0, x\}$, where the output will be 0 if the $\boldsymbol{x}$ is less than 0, else it will be $\boldsymbol{x}$. The benefits of ReLU is that they generalise well as they preserve information as it passes across multi-layer networks (Goodfellow, Bengio, & Courville, 2016; Nair & Hinton, 2010). ReLU is also computationally cheap, allowing a model to be as efficient as possible. Recently however, ReLU has been the subject of the dying ReLU or dying neuron problem, where neurons get stuck and die as there is no gradient flow backwards. Simply, any training data point that regularly falls into the "zero negative part" would not be reactivated during the remainder of the training process. This is a known drawback because ReLUs cannot learn using gradient based approaches where the activation is zero (Agarap, 2018; Lau & Lim, 2017). To solve this problem, alternatives such as Leaky ReLU and PReLU are explored. Leaky ReLU solves this problem by allowing for a user defined alpha value for turning any negative input to a non-zero gradient. The alpha value is a parameter which may need to be experimented with to find the best fit for the task at hand. PReLU, on the other hand, requires an alpha value as well, but this alpha value is treated as a learnable parameter by the network (He, Zhang, Ren, & Sun, 2015; Maas, Hannun, & Ng, 2013). Both Leaky ReLU and PReLU solve the "dying ReLU" problem, that too at negligible computation cost. They can be expressed as:

$$LeakyReLU(x) = \begin{cases} x, & if\ x \geq 0 \\ 0.01x, & if\ x < 0 \end{cases} \tag{2.2}$$

$$PReLU(y_i) = \begin{cases} y_i, & if\ y_i > 0 \\ \alpha_i y_i, & if\ y_i \leq 0 \end{cases} \tag{2.3}$$

From Eqn. (2.3), we can observe that if $a_i = 0$ then it is a ReLU whereas if $a_i$ is small and fixed i.e. $a_i = 0.01$, then it is a Leaky ReLU.

Depending on the machine learning task at hand, computing the error of the model can be done is several ways. As our task is predicting the steering angle of an autonomous vehicle, a distance function that calculates the difference between the predicted value and actual value will work best. Mean Squared Error (MSE) is commonly used for tasks such as these. Another measure that is used is the Root Mean Squared Error (RMSE). This simply takes the square root of the MSE. These can also be expressed as:

$$L_{MSE} = \frac{1}{n}\sum_{i}(Y_i - \hat{Y}_i)^2 \tag{2.4}$$

$$L_{RMSE} = \sqrt{L_{MSE}} \tag{2.5}$$

Our goal is to ensure that the neural network is minimising the MSE in a way that ensures our model does not overfit and generalises appropriately on the validation dataset. The task of minimising the error is known as optimisation and in current neural networks gradient based optimisation techniques are common and fruitful. The method of finding the local or global minima in a search space by moving in small steps in the opposite direction of the gradient, is known as gradient descent. This method of gradient descent only updates the weights after computing the error of all samples in training. The step size in this algorithm is known as the learning rate, $\boldsymbol{\eta}$, where $\boldsymbol{\eta > 0}$. As mentioned earlier, the step size must be chosen carefully as too large or small off a step size may cause the model to never find the minima or take too long to converge. Formally, the gradient descent training rules can be written as:

$$\vec{w} \leftarrow \vec{w} + \Delta\vec{w}, \qquad where\ \Delta\vec{w} = -\eta\nabla \mathrm{E}(\vec{w}) \tag{2.6}$$

There do exist some problems with gradient descent methods. Namely, the convergence speed maybe too long and the learning rate may not guarantee the global minimum as it may get ‘stuck’ in a local minimum. Stochastic Gradient Descent (SGD) helps to solve this problem. Opposed to gradient descent, SGD updates the weights after computing the error of each sample.

## 2.5 Deep Learning

Deep learning is an extension of machine learning where any machine learning algorithm with more than 1 layer can be considered a deep learning algorithm. In the recent years deep learning has taken off in many applications such as image recognition, object detection, natural language processing etc. These fields have either achieved or surpassed human level performance.

As mentioned earlier, the solution to the XOR function was not possible with a single perceptron. Hence, once more layers were added, transforming it into a deep network, this function was solved. A deep learning approach does indeed require more computational resources as the number of calculations increases with each addition. Apart from that, deep learning networks work best with large amounts of data, especially in end-to-end approaches. This is because deep learning algorithms can filter through the noise and find hidden patterns in large amounts of data. Both these reasons can partially be attributed to the success of deep learning and its current celebrity like status. With the computation power increasing, especially through GPUs and FPGAs, and the scores of data being shared online every second, deep learning is being applied to any task within reach. In recent years, there have been many image datasets set up for the purposes of testing and benchmarking new deep learning architectures. The datasets include ImageNet, CIFAR-10, CIFAR-100, COCO etc., which have all been the testing ground for many great architectures.

Taking the ImageNet challenge as example, over the years many great algorithms have been introduced to us as they either achieved the highest accuracy or significantly decreased the number of parameters for the model. In 2012, AlexNet achieved a significantly low error rate, 15.3%, as compared to the runner-up, 26.2%. Their model was trained on 1.2 million images across 1000 classes and consisted of 60 million parameters (Krizhevsky, Sutskever, & Hinton, 2012). In 2014, this model was bested by the GoogLeNet which achieved an error of 6.67% on this dataset. Their model consisted of 4 million parameters, significantly lower than the AlexNet. Apart from the parameters, the architecture of this model was more complex too. (Szegedy, et al., 2015). However, in 2015, an error rate of 3.57% was achieved by the ResNet 152. The 152

in the name signifies the number of layers present in this model, as many variations of this model exist. The architecture of the ResNet152 model is quite sophisticated as they shift away from "plain networks" and use shortcut connections (He, Zhang, Ren, & Sun, 2016). A common network which achieved great results in the ImageNet challenge, is VGGNet. Although it did not win this challenge, it is used by many people due to its simplistic architecture which is based off the AlexNet. One of the biggest issues that surrounds this network is the high number of parameters, 138 million to be exact (Simonyan & Zisserman, 2014).

### 2.5.1 Deep Learning for Self-Driving Vehicles

Apart from image recognition challenges, there has been research undertaken with applying deep learning techniques to autonomous vehicles for road following and steering angle prediction. In 2016, NVIDIA carried out research for an end-to-end learning system using an 8-layer network, 5 convolution layers plus 3 fully connected layers. Their training dataset contains images sampled at a frequency of 10 frames per second from the video recorded on 3 cameras. They found that a higher sampling rate has images that a very closely related to each other. After their training process, they achieved an autonomy score of 98%. The autonomy metric measures the amount of time the vehicle was able to drive without being corrected by the human driver (Bojarski, et al., 2016). Their research showed the effectiveness of using Convolutional Neural Networks (CNNs) for a road following task, given just one input type, without the need for other pre-processing tasks.

Research conducted by Chen et al., 2017, used a CNN to experiment with lane keeping for self-driving cars. As compared to Bojarski et al., 2016, their model was slightly smaller with 3 convolution layers plus 1 fully connected layer. Their model does not include any sub-sampling layers as the output feature maps are already quite small. This is due to the image size being 320 x 160 px and the first convolutional layer kernel and stride being 9x9 and 4x4 respectively. Given the small model size, and the low amount of training data, they were able to achieve a Mean Absolute Error (MAE) of 2.42 (Chen & Huang, 2017). Implemented in this research was the use of dropout layers to prevent overfitting, which could lead to a low MAE score. When using small

image sizes as inputs, dropout layers have proven helpful in such scenarios. This method goes to show that experimenting with different model sizes may yield positive results and could save time and money by decreasing computational resources.

Du et al., 2017, looked at using a method which incorporates both 3D convolution layers followed by Long Short-Term Memory (LSTM) recurrent layers. A 3D convolution layer is similar to a 2D convolutional layer with the addition of a third axis, depth. This allows the model to learn some temporal information in relation to the images. Following this was a Recurrent Neural Network (RNN) with LSTM layers. The goal of an RNN is to allow the model to learn the current image based on the previous images. Based on this nature, RNNs are commonly used with time-series data which has follows some sequence. The LSTM block allows information to flow through the RNN due to the gradient vanishing problem. They used the same dataset as Chen et al., 2017, and carried out some data augmentation techniques to further generalise the dataset. Augmentations include brightness augmentation, shadow augmentation, horizontal and vertical shifts and rotational augmentation. After training their model, they achieved a validation Root Mean Squared Error (RMSE) of 0.1139 (Du, Guo, & Simpson, 2017). This research shows the effect of using a recurrent approach for predicting steering angles. It may signify that previous driving actions may have some impact on the current or future driving actions. By adding augmentation techniques, they further generalise the model making it adaptable in more situations.

Chi et al., 2017, take a temporal approach to tacking the problem of steering angle prediction using deep networks. They use a combination of convolutional and LSTM recurrent units to map images directly to steering angles. Their approach was that the predicted steering angle is determined by the current and previous angle. They construct a feature extraction method using convolution layers and LSTM blocks so that the output features have some spatio-temporal information. This was a 4 layers CNN with a fully connected layer after each CNN block. A final LSTM block to learn the temporal cues from the output feature maps. The task of steering prediction was carried out by an RNN with a LSTM unit which takes as input a sequence of feature maps extracted from their feature extractor. Using this method, they achieved a RMSE of

0.0637 (Chi & Mu, 2017). Their dataset was relatively small, containing 6 hours of driving footage, mostly highway. It did, however, include a wide range of weathers, and times.

The research conducted above utilise single input models, i.e. their model only takes in an image and outputs a steering angle based on that image. Multi input models are based on the idea of multi-modal learning, meaning the model relates information from multiple sources. In the task of autonomous driving and steering angle prediction, multi-modal models have also been utilised. These can take in an image alongside another input such as speed sensor, LIDAR sensor etc. to provide more dimensionality and learnable inputs for the model.

In the case of Yang et al., 2018, a multi-modal approach was taken wherein their model took as input an image and speed. To handle the image-based input, they implemented 5 convolution layers with 2 fully connected layers. To handle the speed input, they implemented a recurrent LSTM block with a sequence length of 10 timestamps. This is then followed by 2 fully connected layers. The image output of the CNN block is used to predict steering angle and is also concatenated with the output of the LSTM block to predict speed output. In their case, the speed is not used to predict steering angle, they too only rely on images for steering angle. Their model achieved low MAE for both steering angle prediction as well as speed prediction (Yang, Zhang, Yu, Cai, & Luo, 2018). This research not only reinforces the use of images in steering angle prediction, but also how images can also aid in predicting speed.

Chowdhuri et al., 2019, took the approach of an imitation learning based multi-modal CNN model. Imitation learning is the process of learning, sequentially, from the actions of a human domain expert. The issue of covariate shift arises in imitation learning approaches where a trained model is presented a situation not yet trained for. To counter this, the researchers added correctional data through simulations, not using data augmentation like the others. Their model is a simple 2-layer CNN followed by 2 fully connected layers. Each of their convolution layers are followed by a subsampling layer, to reduce dimensionality, and a normalisation layer. Their model achieved a validation MSE of 8.16% (Chowdhuri, Pankaj, & Zipser, 2019). As their

dataset was based on a scaled representation of actual vehicles in different environments, a large network, as used by other researchers, may not be required.

## 2.6 Convolutional Neural Network

Past literature mentions the use of convolutional neural networks for their projects. Here we take a closer look at how they work, layers present and their impact on the network, in our research. In 1989, Convolutional Neural Networks (CNNs), were brought forward by LeCun et al., which applied the backpropagation algorithm on handwritten digits, specifically for zip code recognition. They found that if constraints from the task domain are integrated into the backpropagation network, the ability of neural networks to generalise will be enhanced greatly (LeCun, et al., 1898). Since the development of methods to extract local features and map them to higher-order features had been experimented with. They found that the location of the feature on the plain does not matter towards the classification. This is a property of CNNs known as location invariance. Instead, having some feature detectors which could identify certain features anywhere on the plain matters more for classification. However, using this method not all spatial information is lost, instead the higher-order feature maps store approximate features. Since our task of training a multi-input model end-to-end is heavily image processing based, CNNs are the go-to approach, as done by other researchers in this field.

In a CNN, apart from the layers discussed below, there is an Input layer and an Output layer. Depending on the type of CNN, the input layer can contain either a one-dimensional, two-dimensional or three-dimensional array that represents an image. In our case, this will be a two-dimensional image with 3 channels that represent Red, Green, Blue (RGB) pixel values. The output layer of a network contains the same number of nodes as the number of class labels. In our case, we have 1 output node as we have a task of predicting a steering angle. These nodes are fully connected to the previous nodes of the dense, fully-connected, network. It also includes an activation function that will output the required value within a certain range. In our case, we will require 1 node in out output layer with the tanh activation function to output a value between -1 and +1.

### 2.6.1 Convolution Layer

The convolutional layer is responsible for extracting features from inputs. This is done by training filters, or kernels, on a training dataset. Kernels are responsible for learning a set of weights that are activated when the kernel comes across a similar feature. As the kernels scan the whole image, they learn to filter over spatial information that is not relevant. The number of kernels is defined by the user and are a parameter that can be tuned. Consideration must be taken, when selecting the number of kernels, as too few may lead to some features not being learnt and too many may overfit on the noise within the training dataset. Apart from kernels, the user can define the stride, kernel size and padding. The stride parameter allows the user to set the step size of the kernel to scan the whole image. Kernels learn features by moving across the image and choosing a step size impacts the size of output feature map. If the step size is too large, some spatial information could be skipped over. The kernel size is a value $m$ x $n$, which is responsible for the size of all the kernels in the specified layer. A kernel size of 3 x 3 would mean that there are 9 total trainable weights inside the filter. It contains a set of randomly initialised weights that map to the input. The size of the kernel should be chosen carefully as the larger size may not be generalised. In some past literature for steering angle prediction, initial kernel sizes were ranged from 3 to 11, whereas, in image recognition tasks kernel size is usually 3. The kernel size also has an impact on the output shape of the feature map. A larger kernel size will result in a smaller output feature map. Not only that, the larger the kernel the more complex feature it will learn. As such, the smaller the kernel size, the learnt features will be more generalised. Increasing the kernel size may also require an increase in the number of epochs to ensure the model is learning complex features. On the other hand, smaller sized kernels are prone to overfitting as initialising many kernels may mean the noise is being learnt. Padding can be used to deal with edge cases and keep the same output shape. There are multiple ways to pad such as zero-padding, wrapping etc. Each have some benefit, but it is usually left up to the experimentation process of domain knowledge. To calculate the output shape, we can use the formula in Eqn. (2.7). Here, $W$ is the input shape, $K$ is the kernel size, $P$ is the padding, $S$ is the stride. An example convolution process can be seen in *Figure 2*.

$$OutputShape = \frac{W - K + 2P}{S} + 1 \tag{2.7}$$

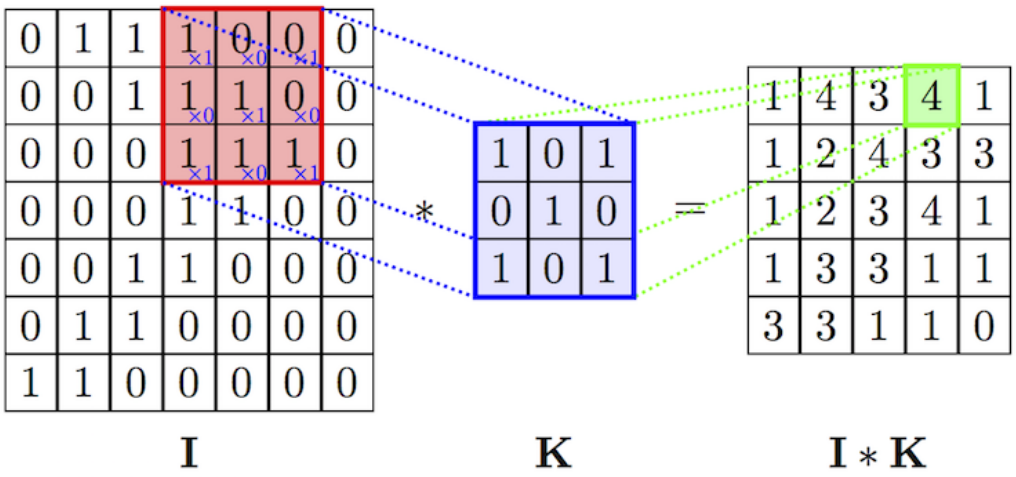


Figure 2- The convolution process taken from *(Prakash, 2018).*

## 2.6.2 Pooling Layer

Once the output of the convolution layer is activated, through an activation function, it can be sent to a pooling layer. Pooling layers were commonly bundled with convolution layers, but it is possible to construct a network without these. The pooling function also has a kernel like window, with a user defined size, that slides across the input shape. Simply put, a pooling function computes an output for the inputs present in the kernel like window. The difference between the kernel of the convolution layer and pooling layer is that the weights in the pooling kernel remain fixed. Unlike the convolutional kernel where the weights are a trainable parameter. The stride parameter here can also be adjusted by the user. It serves the same function for defining how much step size the kernel window should take to scan the image. There are many types of pooling layers, such as max pooling, global average pooling, $L^2$ normalisation etc. As the name states, the max pooling function computes the maximum value from the input window and adds it in the output feature map. Global average pooling computes the average of the rectangular window and adds it in the output feature map. $L^2$ normalisation computes the root of the sum of the square of the values inside the rectangular window and replaces it in the output. The other benefit of pooling layers is to reduce dimensionality and making it easier to create larger networks. As there exist many pooling methods, Bourean et al., has provided some theoretical

input on which works best for visual recognition algorithms (Boureau, Ponce, & LeCun, 2010). They saw max pooling performed well theoretically and this could be used as a starting point for many projects. However, practically, applying and evaluating different methods may provide other results. An example of a max pooling function, with stride 2 and a 2 x 2 kernel, can be seen in Figure *3*. As shown, max pooling will reduce the output feature space.

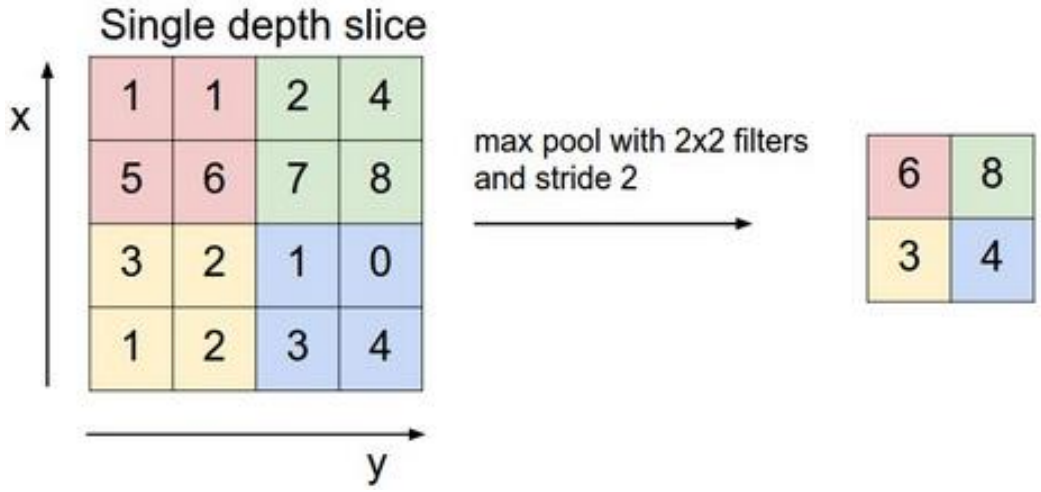


Figure 3 - The sampling process using max pooling taken from *(Ricco, 2017)*.

## 2.6.3 Dense Layer

Dense layers, also known as fully connected layers, are multi-layered ANNs that follow a CNN block, in the case of image recognition. The role of these layers is to transform the feature maps output by the CNN and perform classification or regression tasks. The convolution layers themselves only output put a feature map of all significant features inside the training dataset. The classification and regression tasks are dedicated for the fully-connected network. Dense layers consist of a number of neurons, defined by the user, which contain weights, a trainable parameter. During backpropagation, weights of this network are adjusted to find the best fit. When creating this network, the user must keep in mind the number of hidden layers and the number of neurons in each layer, as this may cause overfitting or underfitting. Apart from that, adding more layers or neurons in each layer increases the computational requirements for the network. Following each layer is an activation function which must also be chosen by the user. In our network, the responsibility of predicting the steering angle falls on this fully connected network, as the CNN is only responsible for extracting relevant features from the input images.

### 2.6.4 Dropout Layer

Overfitting is a common issue in deep neural networks as they contain a high number of parameters. Dropout layers are a computationally inexpensive method of preventing overfitting, as compared to other regularisers such as weight decay. In terms of complexity, dropout offers a cost of *O(n)* per example during training. The way that dropout works is it creates sub-networks of all non-output units and constructing an ensemble of subsets of the original sub-network. Then the ensemble is trained, and the best subset is moved forward (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). From this we can see that dropout is similar to a bagging approach, but not quite. Bagging is also an ensemble technique where multiple models are created, and the best model is selected. Bagging assumes that all models are independent whereas in dropout, the models are a subset of a sub-network, they have shared parameters. Also, in dropout, training all the possible combinations of sub-networks is impossible, for large neural networks, therefore, sub-networks are chosen at random and trained. The dropout rate is a tuneable hyperparameter, which specifies the probability of retaining a unit in the network. It is suggested that the dropout rate between 0.4 and 0.8 works best (Goodfellow, Bengio, & Courville, 2016; Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014).

## 2.7 Transfer Learning

Traditional machine learning models work under the assumption that training, test and production data are from the same space. For instance, if a model is trained to classify an input image as either a cat or dog, it will assume all input images will either contain a cat or dog and classify appropriately. If the user wished to add some learning to classify a lion alongside the cat and dog model, the model would need to be trained from scratch. This causes problems for real-world scenarios where collecting labelled data for a certain task is difficult, i.e. healthcare data. In cases such as these transfer learning between domains can be leveraged (Pan & Yang, 2009).

Transfer learning is the process of using a model that has been taught a task and using it for another similar task. A benefit for using transfer learning can be to improve generalisation in the

transferred setting. Transfer learning may also help us arrive at a better solution faster, or it may provide a better solution than training a new model. The motivation behind transfer learning is the fact that humans can apply knowledge from one task to another regularly. This can be advantageous especially when there exists less labelled data for the target task. This is because, we can assume that the factors in task 1, maybe somewhat similar to the factors in task 2. As we begin to examine feature maps of the trained model, features near the input maybe too input specific. Features closer to the output are more generalised and maybe worth sharing those layers instead. This would mean that we are not training the target task from scratch, instead sharing features from a source task. This method of transfer learning can be known as Hierarchical Transfer, where solutions for simple tasks are combined to learn more complex tasks. More specifically, by only transferring general solutions, such as lines, curves, edges from a source task, we can learn objects in a target task (Torrey & Shavlik, 2010).

The similarity between source task and target task is very important. By ensuring that the source task and target task share some similar domain, we can make assumptions on the compatibility of shared features. This will also give us an idea on how successful transfer learning will be to our applied task. In *Figure 4* we can see how different tasks are placed in relation to each other based on how effectively they support neighbouring tasks (Zamir, et al., 2018). The more connected they are the more significant their role in transferring to other tasks.

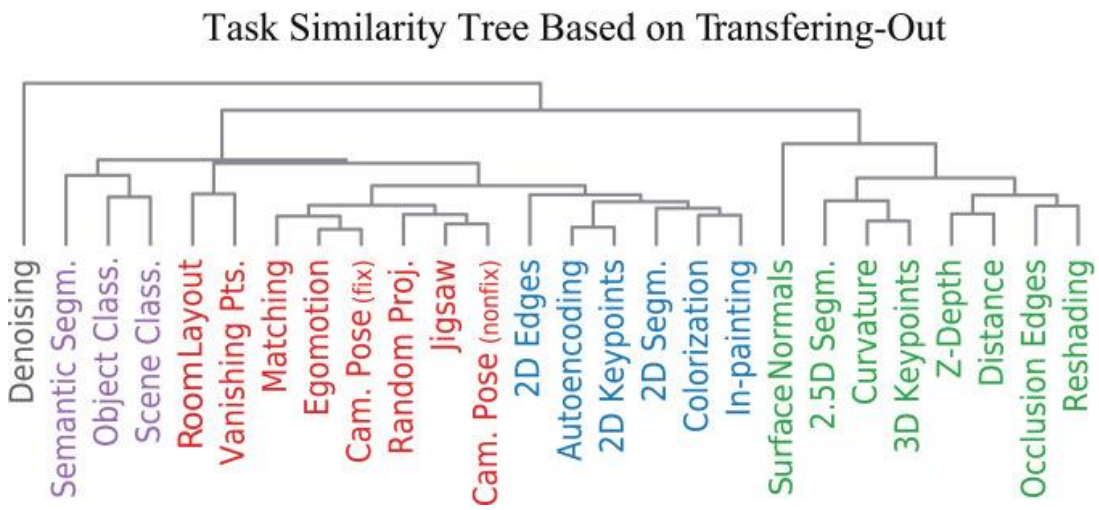


Figure 4- Task similarity tree taken from Zamir et al., 2018.

### 2.7.1 Transfer Learning for Self-Driving Cars

Transfer learning has been used in the field of self-driving cars to predict the steering angles on scenarios such as highway driving. In this sense, knowledge was transferred from object detection models to steering angle prediction. This is explained more below on why this method should not be as effective as creating a machine learning model specific to steering angle prediction and transferring said model. Some of the common architectures used for this task is the ResNet models and its variants. This would simply involve the researcher pre-processing their image data and freezing the weights in the layers they seem fit. In the case of Maqueda et al., 2018, event-based inputs were used instead of raw RGB images. The event camera captures the stream of intensity changes at the time they occur. They apply this input to the ResNet18 and ResNet50 models, they attach a 256-dimensional and 1024-dimensional fully-connected layer, respectively, before the one-dimensional output. Both the ResNet models have been trained on the ImageNet dataset and thus contain features for RGB images. They found RGB features positively transfer to event-based features, which resulted in a low RMSE score. They also found that using the weights trained on the ImageNet dataset results in a lower loss as compared to random initialisations. The ResNet50 seemed to perform must better than the ResNet18 and achieving a RMSE score as low as 2.33 on the day-time driving event dataset (Maqueda, Loquercio, Gallego, Garc'ia, & Scaramuzza, 2018). As other studies used RGB images, this shows the value of using other format images as input images. Including different time of days, as they did, also helps with generalising the model, resulting in better performance.

Du et al., 2017, also used a ResNet50 model to identify if that is useable for a dataset of RGB images. This ResNet50 model contains pretrained weights from the ImageNet dataset as well. As a pre-processing step for the model, they froze the weights of the first 45 layers of the ResNet model whilst the remaining layers were allowed to train. Attached after the ResNet layers were 3 fully-connected layers and a final output layer. Using this method, they achieved a validation RMSE of 0.0775 (Du, Guo, & Simpson, 2017). They found that the ResNet50 model showed signs of over-fitting, which it is known for. Freezing the top 45 layers of the model also proved to be an effective method of producing results. This strategy was taken due to the inputs

being of similar domain, and thus would have some similar high-level features. The attached fully-connected network was much more complex than others, possibly due to them having more data during training time.

Eraqi et al., 2017, carried out a steering angle prediction task using RGB images as well and compared it with a simple CNN model trained end-to-end. The models used for transfer learning were the ResNet152 and InceptionV3, both trained on the ImageNet dataset. They found that the deeper models performed the best as the ResNet152 achieved a RMSE of 17.77 (Eraqi, Moustafa, & Honer, 2017). The weights for all layers were frozen and there were no extra layers attached before the output. This could explain the, relatively, high RMSE score along with the fact that the ResNet and Inception models have a different source domain. Not only that but the weights transferred were trained on the ImageNet dataset which also shares a different domain. Adding extra layers or freezing only some layers may have resulted in a lower RMSE.

Kim et al., 2017, looked at using transfer learning methods of ego lane estimation. Their view is that for self-driving cars to be truly intelligent in the task of driving, semantic segmentation is essential. Semantic segmentation is the process of segmenting objects in a scene based on their semantic and relational properties. They provide a method to utilise semantic information for ego lane estimation. For this task, a SegNet, which was trained on segmenting a driving scene into 12 classes (Badrinarayanan, Kendall, & Cipolla, 2017), was transferred to segmenting left and right ego lanes. These ego lanes would be useful for understanding if in the absence of lanes or partially visible lanes, a vehicle can still stay on the path. Their method gave good results when segmenting both left and right ego lanes (Kim & Park, 2017). Ensuring the model can detect lanes is an important aspect of end-to-end learning. When exploring the filters, the model should train some lane marking detection filters to base the output steering angle on. Initial studies showed through feature maps that outside areas of roads were activated and sometimes the lane markings. Without explicit segmentation, the CNN model should be learning these as, for us humans, they are an important part of steering.

## 2.8 Summary

By examining the past and current literature, we can see the transition of the self-driving vehicles. What initially started as hard-coded rules for detecting roads, grass, curves etc. can now be done with state of the art CNNs. We also identified that some methods carry limitations which are impractical for the real world, i.e. model cannot work if vehicle is travelling faster than a limit. Currently deep neural networks have shown to improve performance and accuracy significantly. This is mainly due to the computational hardware available and the amount of data we currently possess. Although limitations of these networks are that they need new data if one wanted to use them for another purpose. For this task, transfer learning was found to provide some benefit.

Based on these limitations and gaps in current literature we generated some research questions which may provide input into this field. We identify is a model can be trained to predict the steering angle on simulated data. Then, can this model be transferred from a left-hand drive environment to a right-hand drive environment. Our research addresses the lack of work done on using simulated datasets for steering angle prediction, as well as the ability of self-driving cars to be shipped to a region they are not trained in. We now design a workflow which aims to best tackle these research questions.

# Chapter 3
# Methodology

*In this chapter, we describe the tools used to create our environment and the workflow that our research follows. The steps taken for scenario creation, data collection, data aggregation is outlined as a data collection step. Data pre-processing, model selection and evaluation are also discussed and analysed. Also, discussed are the steps taken for creating, testing and evaluating experiments for all our research questions.*

## 3.1 Environment Design

For this task, Convolutional Neural Networks (CNNs), are proven to work best with image-based inputs. Since the introduction of the CNN by LeCun et al. they have been widely used in the field of pattern recognition in images (LeCun Y. a., 1998). As we will be focusing on multi-inputs, other data will be concatenated with the output of our CNN block and fed into a multi-layered Artificial Neural Network (ANN). To build our model we will be using the Python programming language, and deep learning frameworks like Keras and Tensorflow. Other frameworks such as OpenCV, Pandas, NumPy will be used for manipulating images, csv data and image vectors respectively.

Prior to the model building process, data collection is carried out within the parameters of our task. The major tools used were RoadRunner, a software to create 3-D road networks and CARLA, an open-source autonomous driving simulator. Using RoadRunner, we could perfectly create our driving scenario, including the minor details, such as type of road, road signs or lane markings. By loading this map into CARLA, we can programmatically, using Python, run our data collection simulations. CARLA gives us greater control in regard to driving conditions, vehicle selection, type of sensor, lighting etc. Using the combination of these tools our initial dataset can be made.

Our models will be trained on a GPU by configuring our Tensorflow backend as per NVIDIA guidelines. The GPU available for this research is the NVIDIA GTX 1060 6GB. The reason for using a GPU is due to the speed of vector calculations performed. On my hardware, the GPU is 60-100x faster than my CPU for image-based machine learning tasks. The step of data pre-processing is done using the CPU as it avoids excess costs, if scaled up to large scale projects.

Our research follows a linear methodology, which can be seen in *Figure 5*. This methodology includes the process to answering all our research questions. The left-hand branch aims to achieve our goal of making a CNN for steering angle prediction. The dotted line for variable sampling aims to achieve our research question two. Finally, the right-hand branch aims

to achieve our goal of testing if a left-hand drive model can be transferred to right-hand drive data. To carry out our research and achieve our objective, we started off with identifying and creating a research environment of our autonomous driving vehicle. The environment includes a driving scenario, in our case a single left bend as a bending road is common in New Zealand. This is because testing our research questions on all scenarios that drivers face will be endless. To create the environment two tools were required, RoadRunner by VectorZero and CARLA by Dosovitskiy et al., 2017 (Dosovitskiy, Ros, Codevilla, Lopez, & Koltun, 2017).

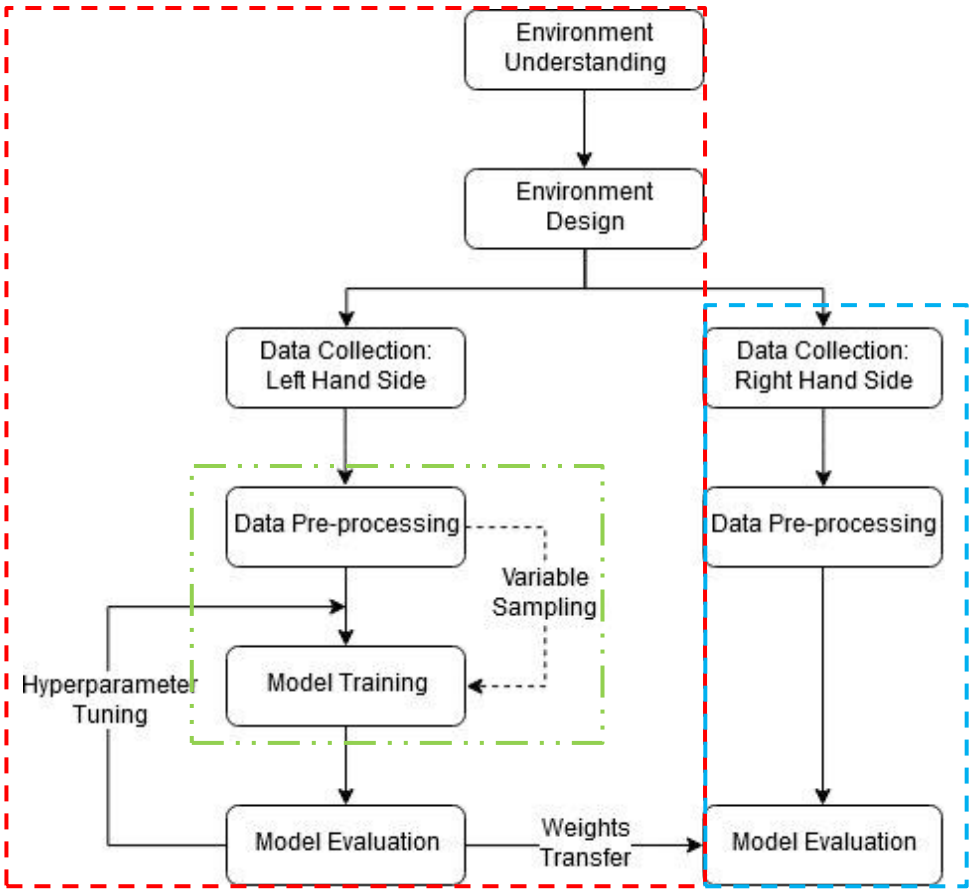


Figure 5- Flow of the research methodology which goes over all our research questions. Red annotation is the workflow of our research question 1, green annotation is for research question 2, blue annotation is for research question 3.

RoadRunner is an advanced road network and environment generating software. It is useful for creating complex road networks that can be fit for any environment and many adjustable parameters. RoadRunner is a paid software but VectorZero offer a two week free-trail for academic licenses which we leveraged. They also offer an in-depth prop kit which includes many different road signs, markings etc., but that comes at a price. During this period, we created our

left bend scenario, with adjustments such as road marking type, driving direction, and type of road. Multiple versions of this network were created and exported in the appropriate format for CARLA.

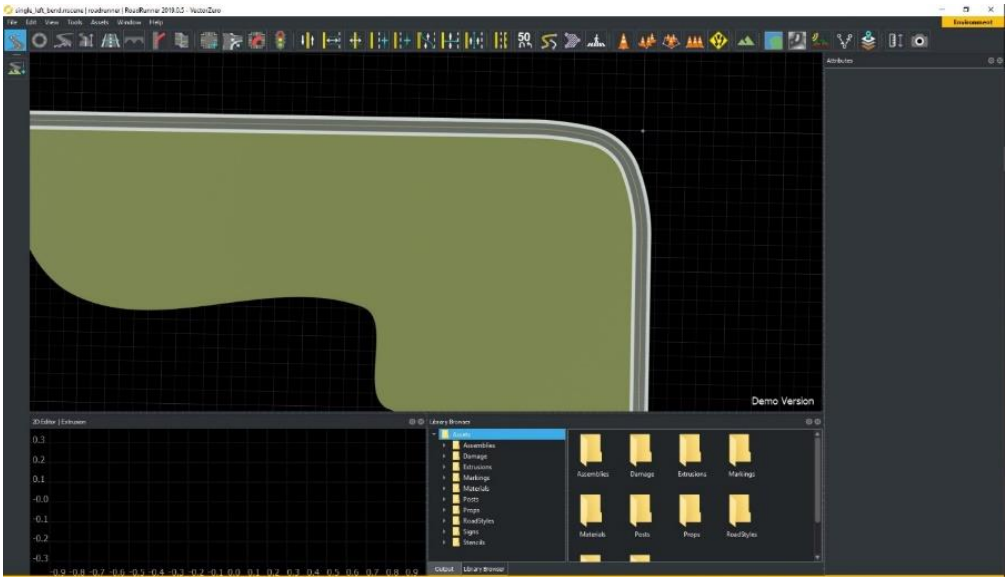


Figure 6- Figure showing the user interface of RoadRunner

CARLA is an open source autonomous driving simulator which works on top of Unreal Engine, developed by Epic Games. Unreal Engine is a graphics development engine which makes it possible to create and interact with CARLA and any custom maps that we need to load. CARLA is originally written in the programming language C++, but they provide wrappers for Python 3.x for seamless programmatic communication. CARLA features many resources and assets for making a realistic driving environment, including road signs, sensors, vehicles, natural environments, weathers etc, which we can leverage free of cost. After loading the map in CARLA, we added some props such as houses, railings, road signs, etc., and chose a vehicle, Audi TT, that will best serve our purpose. We then attached an RGB sensor and a speed sensor on our vehicle, to capture the footage of the car and the speed at any given timestamp. We also provide the RGB sensor with the desired resolution and the frame rate which will be beneficial for the data collection step.

## 3.2 Data Collection

Data collection is, arguably, the most crucial step in ensuring we get reliable results as output. There exist many real-world driving datasets containing 100s of hours of driving footage

including steering angles, GPS data, speed and other sensors. Some limitations of these datasets are they, individually, possess imbalanced data, lack of corrected actions, weather scenarios etc. Although having a combination of these datasets solves this problem, but they introduce compatibility problems. For our purpose, a combination of the datasets would need to be used and because of that it makes sense to make use of a simulator which we can adjust as per our needs.

Based on this, we decided that our data collection step requires the use of CARLA. With CARLA we can manipulate the environment as we wish; removing the need for synthetic data augmentation. Initially, we collect two training and validation datasets for both left-hand side and right-hand side driving. As mentioned earlier, we implement an RGB sensor, a steering angle sensor and a speed sensor to our vehicle. To collect this data, we created multiple routes inside CARLA on the same map and initially had autopilot drive the routes. Most of the current real-world driving datasets also have a lack of corrected driving, where the driver must correct the positioning of the car if it is out of line. This data is very important for ensuring our model learns those situations and be more generalised. To add some corrected driving data, a human also controlled the car using the keyboard. As the human was not used to the CARLA control environment, a lot of mistakes were made which were then corrected whilst being recorded. Now, there are some limitations with this approach. The human controlling the car knew this was a simulation and therefore was too relaxed when going through the process. This may have had some bias on the corrected data that we collected. Some limitation might be the fact the human is not scared of making mistakes in the simulation. This would present some probability of the reactions not being as urgent as if the human was actually present in the situation. Although in a real situation it is not guaranteed that the human wont freeze. As shown by Morton et al., 2013, driving simulators do have some impact on the user driving, if the user is under stress (Morton & White, 2013). This would reinforce our limitation of the corrected driving not being as effective as the user was not under stress apart from the fact, they did not know how to use the environment. However, the corrected driving added in our dataset is much more raw, cleaner and generalised than that collected through data augmentations. Apart from that, the time of day and

the weather was also changed to add some generalisation. We looked at using dawn, afternoon and dusk along with clear, cloudy and slight shower weathers.

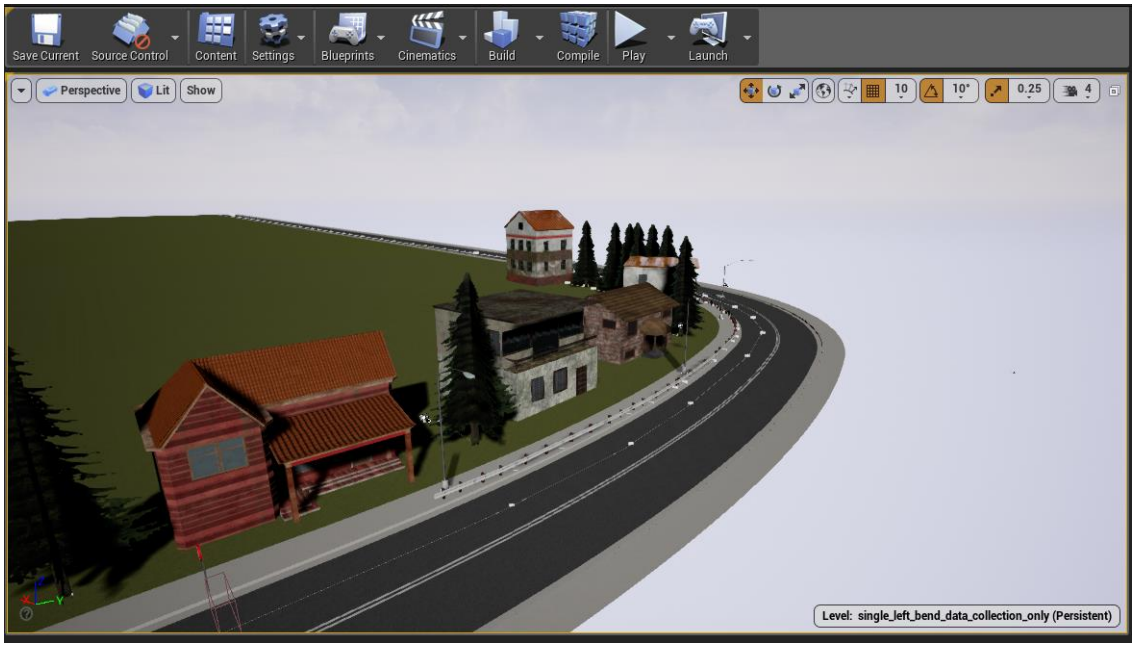


Figure 7- Image of the CARLA environment in Unreal Engine 4

A common problem that arises in real-world datasets is imbalance between straight roads and turns. Without correcting this imbalance, our data will be spread unevenly. This will cause the model to have a bias to always go straight. Literature solves this issue through augmenting images containing bends and creating synthetic data samples. In CARLA, we can create scenarios that will run multiple turning sessions to allow us to gather enough bend data. This allows us to skip any un-needed data augmentation and gives us greater flexibility on collecting more generalised data.

Once this process was finished, we had a total of approx. 52 minutes of left-hand side drive training and validation data and approx. 11 minutes of right-hand side drive training and validation data. Quantitatively, we had approximately 32,000 left-hand driving images and 7,000 right-hand driving images. The images were kept in a separate directory of their own, the names of the images, associated steering angles and speed were kept in a CSV file. It is important to note that the data in our CSV file was ordered sequentially so that it can be useful for answering our second research question.

During our literature review we found that the image size varied but was still generally small, approximately 300px * 200px. Due to this we decided to keep our image size at 300px * 250px with 3 RGB channels, i.e. (300 x 250 x 3). RGB images were used as we believe colour is very useful for determining steering angles, especially for interpreting road signs, lane marking etc. The sampling rate was also put into question with 10 FPS being quite common. Anything greater than that would result in similar images and lower will show drastic changes between the previous and current frames.

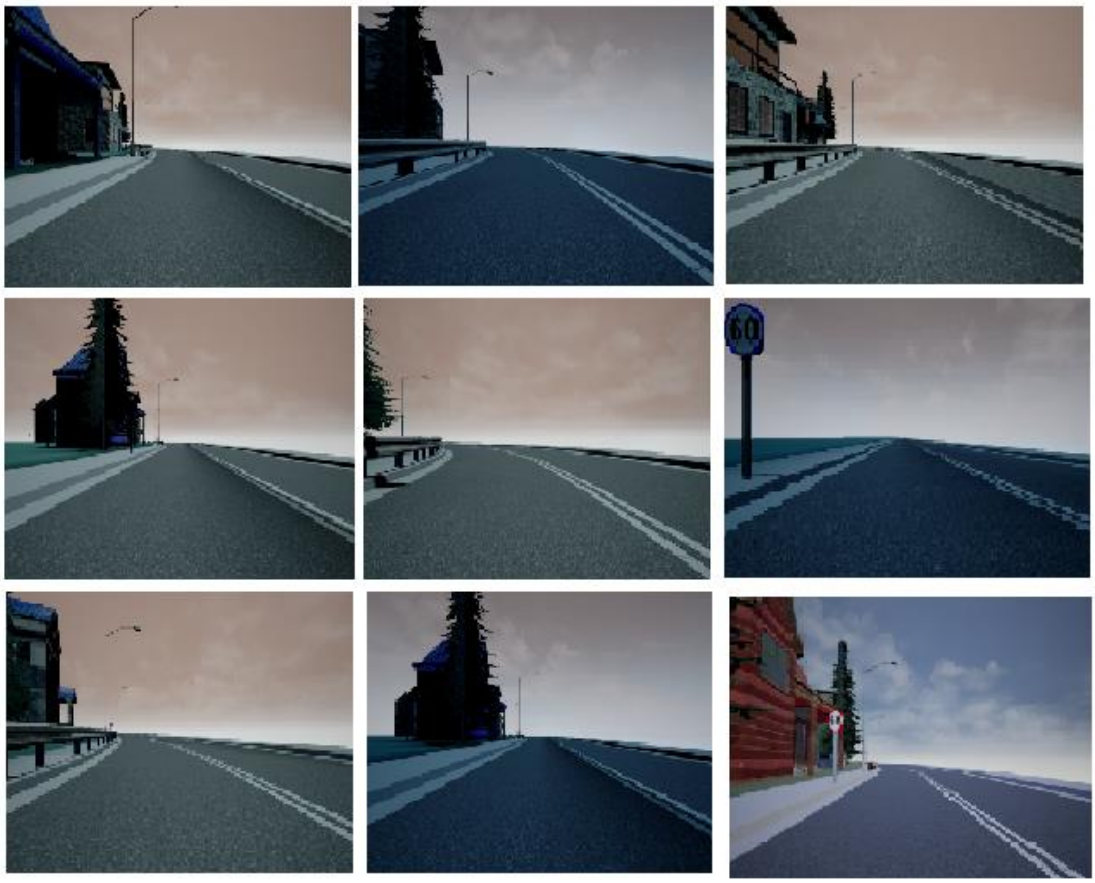


Figure 8 - Random subset of images from our training dataset.

## 3.3 Model Design

For this research, we design a Convolution Neural Network which takes inspiration from the networks used in current literature. Initially, we will look at their 3-layer and 5-layer networks to identify how they perform on simulated datasets. We also construct a 7-layer plus 3 fully-

connected layers to compare the results. The goal using creating and testing these architectures is to identify which can produce the lowest MSE for steering angle prediction.

Apart from predicting steering angles given an image input, the model will also be training filters. Steering angle prediction falls under the umbrella of image processing. These filters will provide insight into the features being learnt by the model from the images. To create rich feature maps, we use 7 convolution layers in our model, with 4 alternating pooling layers. The pooling layers help with reducing the number of parameters and summarise redundant spatial features from the output feature maps. We keep the stride size for both the convolution layers and pooling layers at 1 as to not lost any spatial information. We make use of some padding to help ensure any edge case scenario does not get overlooked. A flattened vector is taken as input by an ANN consisting of 3 fully connected layers. A final output node uses the activation tanh to output the steering angle. To measure the success of our model, we aim to reduce the MSE as much as possible.

The model created above will then be transferred to tackle our third research question. Can a model, created for left-hand side driving, work with right-hand side driving? During transfer, we will experiment with freezing weights in specific layers. Evidence shows that freezing specific layers may output better results as compared to freezing all weights. The success of this research question will also be a low MSE.

All models in this thesis are written in Python 3.6 and created using Keras and Tensorflow. Keras is a high-level API that uses Tensorflow as its backend, allowing us to rapidly create models whilst giving us the flexibility to modify them deeply. Tensorflow also integrates well with CUDA allowing us to greatly speed up our computations.

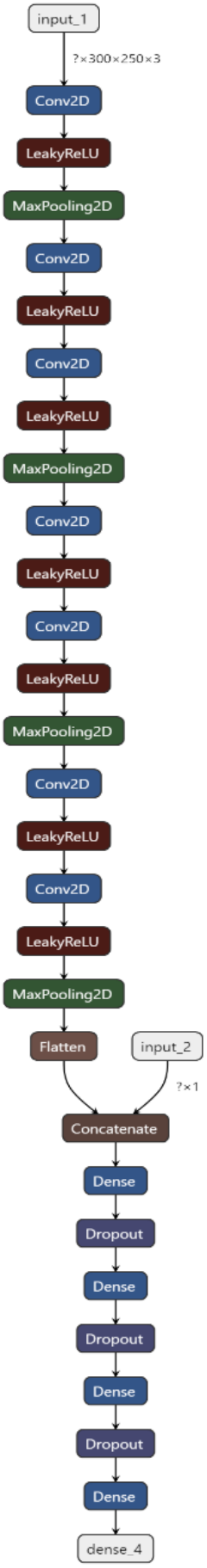


Figure 9- The overall architecture of our CNN

Following the overview of the structure of our algorithm, we present 7 convolution layers. The input to these layers is an array in the form of [*B, H, W, C*]. The *B* represents a value for the batch size to our layer. Both *H and W* represent the size of the input image, which in our case is 300 x 250. The *C* here represents the number of channels in our images, which in our case is 3 for all RGB layers. The convolutional operation is a linear operation which needs to be passed through a non-linear activation function. Initially, we introduced ReLU, LeakyReLU and PReLU, as these have been known to solve the vanishing gradient problem. ReLU introduces the "dying ReLU" problem, solved by the other ReLU variants. For our network LeakyReLU is used as it provided the most value. LeakyReLU also takes in an alpha value argument, which is also a trainable parameter.

For the sub-sampling step, we utilise max pooling. Max pooling takes in as input a kernel size *h x w*, and selects the highest, max, value from that kernel. The output is a reduced feature space with any non-important features discarded. Another common sub-sampling method is Global Average Pooling, which takes the average of all values in a kernel. Our literature shows that in the case of end-to-end autonomous driving problems, max pooling works best.

Following the convolutional layers, a network of 3 fully-connected layers is responsible for predicting the steering angle. It takes as input a flattened one-dimensional vector and finds relationships as a standard multilayer perceptron would. There is also a second input that gets added, which in our case is the speed. Adding this second input allows us to generalise the model and allow for more sensor-based inputs in the future. In each of these layers the number of neurons is adjustable, and we use the tanh activation after each layer. Finally, a final layer containing one neuron is connected which is responsible for the output steering angle. This layer also uses a 'tanh' activation functions as our steering angle is in the range of [-1, +1]. The predicted output of the model is compared to the actual steering angle using the MSE metric. The goal of the model is to achieve a low MSE. Our loss is measured using the MSE to reduce the loss per epoch.

Table 1- CNN layers, shape and parameters used.

| | Layer | Shape | Parameters |
|---|---|---|---|
| 1 | Conv2D | (Batch Size, 300, 250, 256) | 256, 3, 3, 1 |
| 2 | LeakyReLU | | 0.33 |
| 3 | MaxPooling2D | (Batch Size, 150, 125, 256) | 2, 2 |
| 4 | Conv2D | (Batch Size, 150, 125, 128) | 128, 3, 3, 1 |
| 5 | LeakyReLU | | 0.33 |
| 6 | Conv2D | (Batch Size, 150, 125, 64) | 64, 3, 3, 1 |
| 7 | LeakyReLU | | 0.33 |
| 8 | MaxPooling2D | (Batch Size, 75, 62, 64) | 2, 2 |
| 9 | Conv2D | (Batch Size, 73, 60, 32) | 32, 3, 3, 1 |
| 10 | LeakyReLU | | 0.33 |
| 11 | Conv2D | (Batch Size, 71, 58, 32) | 32, 3, 3, 1 |
| 12 | LeakyReLU | | 0.33 |
| 13 | MaxPooling2D | (Batch Size, 35, 29, 32) | 2, 2 |
| 14 | Conv2D | (Batch Size, 33, 27, 32) | 32, 3, 3, 1 |
| 15 | LeakyReLU | | 0.33 |
| 16 | Conv2D | (Batch Size, 31, 25, 32) | 32, 3, 3, 1 |
| 17 | LeakyReLU | | 0.33 |
| 18 | MaxPooling2D | (Batch Size, 15, 12, 32) | 2, 2 |
| 19 | Flatten | (Batch Size, 5760) | |
| 20 | InputLayer | (Batch Size, 1) | |
| 21 | Concatenate | (Batch Size, 5761) | |
| 22 | Dense | (Batch Size, 1164) | 1164, tanh |
| 23 | Dropout | | 0.5 |
| 24 | Dense | (Batch Size, 500) | 500, tanh |
| 25 | Dropout | | 0.5 |
| 26 | Dense | (Batch Size, 200) | 200, tanh |
| 27 | Dropout | | 0.5 |
| 28 | Dense | (Batch Size, 1) | 1, tanh |

Following the convolution layers, a network of 3 fully-connected layers is responsible for predicting the steering angle. It takes as input a flattened one-dimensional vector and finds relationships as a standard multilayer perceptron would. There is also a second input that gets added, which in our case is the speed. Adding this second input allows us to generalise the model and allow for more sensor-based inputs in the future. In each of these layers the number of neurons is adjustable, and we use the tanh activation after each layer. Finally, a final layer containing one neuron is connected which is responsible for the output steering angle. This layer also uses a ‘tanh’ activation functions as our steering angle is in the range of [-1, +1]. The

predicted output of the model is compared to the actual steering angle using the MSE metric. The goal of the model is to achieve a low MSE. Our loss is measured using the MSE to reduce the loss per epoch.

Between these dense layers, we add dropout layers to help prevent overfitting. We kept our dropout rate high, at 0.5 as evidence showed this to be within a good range (Goodfellow, Bengio, & Courville, 2016; Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). We chose to use dropout layers as our model would have a lot of parameters and using a dropout with a high value is common among literature.

Prior to the training process, we normalise the image data and the speed input, within a range of -1 and 1. This is because of our steering angle data is recorded within this range. Data normalisation is not always required but is essential when the scale of values is different between features. For example, our image pixel values are within a range [0, 255]. Whereas, our speed is stored as a value in meters per second. The goal of data normalisation is to rescale the features, so they fit on a common scale. There exist many methods of normalisation with Batch Normalisation being commonly used within literature (Ioffe & Szegedy, 2015). We employ a min-max scaling, which is a simple method for scaling features into a specified range, in our case [-1, 1]. This method of normalising is simply a linear transformation. It takes in an input *x* and returns a normalised value $x'$ . The method used can be seen in Eqn. (3.1).

$$x' = 2\,\frac{x - \min(x)}{\max(x) - \min(x)} - 1 \tag{3.1}$$

Once this model has been trained, we can save the trained weights for each layer to disk. This will allow us to load model for the purposes of transfer learning. From *Table 1*, we can the description, and parameters of all the layers in our final model. When it comes to transfer learning, usually all the layers until the last layer, layer 28, are frozen, and a custom output layer for the specific task is attached. Freezing layers is a practise in which evidence suggests that some layers maybe too domain, or task specific to be beneficial for transfer. If we take as example

Du et al., 2017, they froze the first 45 layers in their ResNet50 model as it yielded good results. They also tried multiple different combinations of layer freezing to arrive at their current conclusion.

We looked at experimenting with layer freezing, including freezing all layers and freezing a combination of the convolutional blocks. In our transfer learning model, freezing layers 9 and 10 from *Table 1*, yielded the best MSE. Freezing all the layers or freezing a combination of the middle layers resulted in a relatively large MSE. As our task was of predicting the steering angle, it was possible to use the output layer from the original model. The optimizer, Stochastic Gradient Decent (SGD), and the loss function, MSE, for this task also remained the same. Some tuning was done for SGD to find the appropriate learning rate and momentum.

To summarise, our methodology consists of initially training our own steering angle prediction model. This model will be trained on a left-hand drive dataset consisting of approximately 32,000 images, or 50 driving minutes. We validate our model on a subset of the training dataset. We also examine the trained weights of our model and visualise the features being learnt. After this, we experiment with freezing the weights from a combination of layers until we find the most combination, the middle layers. The optimal combination is found by training and validating the model on a right-hand drive dataset and reducing the MSE. We then explore and visualise the weights from the transferred model to identify the features it has learnt.

# Chapter 4

# Results

*This chapter goes over our results achieved from experimenting with our neural network model. We critically analyse each research question, the model used, the process of training and the impact of parameters as a whole. We also critically analyse the limitations that may have affected our experiments.*

## 4.1 Model Parameters

To ensure our model achieves a good Mean Squared Error (MSE), we need to continuously optimise the hyperparameters. The goal of hyperparameter tuning is to find the balance between a low error metric and computational expense. The issue of overfitting and underfitting must also be taken into consideration. Our model is created in Python 3.6 using the Keras and Tensorflow deep learning libraries. These allow us to set and tune many hyperparameters with ease. In this section we show the parameters for each research question, prior to experimentation.

- **Research Question 1**

The parameters for a CNN model that can predict steering angles given an image are shown in *Table 2.* Our batch size is very low, value of 8, because of the limitations of the GPU used. We used a NVIDIA GTX 1060 with 6 GB memory. Given the size of our image, 300 x 250 x 3, the number of images being approximately 32,000 and the size of our model, 7 convolution layers with 3 fully-connected layers. This architecture resulted in a nearly 100% utilisation of GPU memory and computation capacity.

We also compared our model against a 3 and 5 convolution layer plus 3 fully-connected layer network. All parameters remained the same except batch size, which could be increased slightly. This is due to not knowing how large the model is in GPU memory even though our image data is relatively small. Due to the small batch size, all our architectures take approx. 3,900 steps per epoch. This equates to approx. 9 minutes per epoch in our 7-layer architecture. We use a 90-10 split for training and validation. The learning task is to lower the MSE to an acceptably low level.

- **Research Question 2**

The goal of the second research question is to identify if a variable sampling rate has any effect on computational expense without compromising the MSE. To achieve this, we apply some programming logic to our data pre-processing step. Before loading the data into the model, we remove either 10%, 20% or 40% of straight road images after each last bend image in a sequence. Polder blindness (Koornstra, 1989; SWOV, 2012) is simulated this way because it takes some

time for the driver to transition into passenger mode. The variable sampling rate will be carried forward as part of the model if the results meets our expectations.

Table 2- Table showing parameter values of our CNN model for Research Question 1

| Parameter | Value | Description |
|---|---|---|
| Batch Size | 8 | Samples per gradient update. Default 32 |
| Epochs | 40 | Number of iterations. Default 1 |
| Learning Rate | 0.0001 | Step size per update. Default 0.01 |
| Momentum | 0.0002 | Accelerates SGD in the right direction. Default 0.0 |
| Optimizer | SGD | Stochastic gradient descent optimizer. |
| Validation Split | 0.1 | Size of the validation dataset. Default 0.0 |

- **Research Question 3**

In the task of transfer learning, we will be using the model created and finalised in our first research question. In this process we transfer the trained model, including the weights, and input different data. Specifically, we use the right-hand side dataset, which is significantly smaller than our original data. Apart from a small dataset, faster convergence, i.e. fewer epochs, has been attributed to transfer learning approaches. The specific parameter values of this model can be seen in *Table 1*. In our literature we found that freezing layers was an effective method of gaining effective results. Another approach we took was to use a model and weights that share a similar domain, i.e. driving.

To measure the success of our model we will test to see if the MSE is either similar or lower than our original MSE. We will also check if a similar or lower MSE is reached in a smaller number of epochs.

## 4.2 Mean Squared Error

Given our above settings, we proceeded to train the model and record their MSE. It is important to note that we used the MSE as the loss function for the model itself. In this thesis, we look at answering 3 research questions:

- Which architecture will best train a CNN that can predict steering angles from a given image?
- Does a variable sampling rate on straight roads have any effect on the computational requirements of the model without compromising on the error?
- Can the final model, trained on left-hand side data, be transferred over to a right-hand side driving scenario? What error rate does this achieve?

After running our experiments and tuning with different parameters, we have left 3 models with various MSE. Instead of selecting the model with the lowest MSE, we must test if that result is statistically significant. To carry out significance testing we use t-tests, specifically paired t-tests. A t-test is a type of significance test that is used to measure and compare the means of two sets of data. We use a paired t-test approach as our dataset remains the same through our experiments and thus the distribution is unchanged too. The paired t-test will let us compare which model performed the best, statistically, on our dataset. To conduct this test, we present our null and alternative hypothesis for the first research question.

In this case, our null hypothesis is that our results are consistent with random chance and changing the depth of our model had no significant effect on the MSE. Our alternative hypothesis is that using a 7-layer CNN is beneficial for this project as compared to a 3-layer or 5-layer CNN as it indeed does lower the MSE. The p-value threshold for rejecting the null hypothesis is set at $\alpha < 0.05$. Given Table *3*, we can see the means and standard deviation of all our architectures. Testing the 7-layer CNN against both the 3 and 5-layer CNNs yields a p-value below our threshold. Given the results, we can reject the null hypothesis.

Table 3- Table showing the mean MSE, standard deviation and p-value of the 3/5-layer CNN as compared to the 7-layer CNN

| | 3 Layer | 7 Layer | 5 Layer | 7 Layer |
|---|---|---|---|---|
| μ MSE | 0.24792 | 0.17458 | 0.25434 | 0.17458 |
| σ | 0.15532 | 0.11792 | 0.16172 | 0.11792 |
| α-value | | 0.00475 | | 0.00166 |

To ensure success on our first research question, our goal was to reduce the MSE as much as possible. The best MSE we achieved was 0.0278, which can be seen in *Figure 11*. Using this model, we will experiment with our next research questions. Please note the scale of both y-axes in each Figure *10* and Figure *11*.

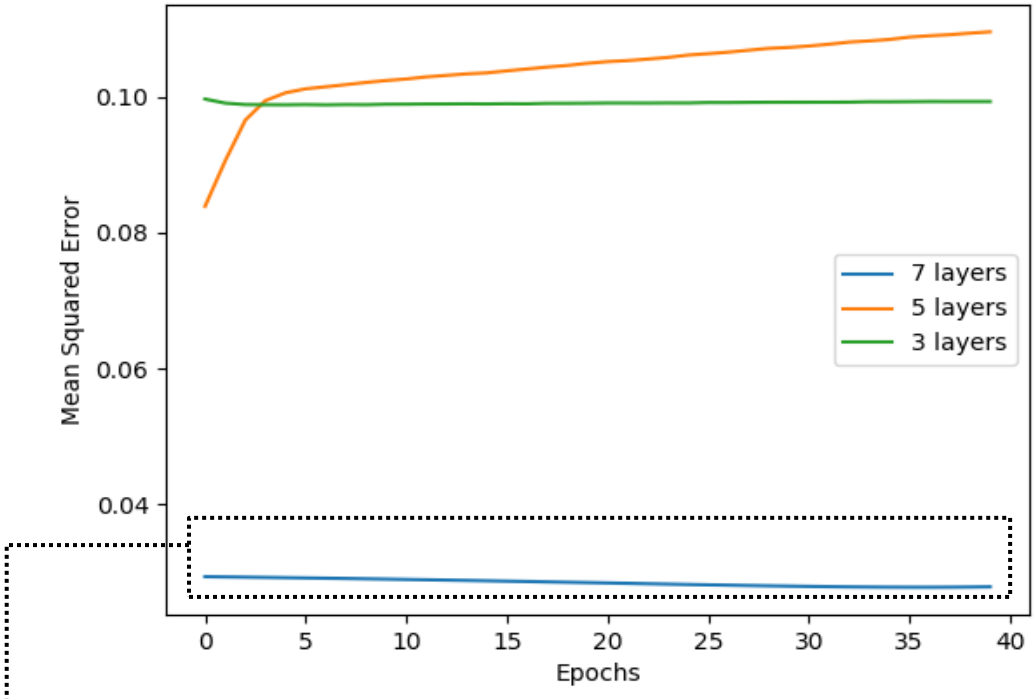


Figure 10- Graph showing the MSE of all our architectures. Note scale of Y-axis

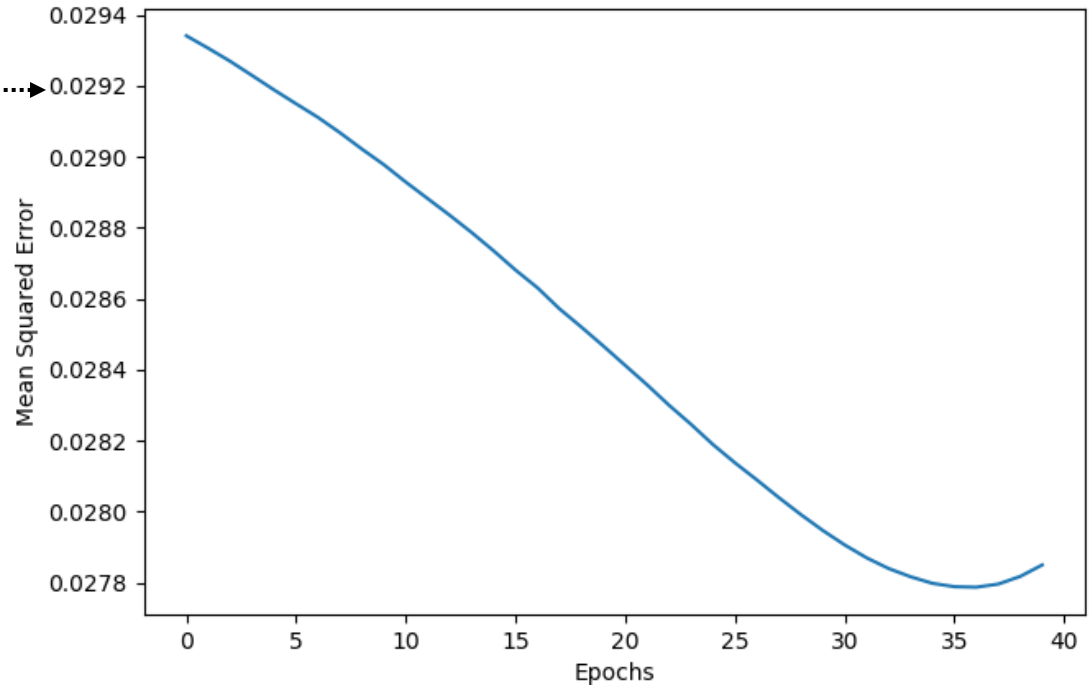


Figure 11- Graph showing the MSE of our 7-layer CNN. Note scale of Y-axis

Our second research question looks at identifying if a variable sampling rate can reduce the computational expense of training our model whilst giving a reasonable error rate.

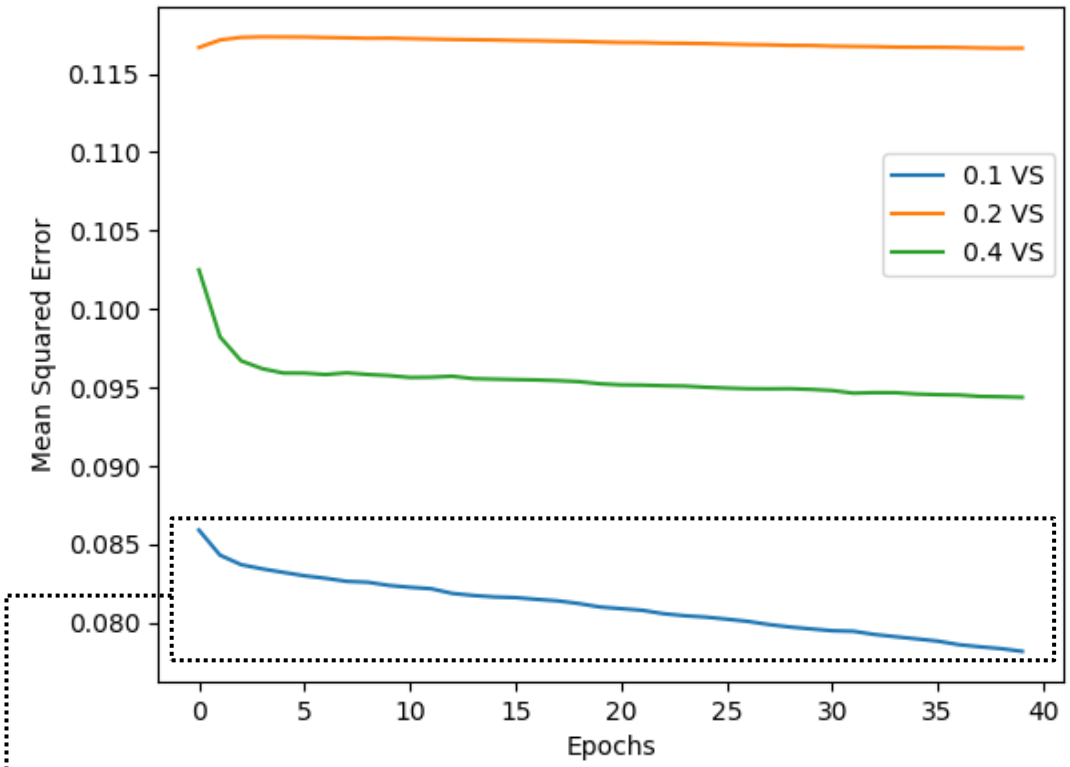


Figure 12-Graph showing the MSE of all variable sampling rates. Note scale of Y-axis

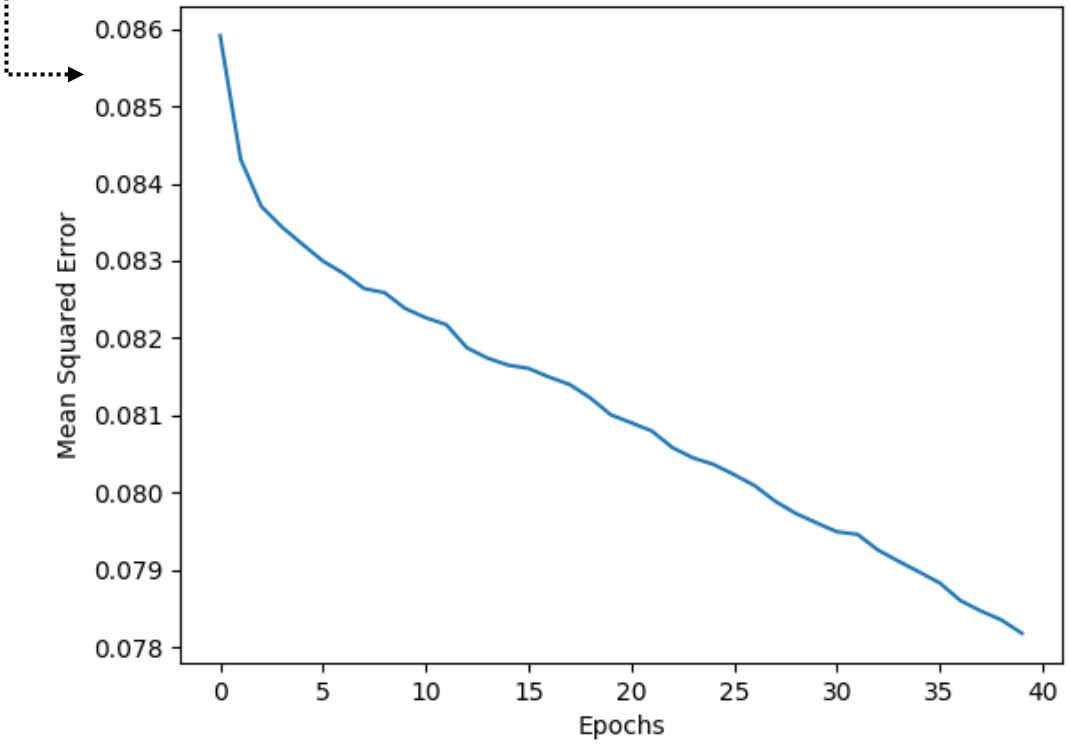


Figure 13- Graph showing MSE with a 0.1 variable sampling rate. Note scale of Y-axis

Given *Figure 12* and *Figure 13*, we can see that a variable sampling rate of 0.1 resulted in the lowest MSE, 0.078. Please note the scale on both the y-axes in each figure. Apart from just the MSE we also put emphasis on the possible decrease of computational resources required when training this model. Without applying a variable sampling rate, the time per epoch fell at approx. 9 minutes. After applying either of the variable sampling rates, the time per epoch fell at approx. 8-9 minutes. There was no significant decrease in the model training time that could justify the significant increase in error rate.

Our final research question looks at transferring a model trained on a left-hand side driving dataset to a right-hand side driving scenario. As mentioned earlier, our aim is to see if we can achieve a similar or lower MSE and faster convergence, i.e. fewer epochs. The approach we took was to only transfer weights from specific layers, in our case, layers 9 and 10 from *Table 1*.

Following this we were able to achieve an MSE of 0.00080, Figure *14*. Not only that, we can also see that the model converged in a shorted amount of time as compared to our original model. When comparing freezing only the middle layers as compared to freezing the whole model, we achieved a significantly low result. Based on the graph, we can possibly keep training out model as it still follows a downward trend near the end. Although the graph shows some instability near the end, so we may need to perform some tuning to smooth it out.

Apart from experimenting with freezing combinations of layers, we looked at the learning rate and momentum. This is because as the model already contains trained weights, the learning rate does not need to be as sensitive and therefore requires less time to converge. From our experiments we found that there is no change to the parameters listed in Table *2*, except for the number of epochs.

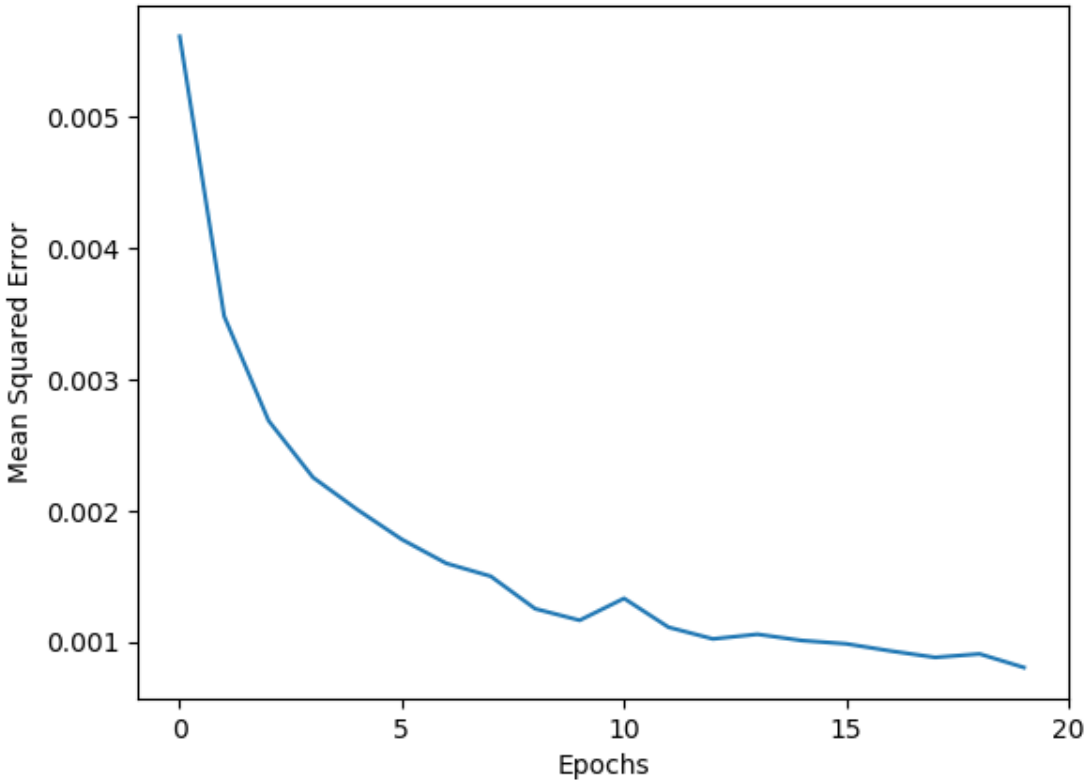


Figure 14- Graph showing MSE of transferring only mid-layer weights. Note scale of Y-axis

## 4.3 Limitations of the Experiment

After training the model for our driving scenario, the results show a successful outcome for our research questions one and three. We tried to remove as much bias during the data collection, pre-processing and training process, but there do exist some limitations in our model and process. Some of these limitations will be highlighted here and discussed, in detail, in the next section.

As we looked at only one driving scenario, this model is limited to that specific scenario. The contents of our map are also limited to be similar to NZ roads, and US roads for the right-hand side driving scenario. When looking at the data as a whole, we use simulated data, which may not be the best replication of a real dataset. The reason for not using a real dataset was due to the constraints and limitations of those datasets discussed earlier.

We conduct an end-to-end approach for learning and predicting steering angles. Our model, created initially, has 2 inputs, image, and speed. The image input is processed using a CNN to extract feature maps, whereas the speed is input directly to our fully-connected network. A

limitation of this method is that we don't provide any speed information in the images, which could make it learn better and be more generalised.

We were also limited by the fact we were only using one RGB camera in our research. This approach is limited by our image input as being a 2D image and applying a 2D convolution process. Our images carry no depth information that may allow them to see how far a turn is, or how sharp it maybe. Having one camera also affects the field of view of the vehicle when it is turning, as the camera may not capture all the information. Adding multiple cameras may reduce this effect, but currently we may be losing some information regarding this.

When we tackle our second research question, simulating polder blindness in an effort to reduce computation expense, we present a hard-coded approach to the pre-process step. The benefit of this method is that it does not hold power when using during a city driving scenario. The downside of it is that during a highway scenario, scene changes, even on a straight road may be disregarded by the model. Using a machine learning approach to simulate polder blindness may yield better results whilst ensuring the model is generalised for driving scenarios. Based on our results, it may provide no beneficial input for an autonomous vehicle. However, this would be a challenge for a completely different problem which will be looked at in future workings.

Looking at our transfer learning approach, we used our own model for training purposes. The limitation of this follow the same limitations that exist when training our own model initially. Transfer learning itself assumes that's there is some knowledge that maybe used from the source task for the target task. This maybe one of the reasons we have gotten good results on our target task, as both tasks are quite connected given Figure *4*. On the contrary to this, our results could be better when collecting enough data for the target task and training a model from scratch.

# Chapter 5

# Analysis and Discussions

*In this chapter, we present an analysis of our findings and how they affect our research questions. We critically analyse the model, parameters and layers to understand the performance deeply. We also look at how the results impact the research goal and the contributions made by this thesis.*

## 5.1 Results and Discussion

In the previous section, we show the results of our model when it is run against our dataset for each research question. We show the effects of a convolutional neural network on an end-to-end steering prediction task. We also highlight some limitations that may have caused shortfalls within our process. In this section, we critically analyse the results for each research question, the impact of those results and the impact of the limitations we identified earlier. We also show some output for why our model behaves as it does, and what the model maybe learning during the training process.

The first step in our research was identifying the type of sensor that will be used to collect our data. We chose to use the RGB for our research as it is commonly used for steering angle prediction. It also provides the model a way to view the road and environment in a manner that humans view it. The downsides of using a single RGB cameras approach is that there is no depth information present. Due to this, other approaches are taken such as LiDAR, stereo RGB cameras allowing the model to learn from depth information. The reason we use a single RGB camera is because their use has been proven to work across a range of image processing and self-driving vehicle tasks.

Data collection was very important as we had to ensure our data was as rich as possible. Using CARLA allows us to collect the type of data that would be beneficial for our research without the huge overheads associated with real-world datasets. This was one of our biggest issues with using real-world datasets as collectively, they covered a range of different scenes but individually, they were limited. The limitations included imbalanced data, lack of weathers, lack of road types and lack of corrected driving. The major shortfalls that mattered most for us were the imbalance of bend roads versus straight roads and the lack of corrected driving. This was one of the main reasons we decided to use a simulator-based approach. Reducing the gap between straight roads and bend roads as it ensures our model isn't biased to a certain steering angle. If we have more of straight roads, where the steering angle is closer to 0 degrees, then the model will output closer to that to not get a low MSE. The reason for including corrected driving is so

that the model learns how to recover to unfavourable situations. Our method of including corrected data was having a user control the car and drive it on the track. It was done in this manner because the human has no experience with CARLA and controlling a car using a keyboard and was a learning curve for them as well.

CARLA provides us with the ability to manipulate the environment as we see fit and allow us to create scenarios which remove the need for data augmentation. Using CARLA, we were able to run scenarios in a manner which allowed us to extract even number of roads with bends and straight roads. During the data collection stage, we wanted to ensure that we don't have any synthetic or augmented data. The reasoning for this was that augmented data may add some bias to the system. There is only so much augmentation one can do before the image becomes a liability for training. It is better to ensure the dataset is as raw as possible. This method of data collection and the possible issues in the data are relevant from both the left-hand drive dataset and the right-hand drive dataset.

- **Research Question 1**

Once we collected our data, we looked at the structure of the model that will satisfy our first research question. Our final model consisted of 7 convolutional layers plus 3 fully-connected layers and achieved an MSE of 0.0278. Prior to this we experimented with 3-layer architectures and 5-layer architectures as well because these were commonly used in steering prediction tasks. We chose to experiment with these initially to set up a baseline and took inspiration mainly from Bojarski et al., 2016 (Bojarski, et al., 2016). Evident from our experiments, these architectures were giving a relatively higher MSE. In our view, this could be because the feature maps were not rich enough due to the fewer layers. As the CNN gets deeper, the filters start learning more higher-level features that maybe important for the task. A way to interpret which features are being learnt at which layer is to visualise the activation maps and trained filters. See section 5.2 for more details regarding this.

Something we considered, which may have limited our experimental results, was the lack of speed information in our images. As we are passing a speed-based input, using 3D

convolutions would provide some spatial data for the model to 'perceive' speed. Another consideration was the amount of input data that we collected for our model. We collected approx. 52 minutes of driving data for a single scenario. Other researchers used real-world datasets which contained up to 10s of hours of driving footage and covered multiple scenarios. For our use case, slightly under an hour is beneficial.

When looking at our result in *Figure 11*, we can see a very smooth curve declining in an even manner. There is a slight increase at the end which could indicate our current result is in a local or global minimum. Since the error plane is not known, we are unable to identify if this, 0.0278 MSE, is the lowest MSE that can be achieve with our current model. The MSE going up can also signify that the model was not generalising enough and maybe be overfitting on the training data. Given the amount of training data provided for a single scenario, this could be the case. We also experimented with 50 and 60 epochs which only showed the curve kept increasing, in most cases, after an initial downward slope. When looking at our parameters, the learning rate and momentum are quite low. We found that literature consistently kept a learning rate quite small but there was a split between the momentum values. Through our experimentation we found that both the parameters performed well with the current values. To combat over-fitting, we made use of dropout layers after each of our dense layers. This was a proactive decision as our model contained a large number of parameters. Based on these factors and the low MSE achieved by the 7-layer architecture, we decided that it would be best for our needs.

- **Research Question 2**

For the second research questions, we use the successful model from the previous research question. We apply multiple sampling rates, 0.1, 0.2 and 0.4, to our input data to measure their effect on the MSE and training time. Based on our experiments, we found that the variable sampling rate of 0.1 yielded an MSE of 0.078. We also didn't see a significant change in the amount of time taken to train the model. Due to these reasons we decided that, given our experiments, simulating polder blindness does not have a positive effect on the task of steering angle prediction.

During the data collection step, one of the shortfalls we identified was that there is an imbalance of bend road images as compared to straight road images. We took steps to overcome this issue so that our model is not biased towards a straight steering angle. Our method of simulating polder blindness involves sampling only some straight road images after bend road images. Effectively, this is reducing the number of straight road images being fed to the model. Unlike before, now our model has not been trained evenly on straight road images, causing bias for roads with bends, thus, non-zero steering angle. We are effectively giving the model an imbalanced dataset for training. A 0.1 sampling rate is less imbalanced than a 0.2 or 0.4 sampling rate and therefore it makes sense it is producing the lowest MSE from the three.

Looking at the graph, *Figure 13*, we can see that it is not as smooth as our initial graph, *Figure 11*. This signals that, during the learning process of the model, there were some irregularities which may have cause the MSE to drastically spike or lower in some places. Although we can observe an initial steep decline followed by a steady decline, we can see that there is no up spike at the end. We also found that if the number of epochs were increased, i.e. 50 epochs, then the MSE still follows a downward trend. It could be possible that with a 0.1 variable sampling rate we can reach a very low MSE, but this would either take many epochs, due to our learning rate, or a more complex model. This does not align with our current goals but can be looked at in further work. Given this outcome, the addition of a variable sampling rate does not yield good results and is not carried forward.

- **Research Question 3**

Given the success of our 7-layer architecture, we decided to use it for the task of right-hand side driving. The goal of this task was to achieve a similar or lower MSE in a fewer number of epochs. Evident from our experiments and *Figure 14*, we achieve as MSE of 0.00080 in 20 epochs. This was done through freezing only the weights from the middle convolutional layers and training the model.

When we experimented with freezing combinations of other layers, or freezing all the layers, the results were very different from the current. In fact, when we freeze all layers, we achieve an

MSE of 0.1130. This may be due to the fact that the model is using the same logic as left-hand driving on right-hand side. We know that as humans travel to other countries, we must adapt and be open to learning the act of driving in that environment. Therefore, as the fully frozen model is not updating its weights, it is not learning anything new. We can assume that as being the reason of such a high MSE. On the other hand, as we freeze the middle layers, we allow the model to learn from the new inputs, and tailor the outputs to its needs and reduce the MSE.

The reasoning for freezing only the middle layers is because the layers closer to the input maybe to domain input specific. As such, the layers closer to the output maybe too domain output specific. The earlier layers may not be extracting the relevant features to generate rich feature maps that are required for the model to be generalised for the target task. Whereas the later layers construct feature maps tailored to ensure domain specific high-level features and thus, a low MSE for left-hand side driving. The middle layers may hold the right amount of generalisation in their feature maps, which allow for them to be transferred successfully to a target task. We can observe some of the trained filters and feature maps, in section 5.2 to get an idea of what the model maybe learning and what features are being activated.

When looking at the specific parameters for training this model we initially thought that the learning rate need not be so sensitive, as per the original model. As the model as already learnt the weights, it makes sense that only fewer steps maybe required for it to converge. However, in our case, most layers still need to be trained hence, not a significant change in the learning rate was needed. If we instead freeze most or all layers, we may not need a sensitive learning rate.

Given the successful result of our transfer learning model, we can provide evidence for transfer learning between different orientations of driving possible. There is of course a need to conduct more testing on other scenarios, given that we only tested a single scenario. This will be presented in future work.

## 5.2 Model Visualisations

To understand the model more deeply and identify what the filters of our model are learning or which regions in an image are activated, we can visualise the filters and feature maps. As mentioned earlier, during the training process, we specify how many filters, or kernels, each convolutional layer should have and the size of those kernels. The kernels are responsible for learning features throughout the image. In this section we go over and understand the visual outputs of the model including filter visualisations, feature maps and saliency maps.

Looking at our first convolutional block, layers 2 – 4 from *Table 1*, we can see that we have 256 filters in that convolutional layer. We chose a relatively high number because as then input image is quite large and may contain a lot of rich spatial information.

From *Figure 15*, we can see some of the filters that the model has learnt. Knowing exactly what the features present in the filters relate to is difficult. We know that the initial layers in a model learn low level features, such as lines and edges. Some filters, such as row 3 middle, and row 6 first, seem to learn some thick lines. In our training set, there are some thick structures present such as poles of road signs, railings, light poles, buildings etc. Row 5 middle, seems to be picking up a thin horizontal line along with row 3 first which seems to pick up a cross of some sort. Other filters, such as row 2 middle, row 4 middle and third, row 5 third, and row 6 middle and third seem to be activating awkward regions. Especially row 1 middle, seems to identify something that doesn't seem connected. Some of those mention filers can be considered slight diagonals and maybe picking up on any diagonal lines in our image. As mentioned earlier we did apply some padding as well, and areas where only some regions in the filter are activated could be considered edge cases. Most of those filters, however, seem to identify lines and edges as expected from the image.

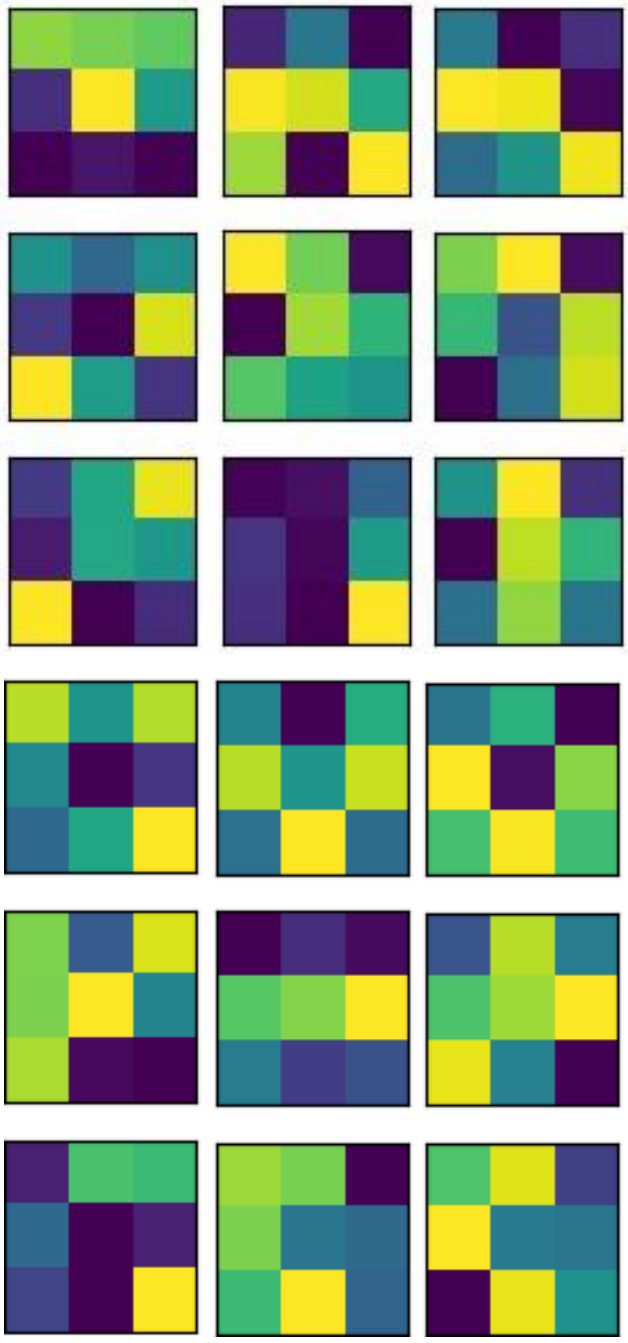

Figure 15- Filter maps from the first convolutional layer of our model. Areas of interest are shown as dark regions.

In our transfer learning task, we transferred the weight of the middle convolutional layer. In the middle layers we should expect to see some more generalised features, such as hints of curves, diagonals etc. The number of filters in this layer is 32 as the feature maps from the previous layers are smaller and richer. Given that the kernel size is still 3 x 3, seeing high level features maybe a bit difficult. Given the perceived quality of the filters and activation maps, we may be able to justify the MSE achieved by our transfer learning task.

From *Figure 16*, we have filters that have more information than just straight lines and edges. The filters in row 2 third, and row 3 first, we can see clear and well-defined diagonal filters. These filters could be identifying the curves on the road signs, the edges of buildings etc. In some of those filters we can see a slight housing style curve too, i.e. row 3 middle and row 4 last. These seem to identify the top of the road signs, or some small curves. Something that does look like the connection between the circular road sign and top of the pole, is the first 2 filters in the last row. The 'T' shape maybe referring to this connection between the pole and the signboard. Lane marking still seem to be present through the filters representing straight lines, thick or thin. Something to note is that even in this set of filters there seem to be filters that may signify edge cases. Although from our sample they appear to be much less than the first layer filter. Apart from the filters we can also look at feature maps, Figure *17*, that are created as the output of the fourth convolutional layer.

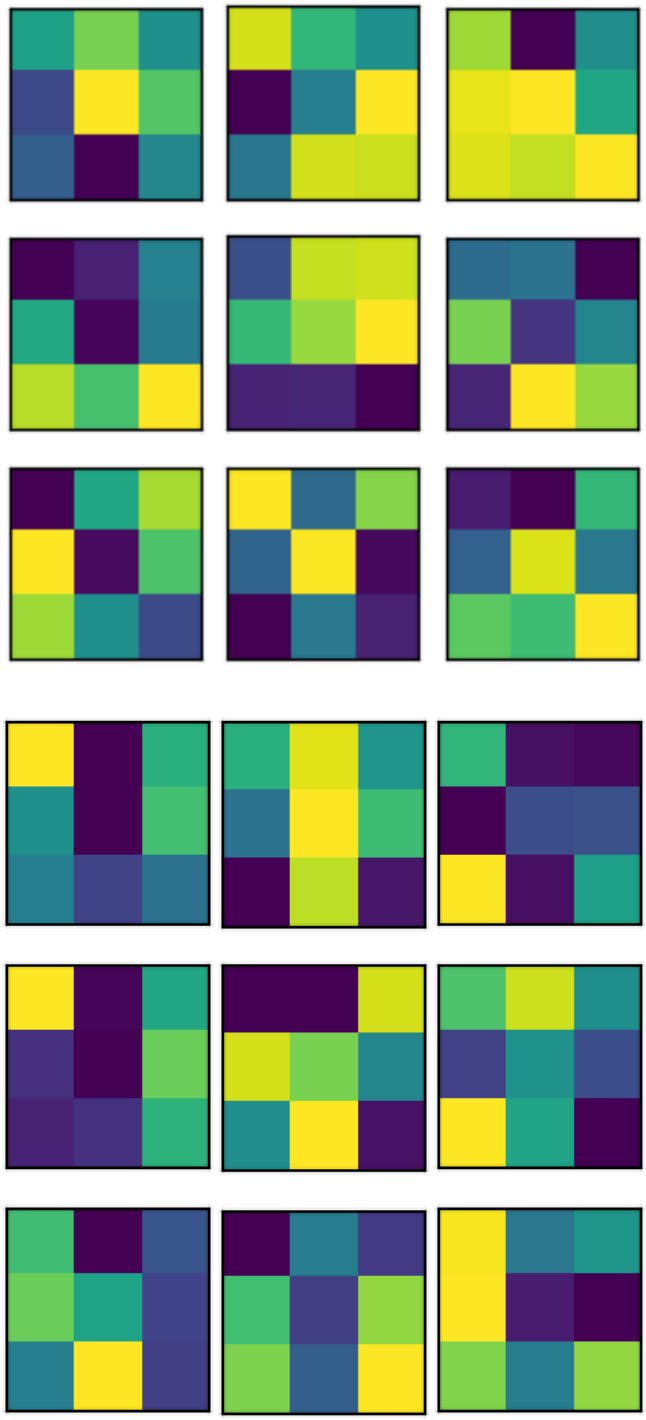

Figure 16- Filter maps from the middle (4th) convolutional layer of our model. Areas of interest are shown as dark regions.

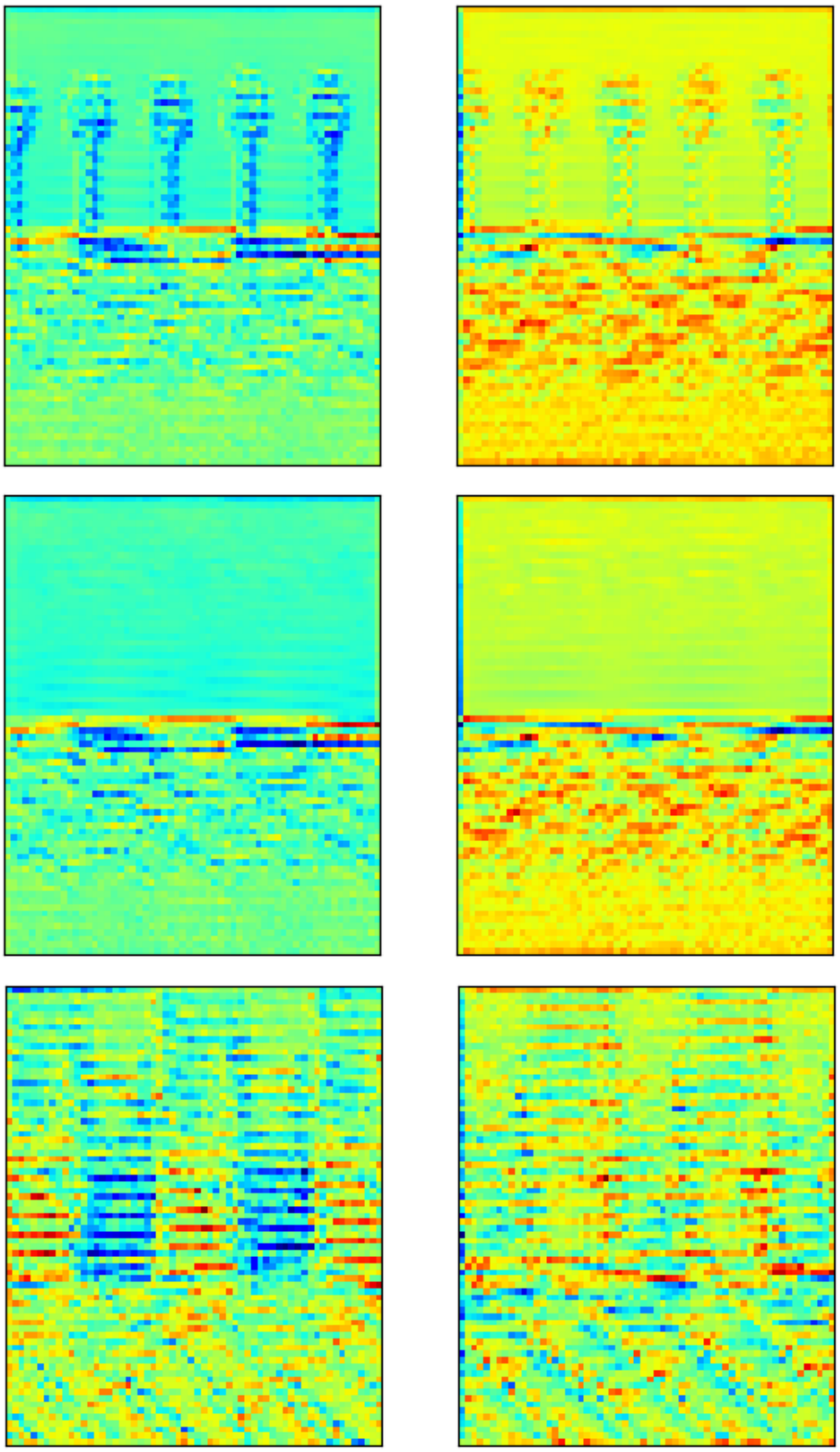

Figure 17- Feature maps from the middle (4th) convolutional layer. Areas of interest appear brighter as they achieve most activations.

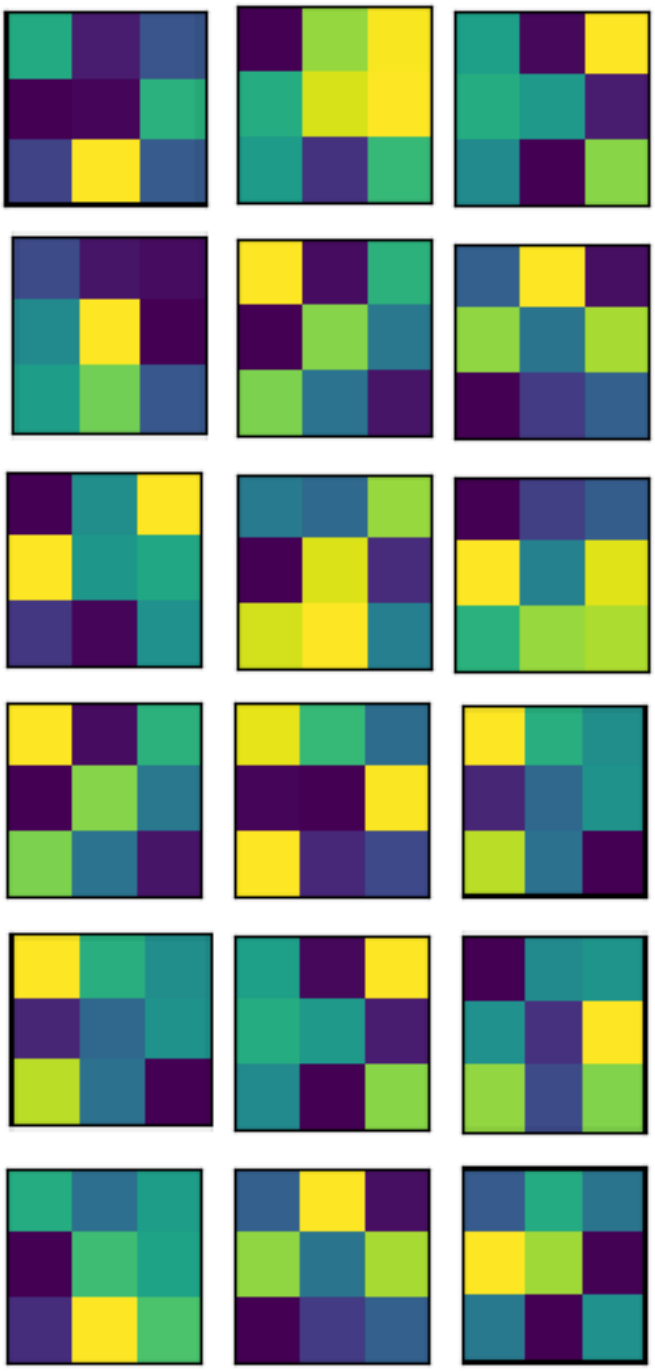

Figure 18 - Filter maps from the final convolution layer of our model. Areas of interest are shown as dark regions.

From the visualisations of the 4th layer of our CNN, Figure *16*, we can see that it is generalising on the inputs. The generalisation is on par with the MSE achieved in both our first and third research questions. This would show that freezing the middle layers during the transfer learning process did prove beneficial. From the visualisation, it is evident that it has learnt the important features that, according to humans, are required for steering angle prediction.

The final convolution layer is expected to have the richest filters as we expect the noise to have been removed. From *Figure 18*, we can see the output of some filters from our final layer. From the sample we have present, there appears to be significantly less noise, from a human perspective, that the previous visualisations. The noise in this sense is filters with random activates that don't form any complex, or general, shape. Some interesting observations, filter at row 2 middle and row 4 first, appear to be quite similar. This could possibly show that some features maybe redundant as they are being learned multiple times. Apart from that we can start seeing some extremely complex shapes com into fruition. Row 1 first filter shows a corner, which is quite a general shape. We can see 'Z' shapes, curves etc. It may seem as though transferring these weights would have a much better effect on our transfer learning task. We have to remember that these filters are highly optimized for our left-hand driving task and out specific to that, which maybe be why they didn't help during training and validation.

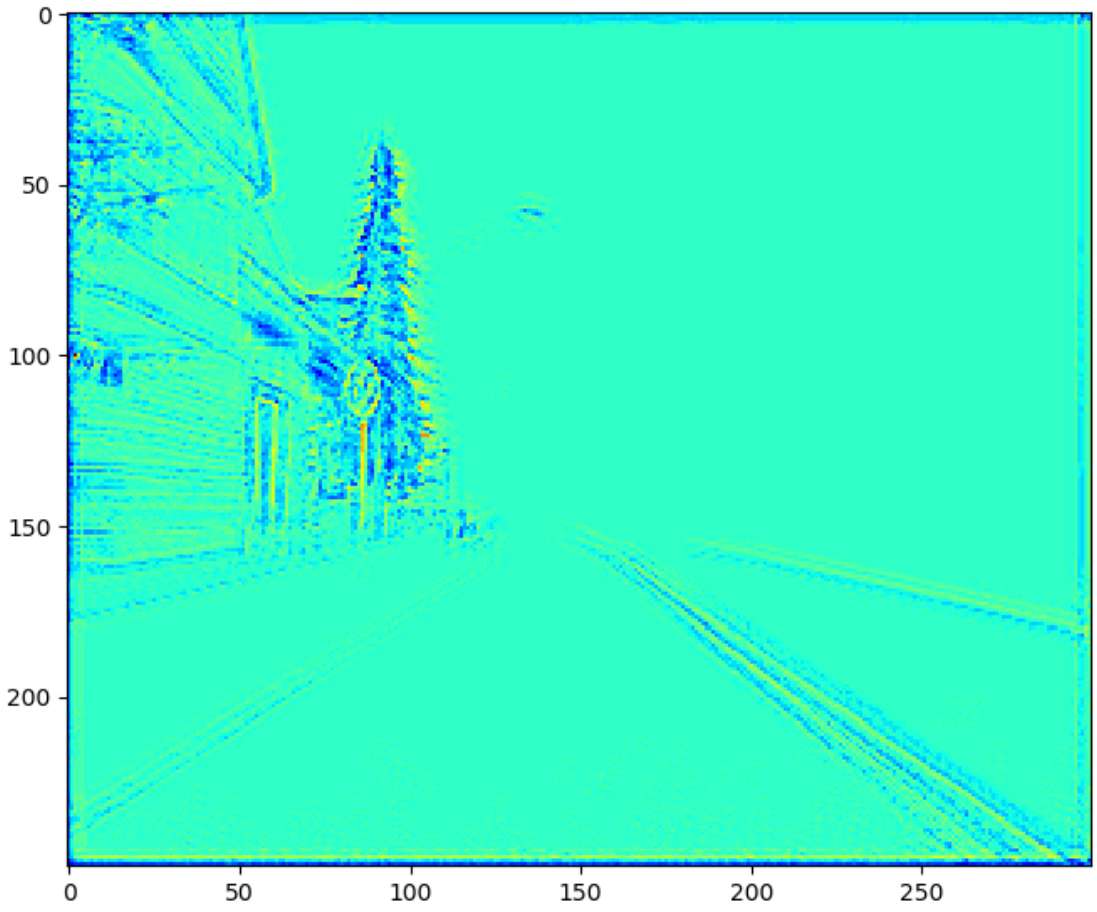


Figure 19- Output Saliency Map (1). Areas of interest are shows as the only regions that are activated. The brighter the image, the higher the weight towards decision. Image from the left-hand drive dataset.

Apart from visualising filters, we can visualise the feature maps on convolutional layers as well. In *Figure 17*, we can see the output feature maps from the fourth convolutional layer. Each row consists of a different input image from our training dataset. We can see the different features extracted by the filters and how they impact the feature maps. Convolutional Neural Networks are known to be location invariant. Which means that, it does not matter where the feature is on the input images, a trained filter will be activated once it detects the feature in question. From our examples, we can see that the features are being generalised well.

Saliency maps (Simonyan, Vedaldi, & Zisserman, 2013) also provide an alternative method for understanding the output of a CNN. These maps show the spatial activations in a given image, and let us, somewhat, understand the activations. Looking at the saliency map presented in *Figure 19*, we can see that our model emphasises lane markings, and edges of the road. These two features should be considered quite important for this task. It seems that the model is also placing emphasis on building and tree edges and outlines. This maybe because, as the vehicle turns the bend, new structures appear. In our image we can also see that the road sign is activated. We knew from analysing the filters that maybe some road signs are being identified, and we can confirm that here. Although in this image the road sign is quite a distance away, we can still make out the '60' on it. At approx. coordinates 120, 150, we can see a blue patch on the edge of the road which signals the start of the railings. In Figure 20, we have a collection of saliency maps to get a broader idea of the output.

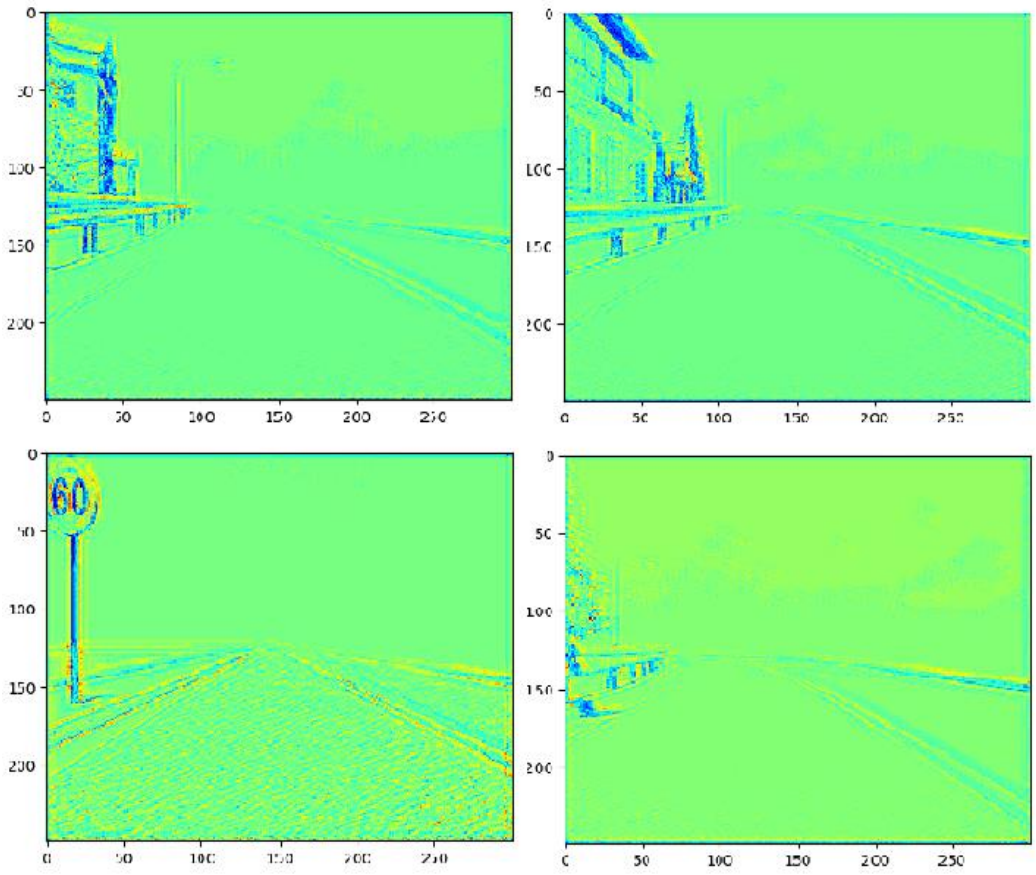

Figure 20- Output Saliency Maps (2). Areas of interest are shown as the only regions that are activated. The brighter the regions, the higher the weight towards decision. Images from the left-hand drive dataset

Finally, we revisit the 6 – level self-driving vehicle system of classification. From our experiments, our model can predict steering angles with a low validation MSE which would put it as a level 2 autonomous system. This is because of the limited training the model has, and the fact that the error is still present in the model which may require a human to take over. Apart from that, from a human perspective, we can see from the various visualisations, the model is learning features that are critical for humans to steer the vehicle. It identifies road edges, railings, signs, building and curves. In our saliency maps, these features are activated which shows they lead to the output decision. However, in some cases it does focus on features that do not provide much input for predicting a steering angle, which may show that our model is still in infancy, like a new driver. It needs more training across more samples and be more generalised to the best results.

We can examine the difference between the features learnt from the left-hand drive model with the right-hand drive model by looking at Figure *21*. We can see that there is slightly more noise in the transferred model, as the model is learning not just the road marking but also the road itself. This is similar to Figure *20* as if there are no other features present on the road, i.e. buildings, trees, the model learns the road itself. In the case of the transferred model, if these features are present, it is still learning the middle of the road. However, the model still learns roads signs and focuses on other aspects of the environment, i.e. railings, buildings etc. The time of day also has an effect on the features being activated in the transferred model as the top two images in Figure *21* are during the evening, and the bottom two are during the afternoon. This could show that, as there are less features 'visible' during less sunlight, the model, like a human, is trying to absorb as much information from the environment as possible. Apart from that, this could shed some light into our understanding of how humans transfer features when they change from left-hand drive to right-hand drive.

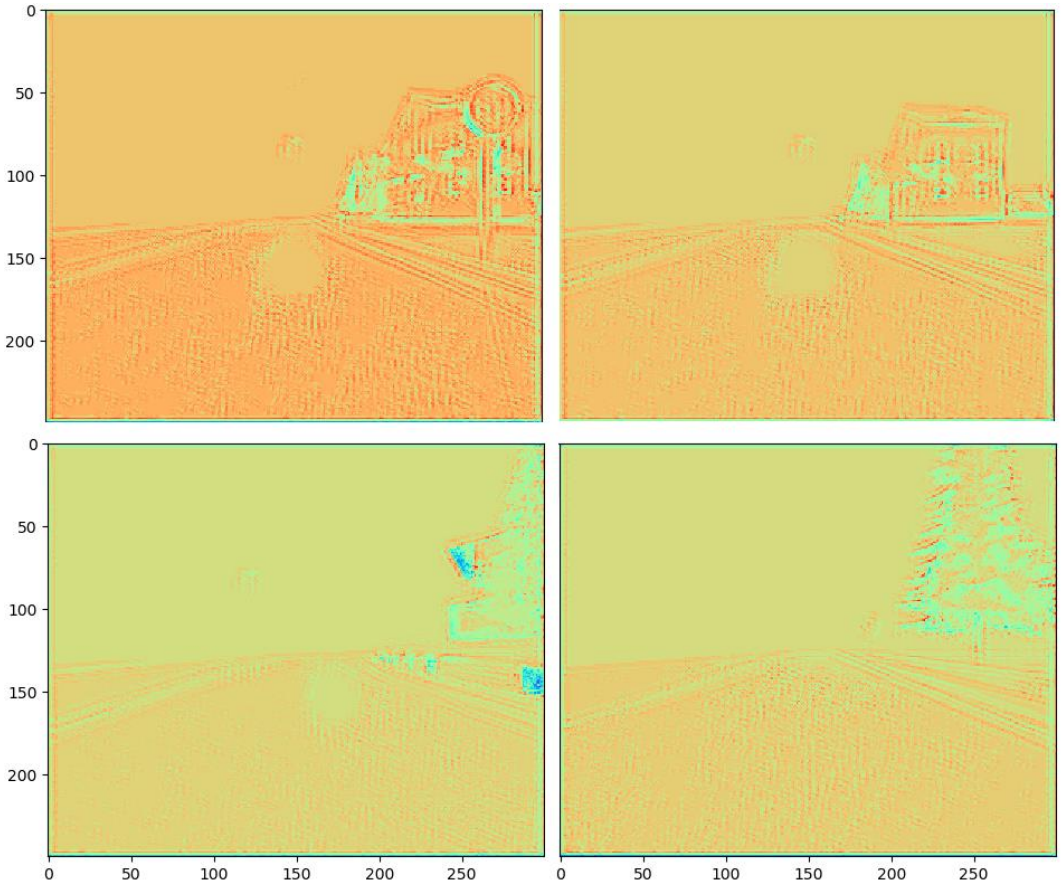


Figure 21 - Output Saliency Maps of Right-Hand drive data. Areas of interest are highlighted as activated regions. The brighter the regions, the higher the weigh towards their decision.

# Chapter 6

# Conclusion and Future Work

*In this thesis, we overview our research goal and research questions. We present how the task of steering angle prediction is carried out and what our results mean. We also present some future work that can address our limitations and expand this research further to contribute to the task of steering angle prediction.*

## 6.1 Conclusion

Human error results in many fatal accidents each year, in many countries. Autonomous vehicles are seen as the way to avoid these tragedies. Given the complex nature of driving, which may seem easy for a human, building capable autonomous vehicles is currently a challenge. A machine learning model requires accurate and annotated training data to learn from. The training data must be generalised enough so that the model can learn how to perform in a wide variety of situations. Training the model for all situation will be a difficult and endless task. From there, an appropriate neural network model must be chosen or created which can learn sufficiently. In the task of image processing, convolutional neural networks have shown significant results over the past few years, mainly due to the advancements in computation hardware and dataset size and richness. Autonomous driving with cameras is primarily seen as a simulated or real-world image processing task, and as such CNNs are used to tackle this. This is because humans rely heavily on their perception to navigate a vehicle and therefore this sense would be the most beneficial. In the case that RGB cameras are not present, autonomous vehicles have been created to only use sensor inputs and perform quite well. This is mainly due to the fact that a vehicle must detect and recognise objects in its surroundings and make the best decision. Recognition of objects need not only be from RGB cameras, but can also be done with other forms of input, i.e. SONAR, RADAR etc. There are some challenges involved in image processing such as partial objects, unknown objects, imbalance of objects during training etc. This also overlaps with the task of ensuring that a rich and well annotated training dataset is present.

In this research, we used CNNs and image processing to understand how deep learning and transfer learning can be accomplished for a common driving scenario. We broke our research down further into three research questions. 1) Identifying which CNN architecture is best for learning and predicting steering angles based on image and speed inputs. 2) Identifying if a variable sampling rate has any effect on the computation expense of the model without sacrificing the error. Finally, 3) Can a model, trained on a left-hand driving dataset be transferred for predicting steering angles from right-hand driving data.

We created a simulated environment using CARLA, and loaded our scenario, custom made using RoadRunner. We collected over 50 minutes of left-hand side driving data, across multiple weathers and time of days using an RGB camera. We then sampled it at 10 FPS to provide images and speed information. The image data was then input into our CNN architecture for experimentation. From our results, we identified that a 7-layer CNN architecture satisfies the first research question. This architecture achieved an MSE of 0.0278, which is significantly lower than other architectures. When visualising the trainable filters and feature maps, we can see that the model is capable of learning the required features. These features may include, road markings, road signs, railings, edges, etc. From a human perspective, these features are arguable very important for the task of driving.

Our second research question, unfortunately, did not yield fruitful results. We could not see any significant decrease in computational expense given the significant increase in our MSE, 0.078. Experimenting for this question reintroduced the data imbalance problem we faced initially. This problem is considered to be one of the main reasons why the model did not perform as expected. Following this, our results indicate that simulating polder blindness, in our model, is not something that can be implemented. Although it is crucial to carry out more work on this topic in the future and address the limitations presented.

Our third research question is considered a success as we saw a lower MSE and faster convergence time. Specifically, we achieved an MSE of 0.00080 in 20 epochs on a right-hand drive dataset. Visualisations from the $4^{th}$ layer of our first model can provide evidence on why both the models were compatible. The layers transferred from our initial model contained the generalised features from the training images. This maybe one of the reasons why the model learnt quicker as it did not have to train those layers. Apart from that, both the source task and the target task lie in the domain of autonomous driving. They are only slightly different but not enough to cause any issues with the learning process. Given our results, we can say that it is possible to use a model trained on a left-hand drive dataset with a right-hand drive dataset, for our scenario. The model will learn to generalise in significantly less time for our scenario.

Given our results, we can confidently say that for our scenario deep learning and transfer learning can be used for the task of steering angle prediction. As mentioned earlier, there exist many limitations with this research that can be addressed in future work. Apart from that, there are a few avenues this research can go towards to explore newer areas in more detail.

## 6.2 Future Work

One of the initial limitations of our research is the fact that we have used a simulated environment for our data collection and experiments. Although there exist some publicly available simulated datasets, due to the constraints identified earlier, and the scope of our research, they were not used. As CARLA is quite easy to use and offers a wide range of tools to manipulate the environment as one see fit, it seems ideal for future experiments too. In the future, a more extensive scenario, including all urban, sub-urban and rural driving environments should be created. This will allow us to collect more data and create a generalised dataset that covers most scenarios. Another reason for sticking with a simulated environment is to add any edge cases to the training process of the model. Edge cases are extremely difficult to find in real-world datasets due to their potentially dangerous nature to collect. Simulated environments can be used to simulate them in many different scenarios. This can then be used to train our model to a larger steering prediction task and create a more generalised model.

Once trained, this model can then be transferred to a right-hand driving scenario which too includes a more generalised dataset. The right-hand side data will also include regional road signs, road markings etc. to ensure the most accurate results. We can confirm if a large network, created in this research, can handle a larger and more sophisticated dataset, or does it need to be adjusted in some significant way. A limitation of this approach maybe that a model trained on simulated data may not work as well on a real-world test case. Although this itself can be explored deeply as computer graphics are arriving at the point at which renders are very close to reality.

Another limitation of our approach is that we do not provide any speed information to our model. Our model takes in 2 inputs, images and speed, yet a 2D image may not contain enough

data to allow the model to learn speed from them. To address this, we could look at 3D convolutions, or possibly a recurrent approach. In a 3D convolutional approach, our model would take as input a 3D representation of our images. We could also batch *n* number of sequentially related images together and provide those as input too. This would give the model some information on 'depth' and thus speed over time. We could also use Recurrent Neural Networks (RNNs) as they perform well in a time-series based problem. RNNs can learn to predict an output based on previous inputs in a sequence. These can be used to also learn from a speed-based input as well. Paired with a feature extractor, i.e. a CNN, could provide some great results for steering angle prediction.

Apart from using just a speed input, apart from our image input, we can look at other sensor input types. There exist many sensors that can help us create a generalised model, i.e. LiDAR, event cameras, GPS etc. Looking at how the implementation of these can have positive effects on our MSE may be ideal. Some of these sensors have already been used by other researchers in our literature for the task of steering angle predict. But as humans rely on multiple senses, autonomous driving should be approached with the same idea.

As 2D image lack depth, or a method to measure depth, applying a stereo camera approach can be considered too. In this approach, multiple RGB cameras will be used, which can help us measure how far objects are from our vehicle. They can also allow us to get a broader field of view allowing more information as input. This can help the model learn various features, such as sharpness of bend, distance, etc. Some sensors can also be utilised, i.e. LiDAR, which specialises in depth sensing. The issue with such sensors is the real-world applications and the fact that they only serve the purpose of measuring depth. These inputs can then be fed into a CNN network as well for steering angle prediction.

In our variable sampling approach, in our attempt to simulate polder blindness, we hardcoded some rules. Of course, in such a process there is no learning involved, which we found to be quite limiting. To expand the scope, we can look at using a machine learning approach to simulate polder blindness. A model can be created which compares the current input to the

previous input for similarity. Siamese networks are capable of carrying out the task of comparing two images and showing the 'distance' between them. Based on a certain threshold and given the steering angle hasn't changed from 0 degrees for some period, the image may or may not be fed into the model for processing. As mentioned earlier, this method was out of scope for our current research, but it provides a way to look at polder blindness in depth. Some limitations of this approach may be the cost associated with it. As now there will be 2 models created, for comparing images and for predicting steering angles, it could become inefficient. Although this can be solved by using better hardware or better software development practises.

Our transfer learning approach currently uses our model to test if a left-hand drive model can be used in a right-hand drive scenario. As mentioned before, our model is trained using simulated data. One could expand this study further by experimenting with transferring a model trained on simulated data, to a real-world scenario. This direction can open up avenues for understanding the limitations of training on simulated data. In terms of compatibility, these both have a similar domain, steering angle prediction, and a similar target task. This shows that, theoretically, it is possible for transfer learning to be successful as both tasks are close on the task similarity tree. If successful, simulated data can accomplish more than a real-world dataset. We can look at training the model for edge cases not found in real-world data. We can also manipulate the environment for many different scenarios to generalise the model further.

Our current limitation is that we only trained on one scenario. If research is done to create a model that has trained on more scenarios or learn navigating a general map through CARLA, it can also be transferred. Research into the sensor types and number of sensors will also benefit for transfer learning. We could also look at using established networks, such as ResNets, to work as feature extractors and attach our own steering angle predictor. This would be beneficial as these networks have proven to be excellent at image recognition on the ImageNet dataset and may work well here too. Although this does bring in the question of the similarity and compatibility between the source and target task.

# References


Agarap, A. F. (2018). Deep learning using rectified linear units (relu). *arXiv*.

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 2481 - 2495.

Barjonet, P.-E. (2001). *Traffic psychology today.* Springer Science & Business Media.

Bebel, J. C., Howard, N., & Patel, T. (2004). An autonomous system used in the DARPA Grand Challenge. *Proceedings. The 7th International IEEE Conference on Intelligent Transportation Systems (IEEE Cat. No. 04TH8749)*, 487 - 490.

Behringer, R. a., Kubinger, W., Herzner, W., & Fehlberg, V. (2005). Rascal-an autonomous ground vehicle for desert driving in the darpa grand challenge 2005. *IEEE*, 644 - 649.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., . . . others. (2016). End to end learning for self-driving cars. *arXiv:1604.07316*.

Boureau, Y.-L., Ponce, J., & LeCun, Y. (2010). A theoretical analysis of feature pooling in visual recognition. *Proceedings of the 27th international conference on machine learning (ICML-10)*, 111 - 118.

Bucchi, A., Sangiorgi, C., & Vignali, V. (2012). Traffic psychology and driver behavior. *Procedia-social and behavioral sciences*, 972 - 979.

Caleanu, C.-D. (2000). Facial recognition using committee of neural networks. *IEEE*, 97 -100.

Campbell, M., Egerstedt, M., How, J. P., & Murray, R. M. (2010). Autonomous driving in urban environments: approaches, lessons and challenges. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences.*, 4649-4672.

Chen, Z., & Huang, X. (2017). End-to-end learning for lane keeping of self-driving cars. *2017 IEEE Intelligent Vehicles Symposium* , 1856-1860.

Chi, L., & Mu, Y. (2017). Deep steering: Learning end-to-end driving model from spatial and temporal visual cues. *arXiv preprint arXiv:1708.03798*.

Chowdhuri, S., Pankaj, T., & Zipser, K. (2019). MultiNet: Multi-Modal Multi-Task Learning for Autonomous Driving. *IEEE*, 1496 - 1504.

Crundall, D. E., & Underwood, G. (1998). Effects of experience and processing demands on visual information acquisition in drivers. *Ergonomics*, 448-458.

Dickmanns, E. D., & Zapp, A. (1987). A curvature-based scheme for improving road vehicle guidance by computer vision. *International Society for Optics and Photonics*, 161 - 168.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., & Koltun, V. (2017). CARLA: An Open Urban Driving Simulator. *Proceedings of the 1st Annual Conference on Robot Learning*, 1 - 16.

Du, S., Guo, H., & Simpson, A. (2017). Self-driving car steering angle prediction based on image recognition. *Department of Computer Science, Stanford University*.

Eraqi, H. M., Moustafa, M. N., & Honer, J. (2017). End-to-end deep learning for steering autonomous vehicles considering temporal dependencies. *arXiv*.

Evans, L. (1991). *Traffic safety and the driver.* Science Serving Society.

Fang, X., Luo, H., & Tang, J. (2005). Structural damage detection using neural network with learning rate improvement. *Computers & Structures*, 2150 - 2161.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* MIT Press.

Hatakka, M., Keskinen, E., Gregersen, N. P., Glad, A., & Hernetkoski, K. (2002). From control of the vehicle to personal self-control; broadening the perspectives to driver education. *Transportation Research Part F: Traffic Psychology and Behaviour*, 201 - 215.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *IEEE*, 1026 - 1034.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *IEEE*, 770 - 778.

Huang, Y., Cheng, Y., Chen, D., Lee, H., Ngiam, J., Le, Q. V., & Chen, Z. (2018). Gpipe: Efficient training of giant neural networks using pipeline parallelism. *arXiv*.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Keskinen, E. (1994). Why do young drivers have more accidents? Junge Fahrer und Fahrerinnen. *Referate der Ersten Interdiziplinaren Fachkonferenz, December 12-14, 1994 in Koln*.

Kim, J., & Park, C. (2017). End-to-end ego lane estimation based on sequential transfer learning for self-driving cars. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 30 - 38.

Koornstra, M. J. (1989). *Road safety and daytime running lights.* SWOV.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *NIPS*, 1097 - 1105.

Kuan, D. a.-C. (1988). Autonomous robotic vehicle road following. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 648 - 658.

Lau, M. M., & Lim, K. H. (2017). Investigation of activation functions in deep belief network. *IEEE*, 201 - 206.

LeCun, Y. a. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 2278-2324.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1898). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 541 - 551.

Lyons, M. J., Budynek, J., & Akamatsu, S. (1999). Automatic classification of single facial images. *IEEE transactions on pattern analysis and machine intelligence*, 1357 - 1362.

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. *ICML*.

Maqueda, A. I., Loquercio, A., Gallego, G., Garc'ia, N., & Scaramuzza, D. (2018). Event-based vision meets deep learning on steering prediction for self-driving cars. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5419 - 5427.

Mercimek, M., Gulez, K., & Mumcu, T. V. (2005). Real object recognition using moment invariants. *Springer*, 765 - 775.

Michon, J. A. (1985). A critical view of driver behavior models: what do we know, what should we do? In *Human behavior and traffic safety* (pp. 485 - 524). Springer.

Morton, R. D., & White, M. J. (2013). Revised reinforcement sensitivity theory: The impact of FFFS and stress on driving. *Personality and Individual Differences*, 57 - 63.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. *Proceedings of the 27th international conference on machine learning (ICML-10)*, 807-801.

Pan, S. J., & Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 1345 - 1359.

Pomerleau, D. A. (1989). Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 305-313.

Prakash, A. (2018, 05 17). *Machine Learning - Convolution for image processing.* Retrieved from Francium Tech | Medium : https://blog.francium.tech/machine-learning-convolution-for-image-processing-42623c8dbec0

Ricco, J. (2017, 06 16). *What is max pooling in convolutional neural networks?* Retrieved from Quora: https://www.quora.com/What-is-max-pooling-in-convolutional-neural-networks

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 386.

SAE International. (2014). *Automated driving: levels of driving automation are defined in new SAE international standard J3016.* SAE International .

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv*.

Simonyan, K., Vedaldi, A., & Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 1929-1958.

SWOV. (2012). *SWOV Fact sheet: Attention problems behind the wheel.* Leidschendam: SWOV.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., . . . Rabinovich, A. (2015). Going deeper with convolutions. *IEEE*, 1 - 9.

Thorpe, C., Hebert, M. H., Kanade, T., & Shafer, S. A. (1988). Vision and navigation for the Carnegie-Mellon Navlab. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 362 - 373.

Torrey, L., & Shavlik, J. (2010). Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques* (pp. 242 - 264). IGI Global.

Van der Molen, H. H., & B"otticher, A. M. (1988). A hierarchical risk model for traffic participants. *Ergonomics*, 537 - 555.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 1550 - 1560.

Yang, Z., Zhang, Y., Yu, J., Cai, J., & Luo, J. (2018). End-to-end multi-modal multi-task vehicle control for self-driving cars with visual perceptions. *IEEE*, 2289 - 2294.

Yanling, Z., Bimin, D., & Zhanrong, W. (2002). Analysis and study of perceptron to solve XOR problem. *The 2nd International Workshop on Autonomous Decentralized System, 2002*, 168 - 173.

Zamir, A. R., Sax, A., Shen, W., Guibas, L. J., Malik, J., & Savarese, S. (2018). Taskonomy: Disentangling task transfer learning. *IEEE*, 3712 - 3722.