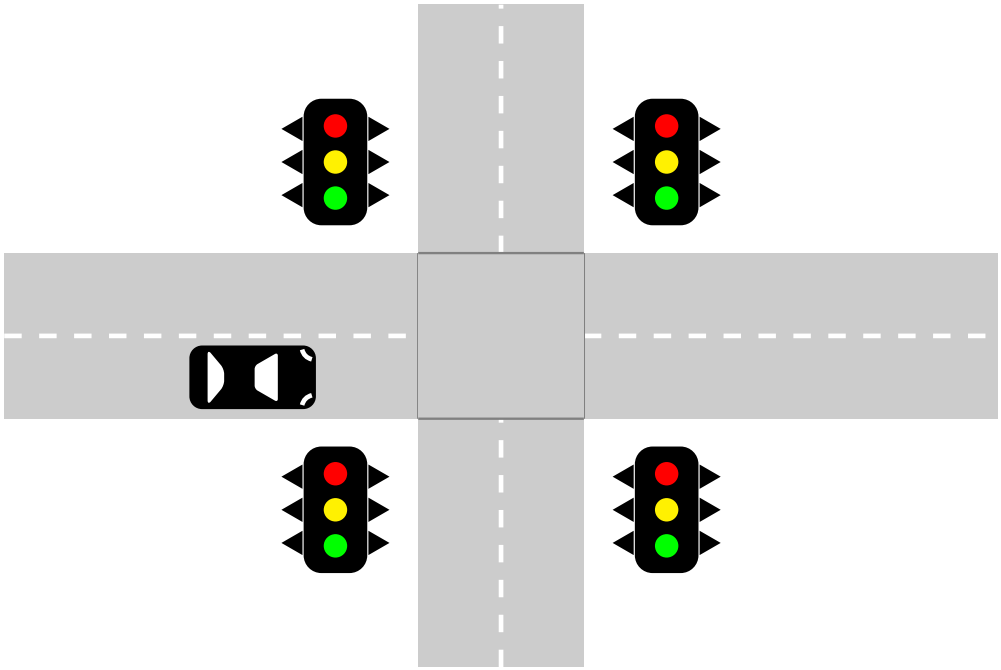
CHALMERS
UNIVERSITY OF TECHNOLOGY



Impact of Feature Representation for Imitation Learning in Autonomous Drive

Master's thesis in Complex Adaptive Systems and
Engineering Mathematics and Computational Science

MALIN DAHL

ELVIRA RAMLE

Department of Mechanics and Maritime Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Master's thesis 2019:29

Impact of Feature Representation for Imitation Learning in Autonomous Drive

MALIN DAHL
ELVIRA RAMLE

Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Applied Artificial Intelligence Research Group
Chalmers University of Technology
Gothenburg, Sweden 2019

Impact of Feature Representation for Imitation Learning in Autonomous Drive
MALIN DAHL, ELVIRA RAMLE

Supervisor: Josef Kindberg, CPAC Systems AB
Examiner: Peter Forsberg, Applied Artificial Intelligence

Master's Thesis 2019:29
Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Applied Artificial Intelligence Research Group
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Schematic figure of a car entering a signalised intersection.

Typeset in LaTeX
Gothenburg, Sweden 2019

Impact of Feature Representation for Imitation Learning in Autonomous Drive
MALIN DAHL
ELVIRA RAMLE
Department of Mechanics and Maritime Sciences
Chalmers University of Technology

Abstract

Autonomous drive in complex traffic scenarios is a demanding task to solve. With high-dimensional input data available, problems related to redundancy and irrelevance are often implicated, hence determining what features bring the most useful information is of vital importance. The purpose of this thesis is to investigate how different dimensionality reduction methods affect the driving performance and how to determine what features are most relevant.

Specifically, these questions were studied in a simulated environment where a car is manoeuvred using deep neural networks through a sequence of signalised intersections. Four different dimensionality reduction methods have been studied: choice of features based on reason, Principal Component Analysis, Auto-Encoders and Integrated Encoders. The results showed that the models which used a feature representation based on reason were shown to perform best.

Also, the weight distributions of a model using all available features indicated that influential features may be partially identified by studying the spread of the weights. Therefore, an approach is proposed where the choice of features should be based on reason as well as a study of the features' respective set of weights. In conclusion, establishing the most relevant feature representation is important since it may benefit the training of the models.

Keywords: Feature Representation, Deep Neural Networks, Imitation Learning, Dimensionality Reduction, Principal Component Analysis, Auto-Encoder, Autonomous Drive.

Acknowledgements

We would like to thank Cpac Systems AB for the opportunity to work on this problem and providing us with the necessary tools. A special thanks to our supervisor, Josef Kindberg at Cpac, for your creative ideas and feedback during the project. We would also like to thank our examiner Peter Forsberg for taking on this project, all the interesting discussions and your guidance throughout the thesis.

Malin Dahl and Elvira Ramle, Gothenburg, June 2019

Thesis Supervisor: Josef Kindberg, CPAC Systems AB
Thesis Examiner: Peter Forsberg, Applied Artificial Intelligence

Contents

List of Figures

List of Tables

Glossary

ADAM	Adaptive Moment Estimation	An optimiser posing as an alternative to stochastic gradient descent, based on adaptive estimates of lower-order moments.
AE	Auto-Encoder	A dimensionality reduction or noise cancelling method in the shape of a neural network.
CWS	Collective Weight Sum	A sum over all the weights, in absolute value, for a specific feature's input neuron.
IE	Integrated Encoder	A specific network structure where the dimensionality reduction is integrated with the rest of the network.
IL	Imitation Learning	A variant of supervised learning where the network tries to mimic an expert.
PCA	Principal Component Analysis	A statistical tool for dimensionality reduction based on the study of eigenvectors of the data.
ReLU	Rectified Linear Unit	An activation function used in neural networks.

1

Introduction

Humans have always had a demand for fast, flexible and convenient transportation. Through time, the requirement of having a human behind the wheel has been regarded as obvious and necessary, but during the past few years there has been a shift in opinion. The introduction of the concept of autonomous vehicles in society is apparent and the industry is exploding with ideas. Giants in the automotive industry are investing enormous amounts of money in development, among which we find Baidu, Google, BMW and Tesla only to mention a few [1].

The status of the industry today is described completely different depending on who you ask. In some regards and opinions, autonomous vehicles are already up and running without humans intervening in most or all situations. The standard system for classifying the level of how autonomous a vehicle is, has been put forward by the Society of Automotive Engineers (SAE). The scale ranges from level 0 to 5, where level 0 indicates no automation at all and level 5 implies that the vehicle is capable of handling all situations and is in practise driver-less at all times [2].

Asking Elon Musk, the CEO of Tesla Motors, the company will have commercial, autonomous cars of level 5 on the road by 2020 [3]. According to others, there are still problems to address and autonomous vehicle experts claim that serious safety issues are still apparent. The accident where a 49-year-old woman crossing the street with her bicycle was killed by a self-driving Uber and the Tesla car crash killing its driver when in autopilot mode, both during 2018, serve the purpose of proving the imperfections of the systems [4, 5].

However, the research and development go on and along the failures there are brilliant examples of what autonomous vehicles can do. One of the early role-models is the well-known ALVINN (Autonomous Land Vehicle In a Neural Network) from 1989, which consists of a neural network with one hidden layer. This relatively simple network structure managed to safely control a truck-like vehicle along a 400 metres long path in a wooded area at 0.5 m/s [6].

Today, the success of ALVINN is not particularly impressive, but that also shows how far the technique has come. More modern examples include the concept car 360c of Volvo Cars, Google's Waymo and the transportation focused truck Vera from Volvo Trucks [7, 8, 9]. Successful control of autonomous vehicles in specific environments is also reality. For example, NVIDIA managed to complete an 80 km long highway route, including lane changes and merges, with no human intervention, in October 2018 [10].

Furthermore, an important motivation for developing autonomous cars is that they have the potential of improving traffic safety drastically. The National Highway Traffic Safety Administration (NHTSA) concluded that 94 % of all serious crashes in USA the year 2016 were linked to human choices [11]. By using computer-based systems, reaction times can be reduced to practically nothing and general information processing like sign and map reading can be performed instantaneously. Also, by optimising the systems, the car may drive more environmentally friendly. Autonomous vehicles therefore shows great potential once the software and technology has been proven safe and reliable.

1.1 Methods and Challenges of Autonomous Drive

During the past decades, the choice of methods used in the automotive industry and autonomous drive has changed. With the massive improvement in computer force and data storage, the possibilities to train neural networks to perform tasks became realisable, even if the idea was born as early as in the 1940's [12]. The structures of the networks tend to increase and the concepts of deep learning and convolutional networks are used by many developers in the field.

In order to train the networks, huge amounts of data are required. By now, the hardware has become cheaper and is within reach of most developers, and many different sensors like cameras, radars and LiDARs are used to collect the data. Big data is not only possible to acquire, but also necessary for training and validating of the systems. However, with big data comes big problems as well. On one hand, having data from many different sources provides redundancy which can be used for evaluation and confirmation of the performance of the system performance. On the other, it also introduces correlation in the data and if all data is used without reflection, the risk of overfitting the network increases which can reduce performance greatly.

There are different approaches to reduce overfitting, where the most common is to simply collect more data. More data forces the network to generalise further and not over-adapt to specific data points. Other options are to use various dimensionality reduction algorithms, such as Principal Component Analysis (PCA) or different auto-encoder structures before or within the network structure [13]. Different dimensionality reduction methods may also help the network in prioritising important data and further generalise its decision-making.

When training a network, there are different strategies to choose from. The simplest variant is to use supervised learning, which implies training the network from labelled data. In some situations, acquiring labelled data is a challenge on its own. In autonomous drive, many decisions on how to control the vehicle are required in sequence and it is difficult to decide at every time instance which specific decision is the correct one. When possible, imitation learning is sometimes applied, meaning that the network aims to imitate an experienced expert when training.

1.2 Purpose

According to the Federal Highway Administration (FHWA), more than 50% of the combined total of fatal and injury crashes occur at or near intersections [14]. Intersections can therefore be seen as a complex scenario for human drivers and are difficult to comprehend for an autonomous vehicle as well. In order to be able to understand the intersections and make wise decisions, it is important to process all available features describing the surroundings correctly. Irrelevant information should be ignored while relevant information should be interpreted properly for the purpose of making optimal choices.

The purpose of this report is to investigate how different dimensionality reduction methods affect the driving performance of a vehicle when driving through signalised intersections. Also, an intention is that by investigating the methods and their projections, an understanding of what type of information is the most relevant when it comes to controlling a vehicle should be developed. Only features of physical quantities, such as the speed of a vehicle or the distance to an intersection will be considered in this thesis, as opposed to, for example, camera images or LiDAR data.

By developing a new understanding of what features are important for vehicle control in intersections, the data gathering can become more effective since irrelevant features do not have to be collected. Also, the training may converge faster since fewer but still relevant features help the network generalise.

The questions that will be discussed are:

1. How do different dimensionality reduction methods affect the model training and driving performance?
2. How to determine what type of information or set of features is the most relevant to take into consideration when driving through signalised intersections?

1.3 Scope

All data used for developing the model of the vehicle's behaviour is simulated in the open-source simulator Carla (Car Learning to Act) [15]. This puts a restriction on how vehicles can behave due to their deterministic behaviour. Using data from a simulated environment also restricts the scenarios the vehicle can be exposed to. For instance, all signalised intersections the vehicle is subjected to are four-way intersections where each incoming or outgoing road consist of two lanes with traffic lights in all directions. Moreover, the same car model is used in all simulations, starting at the exact same position and the weather conditions remained unchanged as well.

Additionally, the vehicle is instructed to always drive straight in every intersection in order to restrict the complexity of the environment and simplify the training. Even though the car is instructed to go straight in each intersection, the road is

sometimes curved which requires the vehicle to learn how to adapt and remain close to the centre of the lane. In total, the environment is set up to include three signalised intersections in sequence. To further simplify the task, no surrounding vehicles are allowed in the simulation. This reduces the need to learn how other vehicles behave and predict their intentions.

It is also assumed that all information provided to the vehicle, for example the velocity and distances, is measured correctly and sent instantaneously without any time-delays. The same is assumed for the control signal of the vehicle. This implies that the signals are assumed to be noise-free and can be fed directly into any network or model. Additionally, as a simplification, the networks used for controlling the vehicle all share the same structure, i.e. the same number of neurons, hidden layers and training parameters.

1.4 Related Work

When teaching a car how to drive, several studies have been carried out where imitation learning was used. Two articles were written by Bojarski et al. and another by Codevilla et al. [16, 17], where camera images were used in order to interpret the environment and the vehicle was then controlled based on that information. Both teams used a human driver to give feedback in all situations of how a human would react and then used that data in order to train their neural networks. The human therefore acted as an expert which the network tried to mimic. Bojarski et al only collected data from the real world and managed to learn lane and road following using less than 100 hours of driving data. Codevilla et al on the other hand collected data both from the real world and a simulated environment. Their model managed to travel to a random location, at least one kilometre from the start location, up to 88% of all runs after using roughly two hours of driving data.

A different variant of imitation learning was proposed by Kelly et al., where a so called Human-Gated Dagger algorithm was used [18]. Just as with the articles by Bojarski and Codevilla, the purpose was to teach a network to control a car. Dagger stands for data aggregation and the term human-gated implies that a gate decided whether the car was controlled by the network or the human. The idea with this algorithm was to train the network to choose the same actions as the human driver at all times, but also by letting a gate functionality determine which alternative that ultimately controls the car. This is to ensure safety at the same time as a wider state space is explored. A largely covered state space is important in order to ensure that the network is capable of handling all possible traffic situations.

Additionally, Curran et al. investigated the convergence-performance trade-off by using Principal Component Analysis (PCA) to reduce the dimensionality of the state space in a Super Mario game [19]. They performed PCA on the entire state space and removed the dimensions that gave minimal loss of information according to the analysis. The lower dimensional state space was then used in a reinforcement learning algorithm which managed to learn how to play Mario. As expected, the fewer dimensions that were used, the worse the game went, but the convergence occurred

much sooner. They observed that it was possible to learn the fundamental skills of Mario using a lower dimensional state space, but in some scenarios where Mario had a lot of surrounding enemies and needed to act quickly it was not enough.

Another common dimensionality reduction method is the use of an Auto-Encoder (AE). Just as with PCA, these can be applied in all problems which may benefit from dimensionality reduction. This method is especially good at noise reduction, but has also proven to learn different features in data compared to other classical dimensionality reduction methods [20, 21]. The applications are many and in an article by Taghanaki et al, auto-encoders were used when classifying mammography images in combination with direct supervised learning. The classification turned out to be more accurate when the dimension of the input data had been reduced [22].

1.5 Outline of Thesis

The remainder of the thesis is outlined as follows: firstly, Chapter 2 includes the necessary background theory in order to understand the remaining contents. The theory includes a basic introduction to artificial neural networks, explanations of several dimensionality reduction methods used throughout the thesis and the theory behind Imitation Learning (IL). Secondly, in Chapter 3, the method is described in more detail. More information about the available features, implemented models and how they are evaluated are specified. Thirdly, in Chapter 4, all of the results are displayed and are further discussed in Chapter 5. Lastly, the main results and conclusions are summarised in Chapter 6.

2

Theory

The theory behind the analysis is divided into three main parts. The first includes a brief introduction to artificial neural networks and relevant concepts needed to understand the results. The second explains the theory behind two of the dimensionality reduction methods, namely Principal Component Analysis (PCA) and Auto-Encoders (AE). Lastly, the concept of imitation learning is explained as well as its benefits and drawbacks.

2.1 Artificial Neural Networks

When it comes to interpreting the world and carrying out various tasks, no machine is yet better than the brain. The brain manages to generalise situations never experienced before and still make appropriate decisions. The suggestion of mimicking the brain when designing a machine therefore seems like a promising idea. A brain consists of numerous nerve cells, so called neurons, which receive electrical signals, process them and then send out new signals to other neurons via different connections and synapses [23]. The amount of information this structure can handle is vast, the processing speed is high and the generality of the various problems considered is apparent.

Artificial neural networks resemble the human brain in many ways. Just as with a real brain, the structures use neurons and connections to process and forward information. The construction of the neural networks make them outstanding in terms of generalisation compared to other more deterministic methods [24].

An example of a simple neural network is shown in Figure 2.1. The general structure consists of sequential layers, starting with the input layer, followed by hidden layers and resulting in the output layer. Firstly, the input neurons marked with $X_{1,\dots,k_{\max}}$ assume values observed from the world in some way. They could be pixel values of an image or the position and speed of a car. Between the input layer and the first hidden layer, there are connections with corresponding weights w_{kj}, determining the value of the hidden neurons marked with $V_{1,\dots,j_{\max}}$. The values of the neurons in the hidden layer are computed as in Equation (2.1)

$$V_j = g\left(\sum_k w_{kj} X_k - \theta_j\right), \tag{2.1}$$

where $g(\cdot)$ is the activation function, w_{kj} the weight for the connection between neurons X_k and V_j and θ_j is the threshold for the neuron V_j, which is often regarded as a weight as well.

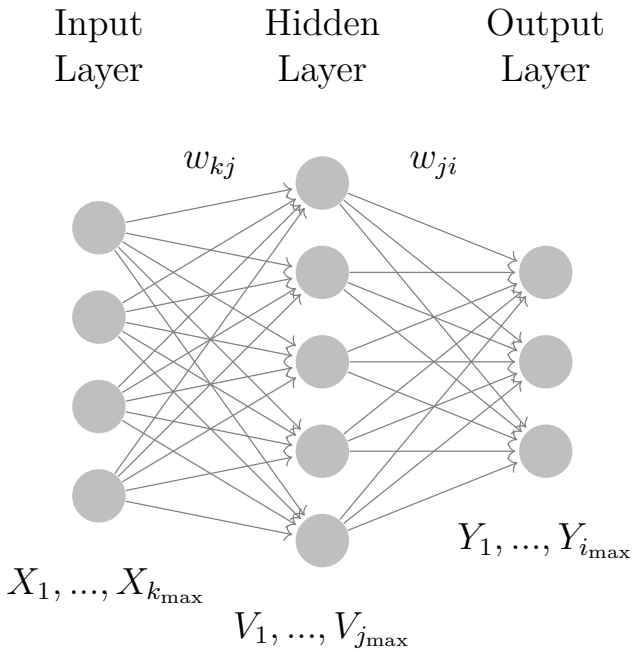


Figure 2.1: Simple example of a neural network, where $X_{1,...,k_{\max}}$ define input signals, $V_{1,...,j_{\max}}$ the values of the neurons of the hidden layer and $Y_{1,...,i_{\max}}$ the values of the output neurons. The values of the hidden and output layers are computed using the weights w_{kj} and w_{ji} respectively.

The choice of activation function $g(\cdot)$ depends on the problem, but is a non-linear function often taking values between -1 and 1 or similar. Common examples are the sigmoid function or Rectified Linear Unit (ReLU). The weights are what defines the network, which are updated iteratively as the training progresses. The values of the neurons in the output layer are computed similarly as in Equation (2.2)

$$Y_i = \tilde{g}\left(\sum_j w_{ji} V_j - \theta_i\right), \tag{2.2}$$

where $\tilde{g}(\cdot)$ may be the same activation function as before, or a different one depending on the problem. Note however that it is possible to have many more hidden layers in a network and that the values are computed accordingly with the same basic idea of using the values of the previous layer when computing the next.

2.1.1 Impact of the Weights

As depicted in Figure 2.1, each input neuron is connected to a set of weights, which further connect to the neurons in the upcoming hidden layer. These weights can take any scalar value as long as it helps the network distinguish relevant features.

In the simplest case, if all weights connected to one specific feature's input neuron are zero, then this feature has no impact on the output of the network, which can be concluded from Equation (2.1). Contrarily, if some weights of a specific feature are large compared to the weights of other features, then this feature is more influential.

In practise, however, the weight distributions are often more complex which makes this type of precise conclusions difficult to draw. The difficulty increases even further when the networks contain many hidden layers, since the weights in the subsequent layers affect the influence of a feature as well. The value of a specific feature is propagated through the network and is intertwined with the effect of all other features which makes direct causality difficult to determine.

What can instead be done is to analyse the distribution of the weights directly connected to the features, i.e. the weights in the first hidden layer, to see if any trends are visible. If many weights are large in absolute value for a specific feature, the chance of this feature having a big impact on the final network output increases. On the other hand, if most of the corresponding weights for a feature are small, then the chance decreases since other features with larger weights will tend to dominate.

Also, important to note is that the data set, on which the network is trained upon, needs so be standardised to mean zero and variance one beforehand if the weights are analysed. This is due to that if the set of weights are compared, they must have the same prerequisites. If the features themselves vary largely in size, so will the weights and the different sets of weights will therefore be incomparable. By standardising the data, comparability is ensured.

2.1.2 Training the Network

In one way or another, the purpose of a network is to carry out a task. How the training of the network is carried out can be done in several ways, but the most common way is through supervised learning. When training the network, a measure is also required to assess how well the task is carried out. In supervised learning, one defines a loss function that measures the difference between the correct value and the output of the network.

The loss function is also used when training the network with backpropagation, which can be done with one of several iterative algorithms, such as Stochastic Gradient Descent (SGD) or Adaptive Moment Estimation (ADAM) [25]. How the loss function is defined also depends on the problem. The mean squared error is suitable for continuous signals, whereas a binary measure is more appropriate for classification problems. For classification problems, a probability measure is also common when computing the loss, where one example is the softmax function.

In general, it is difficult to determine for how long a neural network should be trained, namely for how many iterations the weights should be updated. If the network is trained for too many iterations, there is a risk of overfitting which means that the network has learned the data by heart and does not generalise well. One

way to avoid overfitting is by implementing early stopping. Early stopping is a rule, which can be defined in many different ways, which determines when the training of neural network should stop. A common rule for early stopping is to stop the training once the computed loss does not decrease for some number of iterations.

An additional and different approach to handle overfitting is by introducing dropout in the hidden layers. When dropout is used, a part of the neurons and their corresponding weights and thresholds are not updated for one iteration. This forces the network to generalise further and try to find new connections that were not utilised to their full extent before.

2.2 Dimensionality Reduction Methods

When handling various tasks in autonomous drive, large data sets are required to cover the complexity of the situations studied. Often there are many streams of data as well, resulting in a high-dimensional feature space. The risk of having redundant data is prominent, but still one cannot easily skip various features due to the risk of losing important information. Studying which features and what combinations produce the most valuable input for the algorithm or network is an important part of the modelling process which may help the decision-making tremendously.

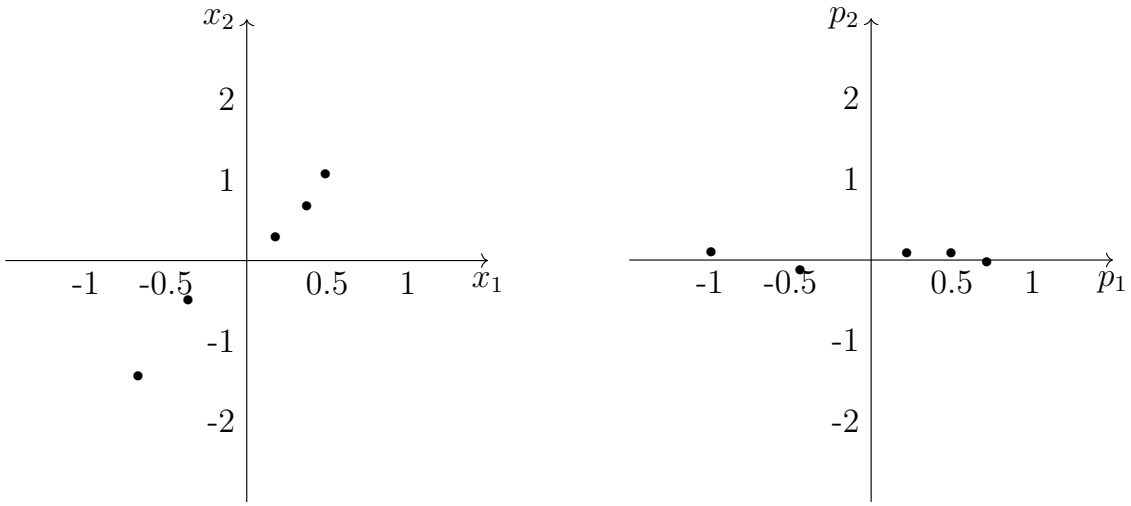
When a set of features have been presented as possibly relevant input data, there are many dimensionality reduction methods that can be applied. In the following sections two different approaches are presented: a linear projection in the form of Principal Component Analysis (PCA) and a non-linear variant using neural networks and Auto-Encoders (AE).

2.2.1 Principal Component Analysis

One way to analyse the importance of features is through Principal Component Analysis (PCA). The most common use of PCA is for dimensionality reduction, but the variance of the data can also be studied. In general terms, PCA is a statistical tool where the feature vectors are projected onto the eigenvectors with the largest eigenvalues. This implies that a principal component projection is linear and can be computed with simple matrix multiplications.

When performing PCA, the data points are sorted row-wise with each measured feature column-wise; call this data matrix $\mathbf{X}$. As a first step, the columns of the matrix are standardised to mean zero and variance one in order for the various features to have equal importance. Secondly, the product $\mathbf{X}^\intercal\mathbf{X}$ is computed. Thirdly, it is diagonalised into $\mathbf{X}^\intercal\mathbf{X} = \mathbf{P}\mathbf{D}\mathbf{P}^{-1}$, where the diagonal matrix $\mathbf{D}$ contains the eigenvalues of $\mathbf{X}^\intercal\mathbf{X}$ and $\mathbf{P}$ the corresponding eigenvectors. For convenience, the eigenvalues are sorted in descending order and the columns of $\mathbf{P}$ are organised correspondingly. Due to the fact that the matrix $\mathbf{X}^\intercal\mathbf{X}$ is symmetric and positive semi-definite, it is always possible to decompose it into $\mathbf{P}\mathbf{D}\mathbf{P}^{-1}$.

The eigenvalues and eigenvectors can now be studied in many ways. One way is to let the data be projected onto the eigenvectors to visually study the actual dimensionality of the data. By actual, this implies that if for example two original features of the data are collinear, then there will be one dimension in the projection that is useless and its values are more or less equal to zero. This implies that this extra dimension adds no variance to the data. The projection is performed by computing $\mathbf{X}^{\text{proj}} = \mathbf{PX}$, where $\mathbf{X}$ is the standardised data. An example of such a projection is shown in Figure 2.2, where the data is represented by the two variables x_1 and x_2 in the left graph and then by the principal components p_1 and p_2 in the right. In this example it is clear that there is more variance in the data for the first component p_1 than for p_2, which is due to the fact that x_1 and x_2 are linearly correlated.



(a) Standardised, original data, represented by the variables x_1 and x_2.

(b) Projected data onto the principal components p_1 and p_2.

Figure 2.2: A comparison between before (to the left) and after (to the right) a principal component projection for an example data set consisting of measurements of the variables x_1 and x_2. In this example, the first principal component p_1 provides most of the variance in the data and the component p_2 contributes only a little.

Another way to use the results from PCA is to determine the dimensionality and spread of the data. In principle, due to commonly occurring noise in measurements, all features in the matrix $\mathbf{X}$ represent one dimension more or less, but as shown in the example in Figure 2.2, some dimensions have much smaller variance. What is usually done is that the cumulative sum of the eigenvalues' proportions is computed. As mentioned, the diagonal of the matrix $\mathbf{D}$ is sorted in descending order resulting in a set of eigenvalues $\{\lambda_i\}_{i=1}^{n}$, where n is the number of eigenvectors in total. The cumulative partial sum is calculated as in Equation (2.3):

$$V_k^{\text{prop}} = \frac{1}{\lambda_{\text{sum}}} \sum_{i=1}^{k} \lambda_i, \qquad k < n \tag{2.3}$$

where k is an integer between 1 and n and λ_{sum} is the sum of all eigenvalues. If one

requires that a proportion η of the total variance needs to be explained, say 95%, then k is determined through Equation (2.4):

$$V_k^{\text{prop}} > \eta \tag{2.4}$$

by finding the smallest k that satisfies this inequality.

In summary, by using PCA the data is projected onto a smaller number of features which manages to explain almost all the variance in the data. As previously explained, this can help the training of the network a lot since it reduces the number of weights that need optimisation.

2.2.2 Auto-Encoders

As an alternative to linear methods for dimensionality reduction like PCA, non-linear mappings from a larger dimension to a smaller can be considered. One such method is an Auto-Encoder (AE), which is a specific type of neural network. An auto-encoder consists of two main parts: the encoder and the decoder. The encoder takes the features as input, for example the position and velocity of a vehicle, and produces a lower-dimensional internal state. This internal state, called the hidden state, is thereafter propagated through the decoder which tries to mimic the original input state.

A simple example of an auto-encoder is presented in Figure 2.3. In this auto-encoder, the encoder consists of one hidden layer where the nine input features are transformed into six and finally to only three in the hidden state layer. The decoder has a similar structure, but mirrored. Note that auto-encoders can be much more complicated than this, using for example Long Short Term Memory (LSTM) or convolutional layers [26, 27].

The purpose of all dimensionality reduction methods is to find a representation of a set of features with fewer dimensions. This implies that even though the whole network structure, as in the example in Figure 2.3, is used for training, only the encoder part is used when the actual dimensionality reduction is performed. When the auto-encoder has been trained, the input data is propagated through the encoder and the hidden state is computed. The values of the hidden state are now the values used when performing a specific task, for example controlling a vehicle.

Furthermore, one shared characteristic between all auto-encoders is that when it is trained, the algorithm tries to minimise the error between the input layer and the output layer. This implies that a difference between an auto-encoder and a regular neural network is that no labelled data is required for training, but the input data itself serves as comparison. An auto-encoder is therefore an unsupervised learning method as opposed to supervised learning where labelled data is used.

A drawback with auto-encoders as dimensionality reduction method is that the number of hidden states, and number of neurons in the intermittent layers, need to be determined beforehand. An iterative and trial-and-error approach can be applied where the behaviour of the loss function is studied depending on how many neurons that are used.

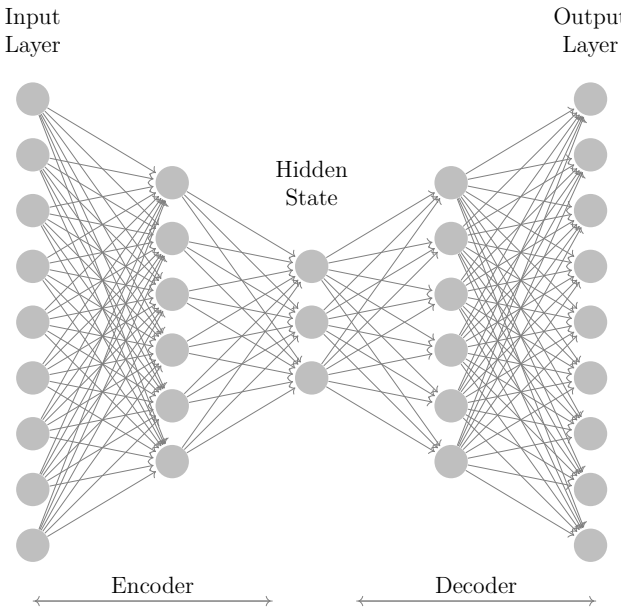


Figure 2.3: A simple example of an auto-encoder. The encoder part consists of one hidden layer where the input is mapped from nine input neurons, to six and finally to three in the hidden state. The decoder has a similar, but inverse structure which tries to reconstruct the original input state.

2.3 Imitation Learning

When trying to make a network learn, extensive amounts of data are needed and gathering all this data is in general expensive and time-consuming. It can also be difficult to gather enough data so that it covers all relevant cases. Also, if the task is to control a car, this involves making sequential decisions which further aggravates the data collection. Gathering a data set that includes most possible states with their corresponding correct decisions becomes nearly impossible due to the infinite number of states.

What is done instead, when appropriate, is try to mimic some sort of expert, which in the case of car control could be a human driver or an advanced autopilot. The field of training a network with the use of an expert is commonly referred to as Imitation Learning (IL). Imitation learning is a variant of supervised learning, but differs in how the data is gathered. In regular supervised learning, the data points are usually independent where a set features are used for prediction or classification. For example, the data set could include images of cats and dogs where the task is to determine which images include which animal. In imitation learning however, the data is instead gathered by observing the expert perform the task over time and store the features and correct actions respectively, on which the network is thereafter trained with.

A schematic figure depicting the workflow of imitation learning can be seen in Figure 2.4. A set of features at a time instance t, i.e. the state s_t, is collected from the environment. This information is passed onto the expert and the network which both

aims to find the best action a_t and $\tilde{a}_t$ to take respectively. The difference in choice of action between the expert and the network is used for computing the loss, which is thereafter used when updating the network and improving its performance.

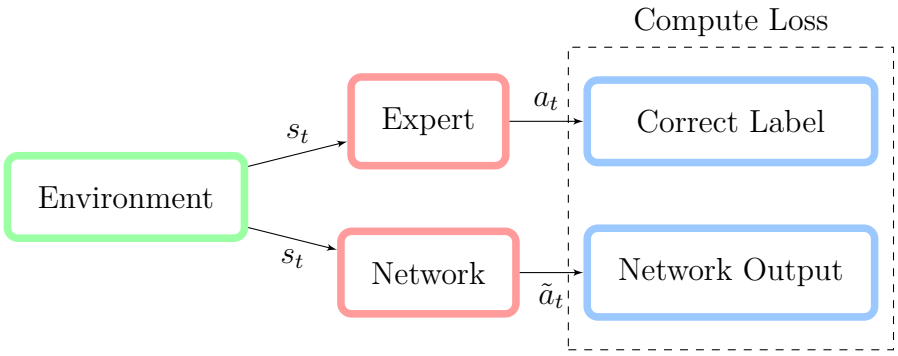


Figure 2.4: A flowchart of the process of imitation learning. The environment produces a state s_t which is used by the expert and the network to propose a suitable action a_t and $\tilde{a}_t$ respectively. The difference in choice of action between the expert and the network is used for computing the loss which defines how the network should be updated.

2.3.1 Benefits and Drawbacks

The strength as well as drawback with imitation learning is that if an expert is used when collecting data, only the states that the expert experiences are used to train upon. On one hand, this is helpful since unrealistic states do not need to be explored and only the perfect and relevant states are selected for training. On the other, this means that if the expert always carries out the task perfectly, the network will not learn how to behave in new and unseen situations. For example when learning to control a car, if the expert always stays within the lane the network will be confused if a small shift from the known position occurs and it will most likely not know how to behave.

Also, since imitation learning is a variant of supervised learning, the networks are trained relatively fast. By simply comparing network output with the expert's recommendation, the network learns to distinguish relevant features quickly. The disadvantage though is that the network can never be better than the expert itself. In the case where a robot tries to mimic a human, this can be good enough, meaning that an almost as good robot can be very useful even if a real human being would perform the job better.

However, in the case where a programmed autopilot is used, the network will only represent a poor and less competent version compare to a human. An alternative that helps fix this problem is, if possible, to combine the knowledge of different experts with different strengths when training the network. Also, an option is to use imitation learning for pre-training of the network and then continue with other methods, such as reinforcement learning or other unsupervised learning techniques.

3

Method

Controlling a vehicle based on different types of information is the primary task in this report. The purpose is to investigate how different dimensionality reduction methods affect the driving performance of the vehicle when driving through signalised intersections. The four various methods used for dimensionality reduction are choice of features based on reason (from now on only referred to as Reason), Principal Component Analysis (PCA), Auto-Encoder (AE) and Integrated Encoder (IE).

Regardless of which dimensionality reduction method is being used, the task of the model is to take a set of features at a time instance as input and produce a choice of action as output. This action then determines how the car drives. A schematic figure of the traffic scenario studied is presented in Figure 3.1. The aim for the car is to drive through three signalised intersections in a controlled manner following traffic regulations, staying in its lane and stopping at red lights.

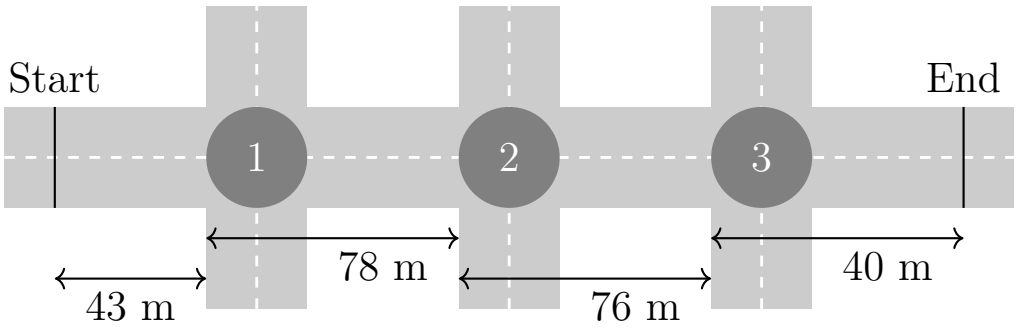


Figure 3.1: A schematic figure of the sequence of intersections that the car needs to pass in order to reach the end. The car is initialised at 43 metres before the first intersection and drives straight through the remaining intersections and finishes 40 metres after the last one.

The structure of this section consists of a description of the analysed features, the simulation environment used when training and testing the models as well as an explanation of how the data is gathered. Additionally, the network architecture that was used is presented. In the sections that follow, the models using different dimensionality reduction methods are described along with methods for comparing their respective performance.

3.1 Description of Features

In order to control a car in a complex environment like signalised intersections, information about the environment needs to be provided. The various features that were considered for further analysis are listed in Table 3.1, which make up 14 different features in total.

Table 3.1: The 14 features considered that were extracted from the simulation environment. The x, y and z subscripts denote in what direction the feature was measured. The simulated town which the car is subjected to lies in the xy-plane and the z-axis is perpendicular to the plane and points in the direction of the sky.

Features	
Velocity	v_x, v_y, v_z
Acceleration	a_x, a_y, a_z
Angle to lane	θ
Pitch angle	ϕ
Distance to centre of lane	d_x, d_y, d_z
Distance to next traffic light	t_x, t_y
State of next traffic light	$s = \begin{cases} \text{Green} \\ \text{Red} \end{cases}$

Firstly, the six velocity and acceleration features were all measured in the global coordinate system of the simulation environment. Secondly, the angle to lane was measured as the heading angle relative to the direction of the lane. This implies that angle to lane is based on the yaw angle of the vehicle. Thirdly, the pitch angle, the three distance to centre of lane as well as the two distance to next traffic light features were also defined according to the coordinate system of the simulator. Lastly, the state of the next traffic light is simply the colour of the next upcoming traffic light. Important to note is that a yellow traffic light was regarded as red as a simplification. This made the state of the next traffic light either red or green, i.e. a binary variable, whereas all other variables were continuous.

3.2 Simulation Environment

In order to train and try the different models, a simulated environment was required and the simulator Carla of version 0.9.3 was used. Carla stands for Car Learning to Act and is an open-source software with the purpose of aiding developers when training and validating different autonomous driving systems [15]. Carla provides up to five different environments, so called towns, with different characteristics. The fifth one includes several traffic light intersections in sequence, which is the reason for it being selected in this project. A bird's-eye view of the environment can be seen in Figure 3.2. To run the simulator, a computer with a GPU GeForce GTX 1050 Ti was used.

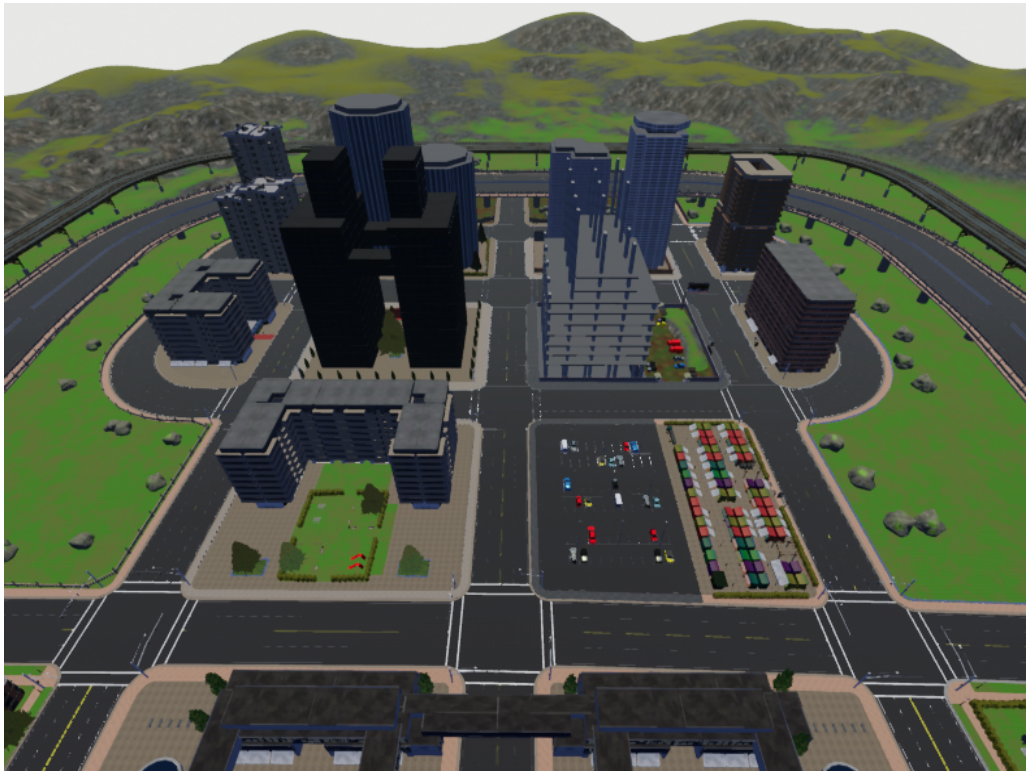


Figure 3.2: A bird's-eye view over the environment used in the simulator Carla. The road stretching from bottom to top in the middle of the image was used when training and running the models [28].

When running the simulator, many different types of data can be accessed. Camera images of the surrounding environment as well as simulated LiDAR data are provided, but also signals from both fixed and moving objects. The various features that were extracted in Carla are presented in Table 3.1, which make up 14 different features in total.

The way the car is controlled in Carla is through two signals: the acceleration and the steer, both of which are discretised. The acceleration is split into five categories (± 1.0, ± 0.5 and 0) and the steer into eleven (± 0.6, ± 0.3, ± 0.1, ± 0.05, ± 0.025 and 0). This results in 55 different classes in total. Note that the values are unitless and relative so that $+1.0$ implies full gas or full right turn and -1.0 implies full break or full left turn.

Additionally, an important feature in Carla is the built-in autopilot. When the autopilot is activated, the host vehicle drives in a correct manner through the town, stops at traffic lights and stays in its lane. Also, it is possible to specify where the autopilot should drive by setting its final destination. Lastly, when in autopilot mode, the control signal can be accessed and replaced if necessary, for example if one wants to introduce noise to the signal.

3.3 Data Gathering

The data used for training the neural networks of the various models was gathered from the simulator Carla, described in Section 3.2, using the built-in autopilot. The autopilot started 37 metres before the first traffic light intersection as shown in the schematic sketch in Figure 3.1. The speed of the vehicle was randomly sampled from a uniform distribution between 0.2 and 30 km/h.

The autopilot was then instructed to drive straight through all three intersections until the end destination, also shown in Figure 3.1. Once the car reached the end, it was removed from the simulation environment and a new car using the same autopilot was spawned at the same starting point, but with a new random start velocity. All events from when a single car was first created until it was removed is hereby referred to as an episode. All cars were of the same model, a 2017 MKZ Lincoln, the weather conditions were identical and the car drove down the same road for simplicity.

Also, the traffic light times for green, red and yellow light in all intersections were randomly assigned for each new car that was created, with values taken from different uniform distributions. The green light was assigned to be between 4 and 7 seconds, the red light between 1 and 3 seconds and the yellow light between 0.5 and 2 seconds. It was also randomly assigned which one of the four traffic lights in each intersection started with green light.

Since the autopilot drove according to traffic regulations and was good at driving near the centre of the lane, noise needed to be introduced in order for the network to learn how to recover from situations if it made an incorrect control decision. The noise would ensure that a larger state space was explored and incorporated into the models' network. The autopilot sends approximately 30 control signals to the vehicle's control system every second. The noise was introduced every 1297^{th} signal for 10 consecutive signals. The choices of these noise parameters were based on visual inspection as well as wanting a prime number in order to reduce the risk of cyclic behaviour. The noise consisted of increasing or reducing the steer signal to make the car either magnify or diminish the steer angle. The magnitude of the noise was randomly sampled from a normal distribution with mean zero and standard deviation 0.5.

Each time the autopilot gave a control signal to the vehicle, both the current state of the environment, described by all 14 features listed in Table 3.1, and the control signal, consisting of the steer and acceleration, were saved in pairs. In total 1 505 454 data points were collected and used for training the models' neural networks. These data points make up approximately 54 hours of driving.

3.4 Model Network Architecture

The neural networks used for controlling the vehicle in the four different models had a common architecture, which is displayed in Figure 3.3. As can be seen in

the figure, the networks consisted of two hidden layers with 600 and 900 neurons respectively. The output layer had 55 neurons that represented the 55 discretised control signals, which were introduced in Section 3.2. The black box in the figure was defined differently for each model and determined what features to use and how they were processed. All layers were fully connected and both hidden layers with 600 and 900 neurons used the ReLU activation function and a dropout of 50%. The weights in the network were updated using the ADAM optimiser and the softmax cross entropy loss function. For comparison, the accuracy of the network was also computed as the fraction of how many correct classifications the network managed to perform.

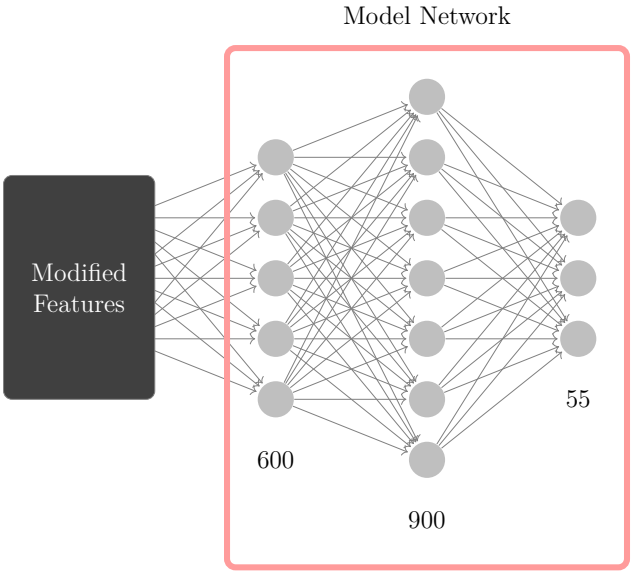


Figure 3.3: The shared network structure between the different models considered. A set of modified features, which vary between the models, were used as input. These were followed by two layers with 600 and 900 neurons respectively and the output layer consisted of 55 neurons, each representing one action.

All neural networks were implemented and updated using Chainer's neural network framework and trained using the same data set, which was gathered as explained in Section 3.3. The input data was standardised to mean zero and variance one and the output signal translated into its corresponding discrete action. The data set was split randomly into a training and validation set where the training set consisted of 80% of all data and the validation set of the remaining 20%. Early stopping was implemented and stopped the training when the validation loss had not improved over three sets, where each set consisted of the mean over five epochs. The batch size was set to be 512 and the model was saved every tenth epoch. The latest updated network, saved right before the early stopping was activated, was chosen as the optimal model and used for further evaluation.

The car was controlled using the 55 outputs of the network in the Carla simulator. In order to get a smooth control signal and avoid using a single discretised control signal, which would narrow down the action space, an aggregated control signal was created. The outputs from the neural network, which had been mapped by

the softmax function, represented a normalised probability distribution of how good each action was to perform. The aggregated, smooth control signal was constructed as the sum over all 55 discretised actions multiplied with their respective probability as described in Equation (3.1)

$$\mathbf{a}_{\text{smooth}} = \sum_{i=1}^{55} p_i \cdot \mathbf{a}_{\text{i}}, \tag{3.1}$$

where p_i is the probability of action i and $\mathbf{a}_i$ is the discrete control signal which neuron i represents consisting of one acceleration signal and one steering signal as described in Section 3.2.

3.5 Dimensionality Reduction Models

Four different model types were implemented for reducing the dimensionality and selecting features for controlling a vehicle in a simulator. The four model types included feature choice based on reason (Reason), Principal Component Analysis (PCA), Auto-Encoder (AE) and Integrated Encoder (IE). More details on how the models were designed are presented in the upcoming sections. Mutual for all models is that once the features had been modified, they were given as input to the model network introduced in Section 3.4.

The features that were available for the network to use are the 14 features listed in Table 3.1. Since the feature containing the state of the upcoming traffic light is binary while the rest of the features are continuous, the traffic light state was extracted and processed separately for all models except the models referred to as Reason.

In order to get an accurate comparison between the models, two models for each dimensionality reduction method were created: one which reduced the dimensionality to ten features and another to six. An additional model was created for the Reason models, which used all 14 available features. This model was created with the purpose of serving as a reference model and to evaluate how well a deep neural network could perform using all accessible information.

3.5.1 Reason

As the name of the method implies, the features that were chosen were selected based on the authors' own reasoning. Table 3.2 shows what features were included for each of the two created models. A marked green box in the table indicates that the feature was used and an unmarked grey that it was excluded. The selected features for each model were used as input to the model network without any additional processing.

The argument for removing the specific features from the model Reason 10 was to remove the ones that were thought of to give the least important information

Table 3.2: The features that were included for the two Reason models. A marked green box indicates that the feature was used for the corresponding model, while an unmarked grey box represents an excluded feature. There were 14 available features in total, namely the same features as listed in Table 3.1. Reason 10 used ten features and Reason 6 used six features.

Features		Reason 10	Reason 6
Velocity	v_x	•	•
...	v_y	•	•
...	v_z		
Acceleration	a_x	•	
...	a_y	•	
...	a_z		
Angle to lane	θ	•	•
Pitch angle	ϕ		
Distance to centre of lane	d_x	•	
...	d_y	•	
...	d_z		
Distance to next traffic light	t_x	•	•
...	t_y	•	•
State of next traffic light	s	•	•

about the surroundings to the vehicle. Since the road that traversed the three intersections lies in the xy-plane and never inclines, all features that were measured in the direction of the z-axis were removed. This included the z-directed values of the velocity, acceleration, the distance to the centre of the lane and the pitch angle of the vehicle.

For the model Reason 6, the same features that were removed in Reason 10 were discarded with the same explanation. The remaining four features that were removed were the acceleration and the distance to the centre of the lane in the xy-plane. The reason for removing the acceleration was that as few features as possible were to be used and the velocity was chosen to represent the kinematic properties primarily. The distance to the centre of the lane was removed with the motivation that since the car was always initially positioned at the centre of the lane, the car's angle to the lane should be enough information in order to continue staying close to the centre. Worth noticing is that if the vehicle was to be positioned further away from the centre, but in the same direction as the road, this reasoning would not hold. Also, the angle to lane implies one extra dimension whereas the distance to lane implies two which also motivates the choice.

The velocity was retained for all models with the motivation that it is important for the car to know how fast and in what direction it is going. Without knowing the direction it would have been difficult for the vehicle to compensate and determine how to angle the wheels. The angle to the lane was kept since the car needed to have some type of knowledge about how the car was positioned relative to the road. The distance to the next traffic light and its state were also kept since they were

considered essential to know in order to not run any red lights and to stop at a correct distance from the traffic light when necessary.

3.5.2 Principal Component Analysis

When using Principal Component Analysis (PCA) to reduce the dimensionality, all 14 features in Table 3.1 were considered. However, the state of the next traffic light was handled separately and the principal component projection was performed for the remaining 13 features. The data was ordered in a matrix $\mathbf{X}$ with 13 columns each representing one feature. The matrix $\mathbf{X}^\intercal\mathbf{X}$ was computed and the corresponding eigenvalues and eigenvectors calculated, as described in Section 2.2.1.

Two models were created using PCA as dimensionality reduction method and the projection was performed by projecting the data matrix $\mathbf{X}$ onto the either nine or five eigenvectors with the highest corresponding eigenvalues. By using the largest eigenvalues, it is guaranteed that the eigenvectors explains as much of the total variance as possible. When combined with the traffic light state, the set included ten or six features in total.

Figure 3.4 shows the information flow in a PCA model. The projection with PCA was performed on All Features, which resulted in a set of Modified Features. These were then combined with the traffic light state and sent as input to the Model Network, which was trained as described in Section 3.4.

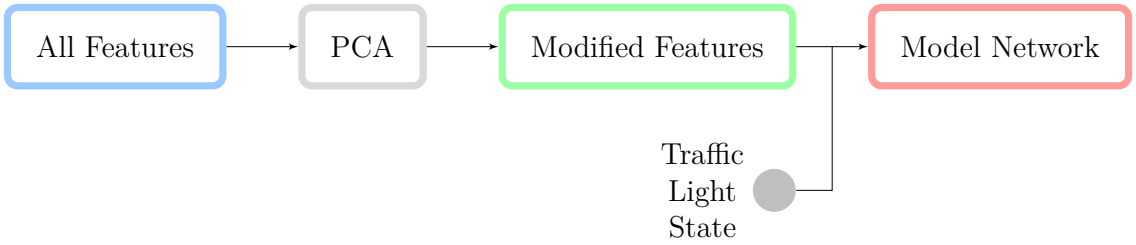


Figure 3.4: The information flow in a PCA model. The block All Features includes all features listed in Table 3.1, except the traffic light state. These features are then transformed using PCA into a set of Modified Features. The Modified Features are merged with the traffic light state and sent as input to the Model Network, described in Section 3.4.

3.5.3 Auto-Encoder

Two models were created that used Auto-Encoders (AE) to reduce the dimensionality; one model used ten features and the other six. Both auto-encoders had 13 input neurons which included all features except the state of the upcoming traffic light. The number of neurons in the hidden state were set to be either nine or five for the two auto-encoders instead of ten or six to compensate for the removal of the traffic light state feature. The upcoming traffic light state was instead added as an input neuron to the model network, unprocessed by the auto-encoder, in arrears for

both models. The number of neurons in the first hidden layer of the encoder and decoder part of the auto-encoder was set to be the mean value of the number of input neurons (13) and the number of neurons in the hidden state (9 or 5), resulting in either eleven or nine neurons.

All layers in the auto-encoders were fully connected without dropout and all but the output layer used the ReLU as activation function. All other parameters for training the auto-encoder were identical to training of the model network, namely using the ADAM optimiser, early stopping and the softmax cross entropy loss function. The same data set and batch size were used. The network that was last updated, before the early stopping interrupted the training, was used as the model's auto-encoder.

Once the auto-encoders had been trained, the modified features computed using the encoder part were used as input to the model network. A schematic sketch of the entire process can be seen in Figure 3.5. The first flow starting at All Features and ending at the Decoder represents the information flow through an auto-encoder network. The Modified Features represent the hidden state in the auto-encoder. The second flow from All Features to the Model Network represents the flow of information used for determining how a vehicle is controlled. The training of the model network began once the auto-encoder had finished its training. During the model network training, only the encoder part of the auto-encoder was used and all the corresponding weights remained fixed.

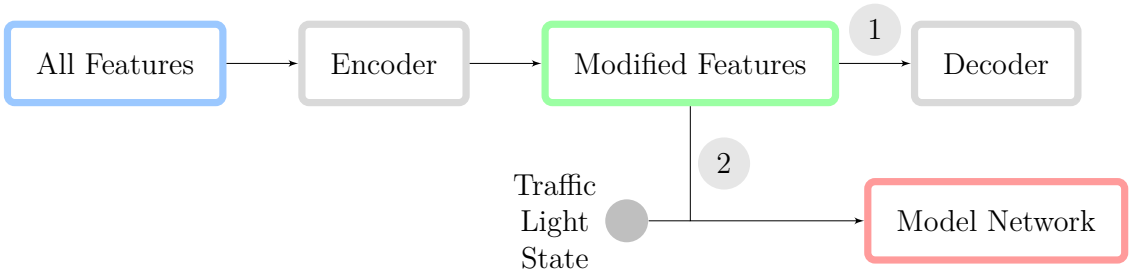


Figure 3.5: The information flow in an auto-encoder model. The model contains two flows, referred to as the first and second flow which are indicated by the numbers in the figure. The first flow goes between All Features and the Decoder and represents the flow of an auto-encoder as seen in Figure 2.3. The second flow starts at All Features as well, but diverges after the Modified Features and ends at the Model Network. Modified Features represent the hidden state in an auto-encoder and the Model Network is the same neural network as in Figure 3.3 which uses the Modified Features and the Traffic Light State as input.

3.5.4 Integrated Encoder

An Integrated Encoder (IE) is inspired by the auto-encoders and is basically a different network structure used instead of the original model network, introduced in Section 3.4. The alternate structure consists of taking the encoder part of an auto-encoder, as seen in Figure 2.3, and prepend it to the original model network.

This results in a neural network with four hidden layers. A generalisation of an integrated encoder can be seen in Figure 3.6.

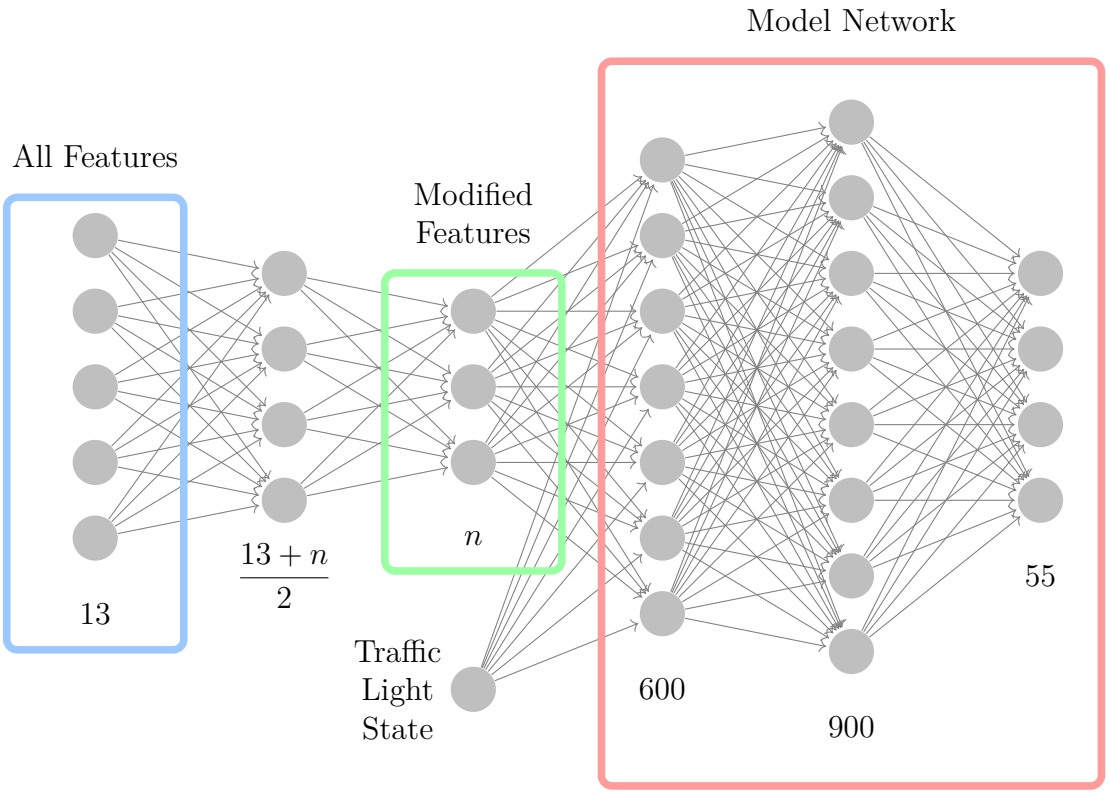


Figure 3.6: The integrated encoder model. The input layer, denoted as All Features with 13 neurons, contains all features from Table 3.1 except the traffic light state. The Modified Features layer is equivalent to the hidden state layer of an auto-encoder, where the number of neurons n is either equal to 9 or 5, making up 10 or 6 in total when the traffic light state is added. Between the input layer and the hidden state, an intermittent layer is placed with $(13 + n)/2$ neurons. The last two hidden layers and the output layer constitute the Model Network, presented in Section 3.4.

The idea of the integrated encoders is that the first two hidden layers should transform the given input features into a lower dimension. The second two, i.e. the hidden layers of the model network, then aim to find the correct action to perform in the same way as the original model network does. All weights in the network are updated simultaneously unlike the auto-encoder models, described in Section 3.5.3, where the encoder and the model network were trained separately. With this integrated dimensionality reduction method, the network should have the opportunity to find connections between what input features are directly relevant for vehicle control.

Two integrated encoders models were created. The auto-encoder layers of the networks, namely the input layer and the first two hidden layers contained the same number of neurons as the auto-encoders used for the auto-encoder models in Section 3.5.3. This means that the input layer always contained 13 neurons representing

all features available except the traffic light state. The first hidden layer had either eleven or nine neurons while the second hidden layer, containing the modified features, had nine or five ($n = 9$ or 5), which makes up ten or six in total when the traffic light state is added. All neurons, except the traffic light state, in the second hidden layer were fully connected to the first hidden layer. The last two hidden layers and the output layer were identical to the original model network as specified in Section 3.4.

The training of the entire network was identical to how the model neural network was trained, which was described in Section 3.4. The only difference is that dropout was not implemented for the first two hidden layers.

3.6 Performance Comparison of Models

When all models for each dimensionality reduction method had been trained, various quantities evaluating their characteristics were collected. During the training of the model network, the classification accuracy was measured. The accuracy was defined as the fraction of how many times the network chose the same action as the autopilot, i.e. how well it managed to imitate the correct behaviour. The fully trained models were also tested in the Carla simulator and other quantities related to the driving performance were measured. Each model was evaluated for 500 episodes, where all episodes were initialised in the same manner as when the training data set was gathered, which was explained in Section 3.3.

In every time step of the simulation, the absolute distance to the centre of the lane was recorded. For each episode, the average distance was saved and used for evaluating how well the models were at keeping the vehicles centred in the lane. Additionally, the maximum speed of every episode was also stored with the purpose of analysing whether the vehicles had a tendency to drive too slow or too fast.

Quantities regarding the models' abilities to handle traffic lights were also collected in the simulation. Every time a vehicle drove through an intersection, the state of the traffic light was stored to determine whether the vehicle ran a red light or passed lawfully at green. Furthermore, if the velocity of the vehicle was close to zero, more specifically less than 0.007 m/s, and the upcoming traffic light was red, the distance to the traffic light was measured. This situation was interpreted as if that the vehicle intended to stop and wait for the light to turn green before continuing down the road.

Lastly, information regarding the way the episodes ended were collected. The various ending categories that were tracked included whether the car reached the end destination or not. If not, the reason for its failure was recorded. This could be due to that the car froze, meaning that it stood still for too long. The episode was therefore ended if the car had not moved during a full time lap of one traffic light, implying that the opposite lights must have been green at some point and the car should have moved. A different alternative to an ending was if the car collided with an object. Additionally, if the heading angle of the car took an unreasonable value,

defined as more than 50 degrees off the direction of the road, the episode ended. The last alternative for an ending category was if the car ended up far outside the main road, defined as more than 15 metres away from the centre of lane.

4

Results

The main questions posed in this report are how different dimensionality reduction methods affect the model training and driving performance as well as how to determine what set of features are the most relevant when driving through signalised intersections. To answer these questions, a range of different models have been developed based on neural networks and imitation learning. All models include some variant of dimensionality reduction, namely choice based on reason (Reason), Principal Component Analysis (PCA), Auto-Encoder (AE) and Integrated Encoder (IE). An additional reference model was also created with no dimensionality reduction performed. The features considered in the analysis are presented in Table 3.1.

In summary, the results firstly include a presentation of the results directly connected to the training of the models. Secondly, a comparison between the different models is presented regarding their driving performance. Lastly, an analysis of the different features, their respective importance and set of weights is presented.

4.1 Training and Accuracy

In total, nine different models were trained: one reference model, two that based their dimensionality reductions on reason, two on PCA, two on auto-encoders and two on integrated encoders. All models were trained on the same data set until an early-stopping trigger was activated. The early-stopping trigger used is described in further detail, as well as the rest of the training parameters, in Section 3.4.

The latest updated models before the early-stopping was activated were used for further comparison. Their accuracies were computed as the fraction of how many times the network chose the same action as the autopilot. All accuracies for each respective model are presented in Table 4.1. The highest value was achieved by using all 14 features in the reference model and the lowest for the model using an integrated encoder with only six features. Generally, all accuracies range between 82% and 95% which implies that all networks manage to find the correct label over four out of five times.

As explained in Section 2.2.1, when dimensionality reduction is performed through PCA, the data is projected onto a set of eigenvectors and the proportion of the corresponding eigenvalues can be calculated according to Equation (2.3). This proportion represents a measure of how much of the total variance is explained in the

Table 4.1: The accuracies of the models for all four implemented methods: Reference, Reason, PCA, AE and IE using either fourteen, ten or six features. The same data set was used for all models. The accuracy was computed after the training was completed and defined as the fraction of how many times the network chose to perform the same action as the autopilot.

	Reference	Reason		PCA		AE		IE	
	14	10	6	10	6	10	6	10	6
Accuracy [%]	94.9	94.7	93.4	93.3	89.0	88.3	86.1	83.5	82.4

new projected data set, i.e. how much of the information that was kept in the projection. Using this equation, it was computed that the PCA model using ten features explained 90.2% of the total variance and the other using six explained 65.2%.

4.2 Driving Performance

To be able to assess the models' performances, they were evaluated for 500 episodes each in the Carla simulator under identical circumstances. Firstly, measurements regarding how an episode ended are presented. Secondly, the models' behaviour around traffic lights is displayed, including both if they stopped before traffic lights and at what distance. Thirdly, more general behaviour such as the speed and how close to the centre of the lane they drove are shown individually. Lastly, a summarised performance evaluation including all measurements is presented.

4.2.1 Episode Endings

The way each episode ended was recorded for all models and the final result can be seen in Figure 4.1. The lower green bars in the figure show the number of times the vehicle managed to drive from the start to end destination, as defined in Figure 3.1. The dashed red bars represent the number of times a car froze, i.e. stopped for too long, somewhere along the route. The upper third bars in the figure, coloured in blue, show how often the heading angle of the car took an extreme value. The remaining two alternatives an episode could end, i.e. by driving off the road or colliding with an object, never occurred and are therefore not displayed in the figure.

As can be seen in the figure, the auto-encoder model using ten features, AE 10, was best at completing the entire path out of all models. The auto-encoder using six features, AE 6, on the other hand, did not perform very well, where less than one third of the vehicles reached the end destination. The second best model was Reason 6. The reference model and Reason 10 also performed well where 87% and 77% of the vehicles in the episodes reached the end respectively.

Furthermore, the PCA models behaved differently compared to one another, where PCA 10 only managed to reach the end destination approximately half of the times whereas PCA 6 completed 93% of the episodes. Neither of the integrated encoder

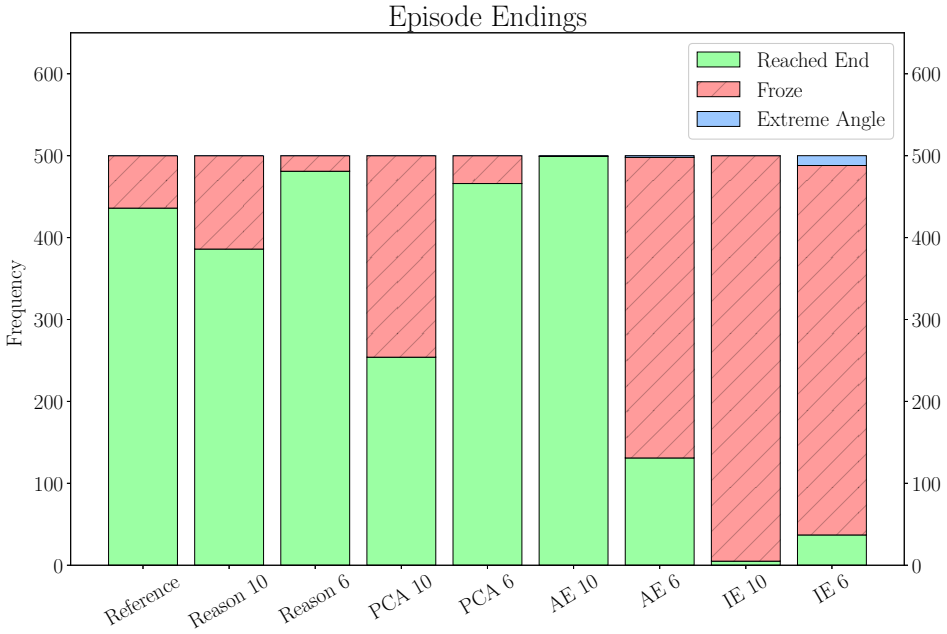


Figure 4.1: The way each episode ended. The lower green bars represent how often the car reached the end destination. The dashed red bars show how often a vehicle froze. The top blue bars, which are only clearly visible for IE 6 but also occurred in two episodes of AE 6, represent how often the heading angle of the car took an extreme value.

models completed more than 10% of the episodes. The integrated encoder and autoencoder models using six features were the only models that generated episodes which ended due to that the heading angle took an extreme value.

4.2.2 Traffic Light Behaviour

This section presents how well the models handled the traffic light scenarios. Important to note when analysing the figures is that the models did not drive through the same number of traffic lights. This was due to the fact that not all vehicles reached the end destination and therefore did not pass all three intersections in every episode, as can be concluded from Figure 4.1.

Figure 4.2 shows the proportions of how often a vehicle passed an intersection when the traffic light was green and red respectively. The lower green bars show how often it drove when the light was green and the upper dashed red bars show how often it ran a red light. As can be seen, Reason 6 drove through the intersections at green light most frequently, 95% of the times. Reason 10 and the reference model passed just below 80% of the intersections at green light. The remaining models performed worse and ran red lights more than 50% of the times where AE 10 performed the worst and ran red lights more than 80% of the time.

Additional data was measured regarding the traffic lights. Figure 4.3 shows how

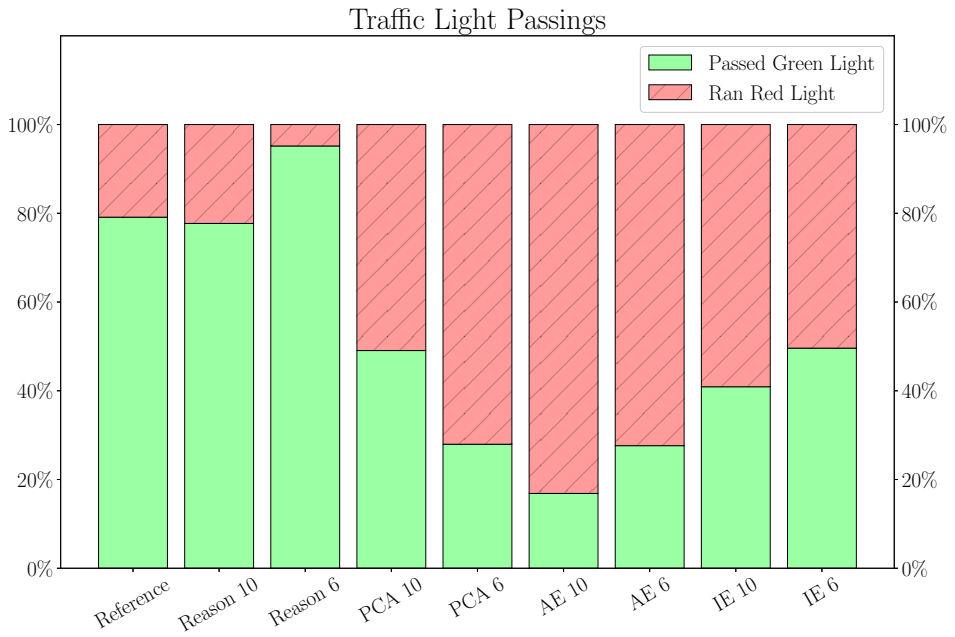


Figure 4.2: The proportion of how often the vehicles passed the intersections when the traffic light was either green or red. The lower green bars show how often it drove when the light was green and the upper dashed red bars show when it ran a red light.

far from an intersection the vehicles stopped when the traffic light was red. The boxes in the figure show the interval between the first and third quartile of the data, namely where 50% of the data lies. This interval is referred to as the interquartile range. The dashed red lines show the second quartile, i.e. the median. The whiskers start at both the upper and lower edges of the boxes and the lengths are set to be the minimum of the most extreme outlier and the interquartile range multiplied by 1.5. The diamond markers are the outliers, i.e. all data points that lie outside the whiskers' range. Important to note when studying the figure is that the vehicles did not necessarily stop at every intersection encountered. This would occur if the traffic light turned green as the vehicle was approaching, meaning that the vehicle did not have to stop, or if a vehicle ignored the light and drove through the intersection regardless of whether the traffic light was red or not.

It can be seen in Figure 4.3 that Reason 10 is the best out of all models at stopping close to the traffic lights in terms of the median distance, with a median of 0.99 metres. Both Reason 6 and the reference model stopped relatively close too, with median distances of 1.47 metres and 1.45 metres. However, all these three models have outliers grouped at approximately 42 and 75 metres. The collection of outliers at 42 metres represents the number of times a vehicle stopped immediately after initialisation at the starting position and the other collection at 75 metres illustrates when a vehicle stopped shortly after the stop mark of a traffic light, see Figure 3.1.

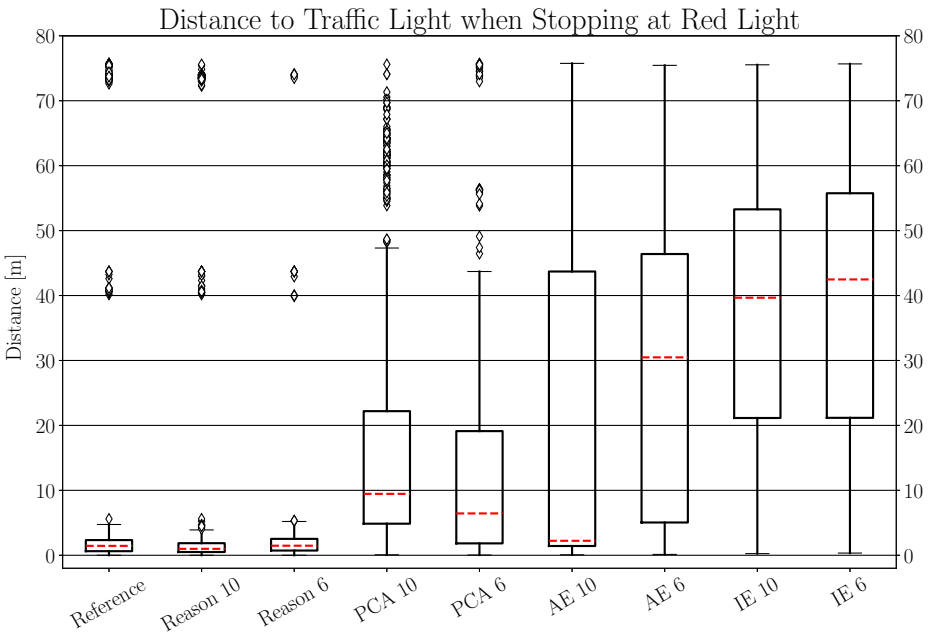


Figure 4.3: How far from an intersection the vehicles stopped when the traffic light was red. The red dashed lines show the median value. The boxes span the first to the third quartile and the whiskers start at the box edges with a length of 1.5 times the interquartile range. The diamond markers represent the outliers.

Furthermore, the PCA models performed worse than the reference and Reason models with a larger median distance and interquartile range. The larger interquartile range might imply that the decision making of where to stop was not as well-defined as for the reference and Reason models, whose stopping distances were relatively concentrated around their small median values. The same applies for the auto-encoder and integrated encoder model where the ranges are even larger.

The median distance to the traffic light for both the auto-encoder and integrated encoder models increased when fewer features were used. Though, it is important to note is that even though AE 10 has a small median distance to the traffic light, AE 10 had a tendency to run a lot of red lights, meaning that it did not stop at intersections very often.

When combining the result from the two figures, 4.2 and 4.3, it can be concluded that Reason 6 performed the best since it drove through intersections at green light most often and when it was red it stopped close to the intersection while having few outliers. It is harder to distinguish whether Reason 10 or the reference model performed the second best, since the reference model passed green lights more often, while Reason 10 stopped closer to the intersections. The rest of the models frequently ran red lights and stopped far away from the intersections and thus did not handle traffic lights well.

4.2.3 Additional Driving Behaviour

Apart from information about how the episodes ended and the performance regarding traffic lights, additional information was also collected concerning the speed and positioning of the vehicle. The average maximum speed for all models can be seen in Table 4.2. For each episode, the maximum speed was noted and the mean and standard deviation in the table was computed using all 500 samples. As can be seen in the table, all models had approximately the same maximum speed where PCA 10 drove the fastest at 21.4 km/h and IE 6 was the slowest at 18.3 km/h.

Table 4.2: The average maximum speed (Mean) and standard deviation (Std) for all models, measured in km/h. The maximum speed was noted in each episode and the mean and standard deviation were calculated using all 500 data points for each model.

	Reference	Reason		PCA		AE		IE	
Speed [km/h]	14	10	6	10	6	10	6	10	6
Mean	19.0	18.7	19.2	21.4	18.6	18.5	20.0	18.7	18.3
Std	6.3	6.3	6.1	6.6	6.2	6.2	5.9	6.4	6.5

Also, how close to the centre of the lane the vehicles drove on average can be seen in Figure 4.4, where the boxes are interpreted in the same manner as described in Section 4.2.2. The distance to the centre of the lane was measured at every time instance and the average was computed over the whole episode. Note that all episodes were of different lengths since the traffic light times were randomised and the episodes ended in different ways, for example by freezing somewhere along the route or reaching the end. The lanes are approximately 3.5 metres wide and as can be seen in the figure, most of the models stayed close to the centre, less than 10 cm, except the AE 6 and IE models. Both of the PCA models have outliers, but the median values are small.

4.2.4 Summary

Overall, four different aspects on driving through intersections were assessed: the ability to reach the assigned end destination, the understanderstanding of how to behave around traffic lights, correct lane positioning and suitable speed. As a general comment, the results show that all models managed to understand some elements of driving through intersections, though some are better than others.

To start with, Table 4.2 shows that all models managed to choose suitable speed and neither drove too slow nor too fast. Furthermore, most models kept the car in the middle of the lane, as can be seen in Figure 4.4. The reference model, both Reason models and AE 10 had close to perfect positioning in the lane and so did the two PCA models, apart from a set of outliers. In regard of the three remaining models: AE 6, IE 10 and IE 6 had more critical behaviours with no median distance smaller than 45 cm.

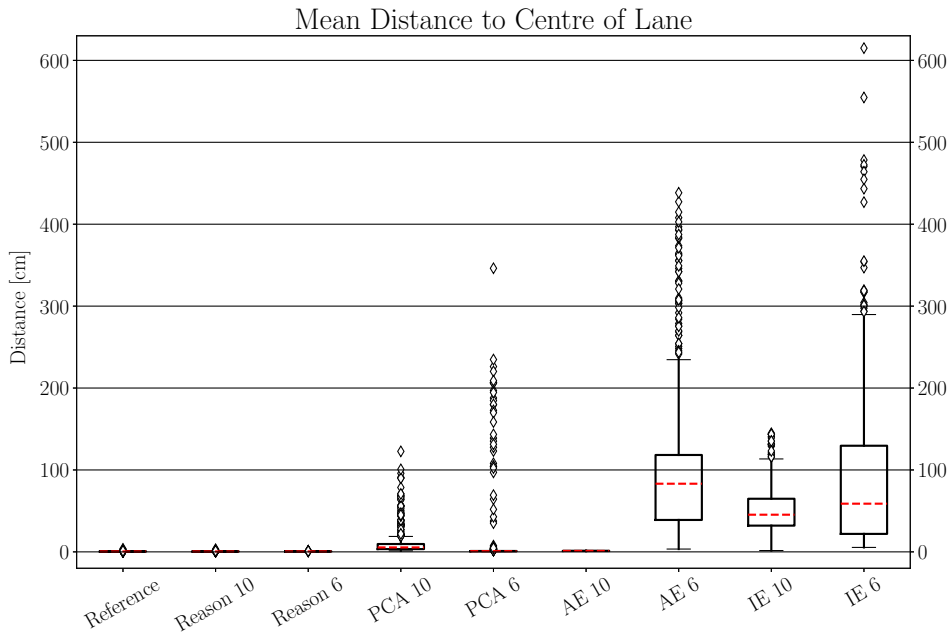


Figure 4.4: How close to the centre of the lane a car was on average per episode. The red dashed lines show the median value. The boxes span the first to the third quartile and the whiskers start at the box edges with a length of 1.5 times the interquartile range. The diamond markers represent the outliers.

What seems to distinguish the different models the most is how they behaved around traffic lights, as described in Section 4.2.2. This is where the reference and Reason models stood out, due to that they stopped at reasonable distances at red light and managed to pass the intersection at green most frequently. They also belong to the models that reached the end most often, even though PCA 6 and AE 10 were successful in this aspect as well, see Figure 4.1. However, as previously noted, the models PCA 6 and AE 10 did not perform well in regard of the traffic lights.

Additionally, all models were recreated in order to study their stability when re-trained and their results can be seen in Appendix A. At recreation, the models were evaluated for 300 episodes instead of 500. The results of the reference and Reason models were easy to reproduce and showed the same type of behaviour as previously described. The PCA models mostly showed the same driving behaviour as before, with the exception that PCA 10 reached the end destination more often than before and PCA 6 passed the intersections at green light more often.

The behaviour of the AE and IE models were more difficult to reproduce, on the other hand. Both the AE 6 and IE 10 model managed to reach the end destination more often than before, but kept the same trend of running red lights. All AE and IE models tended to stop closer to the intersections and drove closer to the centre of the lane, except the AE 10 model.

In summary, the reference and Reason models performed well, or even the best, in all categories. They were also stable in training and the behaviours were easy to

reproduce. Moreover, among these three models, Reason 6 is the one that performed the absolute best since it both reached the end most frequently and handled the highest proportion of traffic light passings correctly.

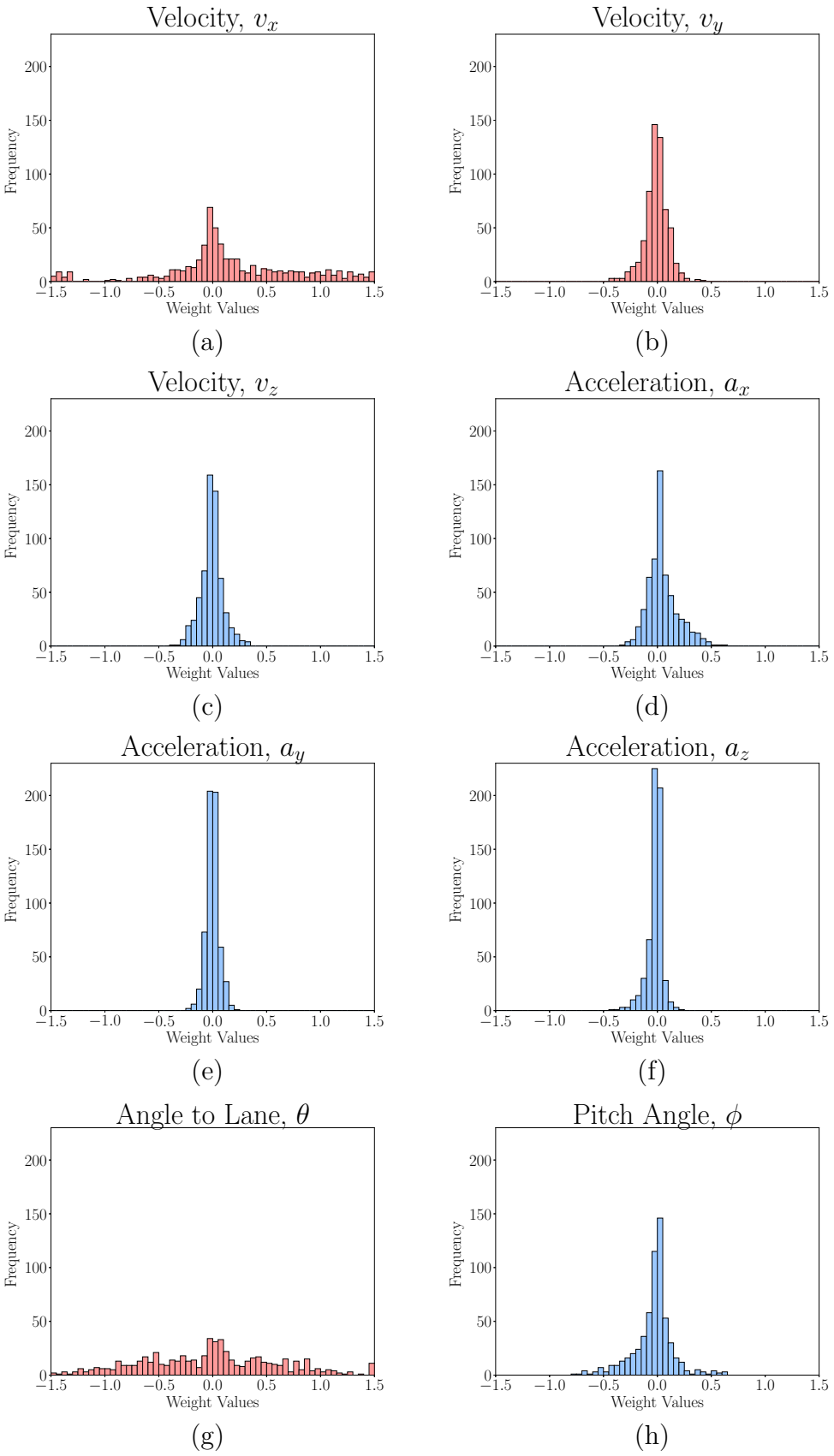
4.3 Feature and Weight Analysis

By studying the distribution of the weights for each feature's input neuron, their respective likelihood to affect the network's output can be analysed, as explained in Section 2.1.1. The distributions for all 14 features are presented in Figure 4.5. All graphs include histograms of the values of the 600 weights for each input neuron split into 0.05 units wide bins between the values -1.5 and 1.5, where the relatively few values larger in absolute value were truncated. The histograms in red correspond to the features chosen for the model Reason 6, see Table 3.2, namely subfigures a, b, g, l, m and n. The features in blue represent the remaining eight, i.e. subfigures c, d, e, f, h, i, j and k. In addition, the collective sum of the weights in absolute value for each distribution is presented in Table 4.3.

Table 4.3: The collective weight sums for all features of the reference model. The sum was taken over all the weights in absolute value corresponding to each feature's distribution in Figure 4.5.

Features		Collective Weight Sum
Velocity	v_x	268.4
...	v_y	46.8
...	v_z	43.5
Acceleration	a_x	63.0
...	a_y	25.2
...	a_z	27.3
Angle to lane	θ	302.0
Pitch angle	ϕ	73.1
Distance to centre of lane	d_x	99.4
...	d_y	93.7
...	d_z	138.5
Distance to next traffic light	t_x	190.8
...	t_y	84.0
State of next traffic light	s	256.2

When analysing the distributions in Figure 4.5 and Table 4.3, several things can be noted. Firstly, the distributions corresponding to the velocity (a, b and c) tend to be more spread than the ones related to the acceleration (d, e and f) when compared dimension-wise. The same can also be concluded from the collective weight sum, where the sum related to v_x is larger than a_x, v_y than a_y and v_z than a_z. The x-direction dominates notably, where the collective weight sum is 268 for the velocity and 63 for the acceleration compared to 46 and 25 for the y-direction respectively.


Velocity, v_x
Velocity, v_y
Velocity, v_z
Acceleration, a_x
Acceleration, a_y
Acceleration, a_z
Angle to Lane, θ
Pitch Angle, ϕ
Frequency
Weight Values
(a)
(b)
(c)
(d)
(e)
(f)
(g)
(h)

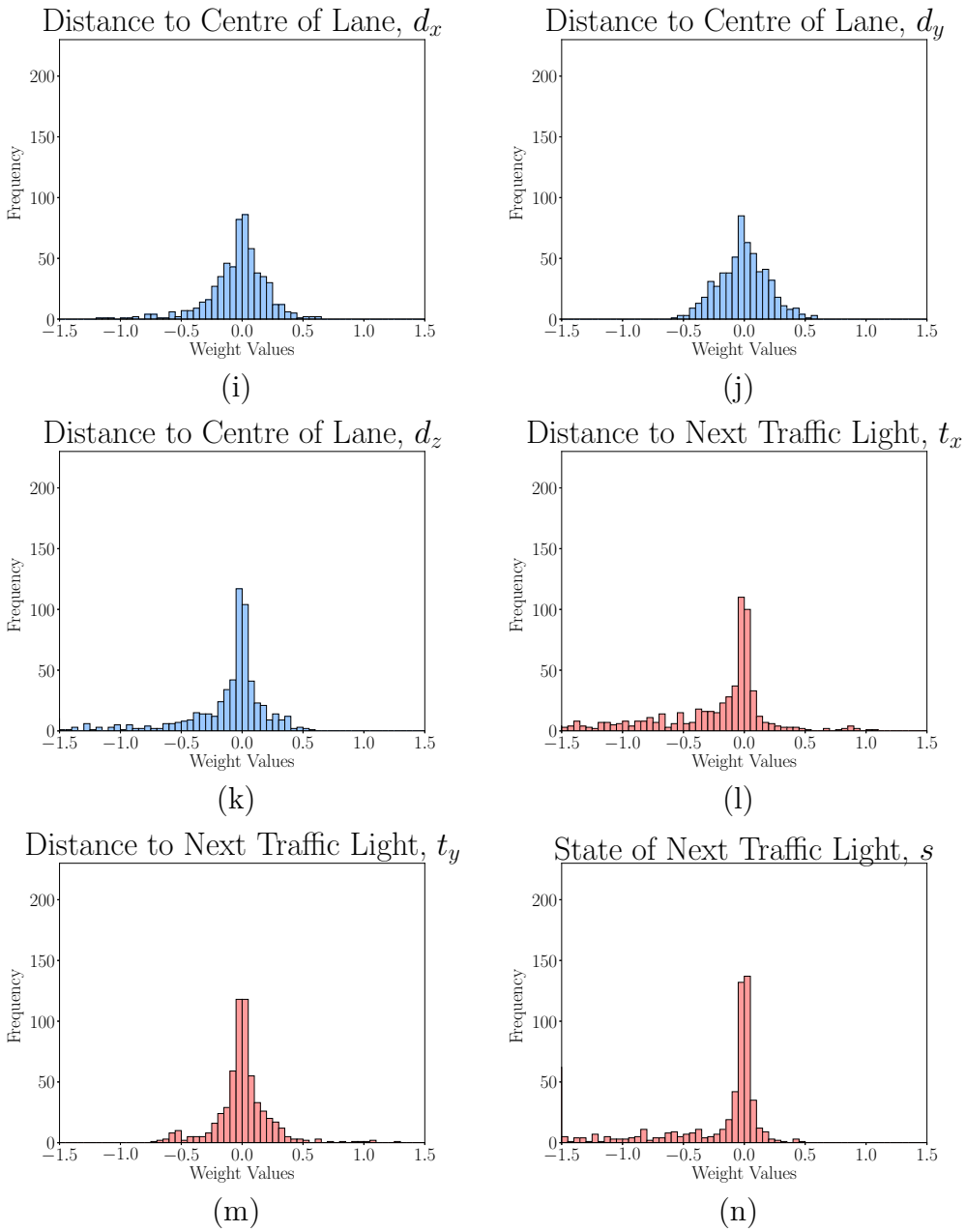


Figure 4.5: The weight distributions for all 14 features in the fully trained reference model. Each histogram contains the 600 weights for each feature, indicated by the respective title, that connect the input layer with the first hidden layer. Only weights that lie between -1.5 and 1.5 are shown in the graphs. The histograms coloured in red (a, b, g, l, m and n) represent the features chosen for the model Reason 6 and the ones coloured in blue represent the remaining ones (c, d, e, f, h, i, j and k).

Secondly, the distribution of the angle to lane (g), which is based on the yaw angle of the vehicle, shows more variability than the pitch angle (h). The collective weight sum suggests the same: 302 compared to 73. Additionally, the angle to lane is more spread than the three distance to centre of lane features (i, j and k), which are all four related to how the vehicle is positioned. Moreover, the distance to centre of lane distributions in each respective direction show relatively equal spread and collective weight sum.

Lastly, the remaining three distributions (l, m and n) are the only ones related to information about where the traffic light is positioned and its current state. What can be noted is that the distribution for the distance to the next traffic light in the x-direction is more spread than in the y-direction and the collective weight sum is 190 compared to 83. Also, the distribution regarding the state of the next traffic light has an asymmetric spread with large weights in the negative spectrum, resulting in a relatively high weight sum at 256.

In order to establish reproducibility of the results regarding feature and weight analysis, the reference model was recreated. The corresponding distributions and collective weight sums can be seen in Appendix A.3. In summary, the distributions and sums show the same overall characteristics which indicate that the results are stable and reproducible.

5

Discussion

Controlling a vehicle through signalised intersections is a challenging task. Managing to stay within the lane, as well as knowing when and where to stop for traffic lights, requires a model well-aware of the environment that makes appropriate decisions at all times. How the environment is represented is an important aspect when modelling how to drive. Two principal questions are addressed in this report regarding this subject: what features and how they should be represented in order for the model to learn how to control a car most efficiently.

The discussion is divided into four parts, where the first discusses the impact of different dimensionality reduction methods when applied before training a neural network. The second part further expands on the weight analysis as a tool for determining which features are the most relevant for the task at hand. The third section reports on the most important critical remarks and source of errors and the fourth elaborates on relevant future work.

5.1 Impact of Dimensionality Reduction

In general, all model types managed to learn some or most aspects of driving through signalised intersections. The difference in behaviour between the models and discussion on why they differed are presented in the upcoming sections, where each dimensionality reduction method and model type is discussed individually at first. Thereafter, a deepened discussion follows on the relation between the accuracy and driving performance as well as on general convergence.

5.1.1 Reason

The models that based their dimensionality reduction on reasoning, namely Reason 10 and Reason 6, are the models that performed best among all different model types. Between the two Reason models, Reason 6 is the one performing the utter best due to the slightly higher percentage of correct traffic light passings as well as a higher tendency to reach the end destination. What is interesting to note is that Reason 6 also performed better than the reference model with the same argument.

However, the reference model produced the highest accuracy which is in order with the theory on neural networks, where more information should generally increase the accuracy. Though, this suggests that accuracy is not a sufficient measure in order to evaluate how well a model performs. Supposedly, the reference model tries to minimise the loss and increase the accuracy, in which it succeeds, but does not manage to generalise as well as the model Reason 6 in terms of driving. This implies that some sort of dimensionality reduction or preprocessing of the features and data may be beneficial instead of simply feeding all available information to the network.

5.1.2 Principal Component Analysis

Generally, when Principal Component Analysis (PCA) was used for dimensionality reduction, the performance was worse both in terms of accuracy and in driving performance than the reference and the Reason models. Also, determining which model is best between PCA 6 and PCA 10 is difficult, since the models showed different behaviour when recreated. However, the reason why the PCA models had troubles controlling the car is probably not related to their lower accuracies primarily. PCA 10 had practically the same accuracy as Reason 6, see Table 4.1, but showed worse performance in all driving measures. Instead, the issues are most likely linked to the nature of PCA and how it projects the data.

The strengths of projection based on PCA is its ability to reduce correlation. When two or more features are linearly correlated, a PCA projection eliminates the correlation and leaves only features linearly independent of one another. This can be very useful if the underlying problem with the data set is correlation. On the other hand when the data is not primarily correlated, but instead includes features which are irrelevant for the task, PCA does not seem to be a suitable method to use for dimensionality reduction.

This is the case in this report where for example several quantities are measured in both x-, y- and z-direction. Since the car only moves horizontally, the speed in z-direction should be irrelevant. PCA, however, does not take this into consideration and only focuses on keeping the statistical variance and features orthogonal to each other. Note that if the car had been driving on slopes on the contrary, the situation would have been different and the z-direction might have been relevant. In conclusion, this implies that PCA does indeed reduce correlation in the data set, but does not further aid the network in finding which features are relevant for driving through intersections.

5.1.3 Auto-Encoder

Concerning the Auto-Encoder (AE) models, they showed more or less the same difficulties as the PCA models, especially in regard of handling traffic lights. The reason for the AE models showing a similar behaviour is most likely due to their similar construction. Both methods try to find a representation of the data with

fewer dimensions, but still keep as much of the information as possible. The major difference between the two is what tools they use to do it. PCA projection is built entirely on linear mapping using matrix multiplication. Alternatively, the AE projection constructed through a neural network may use non-linear mappings in any way suitable.

However, as explained in the discussion regarding PCA in Section 5.1.2, PCA models try to reduce correlation and the same applies to AE models. The dimensionality reduction functionality does not concern itself with the fact that the data is used for controlling a car through signalised intersections, but only focuses on keeping as much of the information content as possible, even if some information is useless for the task. Therefore, since PCA did not show promising results in this application, the results of the AE models follow the same trend. Reducing correlation, even if done with non-linear mappings, when irrelevance is still a significant problem, does not improve the overall performance.

5.1.4 Integrated Encoder

The Integrated Encoder (IE) models proved to be an interesting case. The models tended to freeze almost every run, they rarely stopped at appropriate distances at traffic lights and the accuracies were the lowest for all measured models. However, they should be best in theory. The reason is because they have what the other methods lack: a direct connection between the dimensionality reduction functionality and the control of the car, since the two parts are trained jointly.

As explained in Section 5.1.2 and 5.1.3, the PCA and AE models try to minimise correlation and do not focus on relevance primarily. Contrarily, the IE models have the opportunity of understanding what input features are relevant for driving by forcing the network to learn this connection. The IE models did not show good results generally though and were outperformed by the Reason models. The explanation for why the IE models did not work in practice is probably due to convergence problems. For example, when the model IE 6 is studied, the main results show an accuracy of 82.4% and when retrained, the accuracy drops to 80.0%. This implies that the network has difficulties of finding the global optimum and tends to get stuck in local minima instead.

5.1.5 Further Discussion

As concluded, the Reason models generally outperformed the other models and Reason 6 specifically proved to be the best model in terms of driving. This shows that accuracy is not a good enough measure for driving performance for the models, since the reference model drove worse, but produced a higher accuracy than Reason 6.

There are several reasons why high accuracy might not imply good driving behaviour. Firstly, the data collected using the autopilot proved to be imperfect. The autopilot controlled the car in a stable manner, but ran red lights occasionally and

sometimes confused itself of where to go, which resulted in a spinning-like behaviour. This means that some portion of the data may have been mislabelled due to the incapability of the autopilot.

Secondly, determining how high accuracy is enough in order to drive well is also complicated. As explained, all models have accuracies above 80%, which could be interpreted as if the models take the correct decision in at least four out of five times. But the models showed high variability in driving performance, so the last few percentage units seem to make the whole difference in regard of comprehending the traffic situation.

Lastly, the accuracy values do not directly define which scenarios are covered by the model and which are not, for obvious reasons. For example, if one model shows an accuracy of 85% and another at 88%, then this does not necessarily imply that the second model knows everything the first model does and more. On the contrary, it might have focused on a different set of data points. When the models are tested in practice, it is possible that the first model has learnt more relevant features for driving even if the accuracy is lower.

Another conclusion from the preceding discussion is that the IE models did not work as expected, which is most likely due to instability problems. Instability at reproduction also concerned the PCA and AE models, where PCA 10 dropped 1.1 percentage units and AE 10 increased 2.0 percentage units at recreation, see Appendix A.1. This strongly indicates that none of these models generally converge to the global optimum when trained with the proposed method. Different training settings may be tuned in order to aid the optimiser in finding the global optimum and, in regard of the driving performance measures, longer evaluation may be considered.

5.2 Analysis of Feature Selection

As explained earlier, the model that was concluded to perform the best was Reason 6. From the weight distribution analysis performed in Section 4.3, a relationship between the features used by Reason 6 and the weights in the first layer of the reference model can be seen. Simply put, the features of Reason 6 had some of the most spread distributions and the largest Collective Weight Sums (CWS). This suggests that the weight distributions of a network trained with all available features may provide information about which features are relevant and which are not. Using the collective weight sum may therefore prove to be a useful statistical measure when determining feature relevance.

Listed below are all 14 features sorted in descending order according to their collective weight sums depicted in Table 4.3. The features in bold are the six features used in Reason 6.

1. **Angle to lane,** θ
2. **Velocity,** v_x
3. **State of next traffic light,** s
4. **Distance to next traffic light,** t_x
5. Distance to centre of lane, d_z
6. Distance to centre of lane, d_x
7. Distance to centre of lane, d_y
8. **Distance to next traffic light,** t_y
9. Pitch angle, ϕ
10. Acceleration, a_x
11. **Velocity,** v_y
12. Velocity, v_z
13. Acceleration, a_z
14. Acceleration, a_y

Important to note when studying the collective weight sums in detail is that Reason 6 did not use all of the six features with the largest sums, but differed by two. The sums suggest that the distance to the centre of lane in x- and z-direction should be used, but were instead replaced by the velocity and the distance to the traffic light in y-direction. Those two features are according to the collective weight sums the eleventh and eighth feature that should have been picked.

The most probable cause why the collective weight sums did not suggest the velocity and distance to the traffic light in y-direction is due to the definition of the coordinate system in Carla. The selected road is nearly parallel to the x-axis, resulting in very small variations in the features measured along the y-axis, which is probably why they are not suggested.

However, understanding why the network seemed to prioritise the distance to the centre of lane in z-direction is difficult. Since the road is entirely flat and does not include any elevation, the car should maintain the same height above the road and this feature should therefore be completely irrelevant. The collective weight sum of this feature was nonetheless the fifth highest. A possible explanation may be because of the preprocessing and standardisation of the data. When standardisation is performed on a practically constant feature, extremely small variations in the data may be magnified to an unreasonable extent. It can therefore be concluded that only comparing the weight distributions and blindly use the collective weight sum as indicator are not enough to get a full picture of what set of features to select.

In order to improve the selection method, it could be favourable to combine the weight analysis with some type of reasoning as well. For example, if there is no movement along an axis, then all components in that direction should be removed. Additionally, it could be a good idea to cluster features into groups such as those regarding the position in the lane or velocity related features. By grouping the features together, it may be determined if some groups are irrelevant and can be ignored or if only a maximum number of features is required from a specific group. As an example, this is what was done for the model Reason 6, where the angle to lane was prioritised over the three distance to centre of lane features, which are all four related to the lane positioning.

As a final remark, one issue with the weight distribution analysis is that only weights in the first layer, that are directly connected to the input features, are studied. An exhaustive weight analysis would consider all weights in all layers, but this is not

plausible with large networks due to the large number of combinations and immense complexity. However, as the results show, the first layer may still give an indication about feature relevance and when combined with reasoning, it may result in a set of promising features.

5.3 Critical Remarks

Several restrictions were made to simplify the task, as mentioned in Section 1.3, which might have affected the final results. One restriction was that the same model network structure was used for all models, even though different types of inputs might have benefit using other structures. The same reasoning goes for the decision to use the same data set when training all models, namely that other uniquely adapted data sets might enable better individual performance.

Other aspects of the training and choice of parameters which might have influenced the performance include the construction of the autopilot in Carla. Sometimes the autopilot shows abnormal behaviour such as running red lights or navigates incorrectly and this type of behaviour is recorded and used for training the models. Most often however, the autopilot acts more deterministic in terms of not drifting too far from the centre of the lane or driving faster than 20 km/h. This results in low diversity in the training data and therefore also in the models' behaviour. In order to increase the variety of the data, noise was induced where the rate of the noise was tuned manually after visual inspection in Carla. This improved the models' performances since a larger state space was covered.

Additionally, many of the parameters used in the model network were tested and evaluated using the Reason 10 model. The reason why is because this model did not require any further preprocessing and the initial idea was that this model included enough information to control the car. For example, some specifically tuned parameters are the optimiser, the dropout rate and activation function. The same goes for the number of neurons in the hidden layers which were tested in the simulator and then fixated and used by all models. Though, important to note is that Reason 10 did not turn out to be the best performing model, so the parameters were not excessively adapted.

Lastly, the model network used 55 discretised actions instead of continuous control signals which might otherwise seem like an obvious choice. The reason for this was to be able to evaluate and compare the models using the classification accuracy. When predicting continuous variables, the evaluation would instead be done by comparing the root mean square error of the loss, which is less intuitive to interpret.

5.4 Future Work

As concluded, the best performing model was based only on reasoning without any further dimensionality reduction. In order to continue the work, it would be

interesting to evaluate how well the weight distributions solely disclose what features are the most relevant, as a comparison to the model Reason 6. This implies creating a model using the six features with the largest Collective Weight Sums (CWS) and evaluate this model in Carla with the metrics described in Section 3.6.

Furthermore, as already discussed in Section 5.2, an optimal approach is probably to combine the weight analysis with reasoning in order to get an idea of what features should be used. It would therefore be interesting to evaluate if this method would work in other environments as well and not only the studied signalised intersections in the simulator. For example, the environment could be extended to include roundabouts, highways or sharp turns. Additionally, it would be interesting to test the other dimensionality reduction methods, namely the PCA, auto-encoders and integrated encoders, to see if the same results are achieved in a different environment.

It would also be of interest to investigate the use of a more iterative approach when selecting the set of features based on the weight distribution analysis. The proposal is to train a neural network using all available features and when it has been fully trained, the feature with the lowest collective weight sum is removed and a new network is trained with the remaining features. The procedure continues until the requested number of features have been selected or the model performance does not improve further. This algorithm might not guarantee convergence to a global optimum, but might increase generalisation.

Moreover, the training, validation and evaluation are currently all performed in the same environment. It would be valuable to recreate the models in another set up than the sequential intersections as seen in Figure 3.1. The new set up would still include intersections and the same type of features, but the size of the lane, the distance between the intersections and initialised velocities could vary for example. This would give a better measure of how good the models are at generalisation. Also, the input signals are at present noise-free so it would be interesting to add noise to make the signals more realistic. These noise signals could for example be simulated or generated by gathering data from the real world.

Lastly, the feature selection and dimensionality reductions are only investigated from an autonomous drive perspective. It would be interesting to see if the proposed feature selection algorithm is applicable in other areas which involve high dimensional classification problems. This could for example be applied to the classification of mammography images, which was described in Section 1.4, or microarray analysis, which is used in cancer research and many other areas [29]. Other related problems also include predictive maintenance of complex machines or engines, where the number of possible input features is high and determining which are relevant may be a challenge.

6

Conclusion

Primarily two questions have been investigated in relation to feature representation in autonomous drive. Firstly, how do different dimensionality reduction methods affect the model training and driving performance and secondly, how to determine the most relevant set of features for the specific task at hand. Among the four dimensionality reduction methods, namely feature choice based on reasoning (Reason), Principal Component Analysis (PCA), Auto-Encoders (AE) and Integrated Encoders (IE), the variant that performed best is Reason. Specifically, the Reason model that used only six features proved to be most successful in terms of driving.

In regard of the two dimensionality reduction methods PCA and AE; these models did not improve the model performance, but instead worsened it compared to the reference model using all available features. This is most likely due to how they reduce the dimension in general. Both focus on lowering the dimension by reducing correlation in the data, but does not take the features' relevance for the specific task into consideration. When irrelevance in data is prominent, due to inclusion of features in vertical direction when only driving horizontally for example, these methods lack in efficiency.

Furthermore, the IE models specifically, but also the PCA and AE models generally, were proven difficult to reproduce. This is due to convergence problem, which can be noted from the fact that the accuracy of the fully trained models varied notably. This also serves as an explanation why the IE models did not perform as expected, whom where predicted to perform best in theory due to their model construction.

Therefore, dimensionality reduction through reasoning is the method that was found most successful. This also proves the importance of detailed assessment of the data and task beforehand and that dimensionality reduction may improve performance. Furthermore, neural networks aim to minimise the loss and increase the accuracy as defined by their nature. But, as was shown, the accuracy does not disclose entirely how the car drives ultimately, since the model driving the best was not the one with the highest accuracy.

In order to determine what features should be used when training a model, the features represented in the best performing model was studied in relation to the reference model using all available features. It was shown that the weight distributions of the weights in the first layer of the reference model may indicate which features are relevant for the task and which are not. A well-spread corresponding weight dis-

tribution and high Collective Weight Sum (CWS) may indicate if a feature should be included in the model or not. As proposed, the best performance is most likely achieved when the set of features is chosen through both weight distribution analysis and reasoning based on the specific characteristics of the task and environment. Regarding future work, it would be interesting to investigate this idea further in both tasks related to autonomous drive but also in other high-dimensional problems, for example classification tasks in medical science or predictive maintenance of complex machines.

In conclusion, it is important to assess what features are to be used and in what representation to ensure that irrelevant information is ignored and relevant is accounted for appropriately. Neural networks are indeed famous for their generalisation capabilities, but have proven to benefit further by suitable preprocessing and reasonable choice of features.

Bibliography

[1] Christina Mercer and Tom Macaulay. "Which companies are making driverless cars?" In: Techworld, Jan. 2019. URL: https://www.techworld.com/picture-gallery/data/-companies-working-on-driverless-cars-3641537/.

[2] On-Road Automated Driving (ORAD) committee. "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles J3016_201806". In: SAE International, 2018. DOI: https://doi.org/10.4271/J3016_201806. URL: https://www.sae.org/standards/content/j3016_201806/.

[3] Aarian Marshall. "Elon Musk Promises a Really Truly Self-Driving Tesla in 2020". In: Wired, Feb. 2019. URL: https://www.wired.com/story/elon-musk-tesla-full-self-driving-2019-2020-promise/.

[4] "Uber car 'had six seconds to respond' in fatal crash". In: BBC, May 2018. URL: https://www.bbc.com/news/technology-44243118.

[5] "Tesla car that crashed and killed driver was running on Autopilot, firm says". In: The Guardian, Mar. 2018. URL: https://www.theguardian.com/technology/2018/mar/31/tesla-car-crash-autopilot-mountain-view.

[6] Dean A. Pomerleau. "Advances in Neural Information Processing Systems 1". In: ed. by David S. Touretzky. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989. Chap. ALVINN: An Autonomous Land Vehicle in a Neural Network, pp. 305–313. ISBN: 1-558-60015-9. URL: http://dl.acm.org/citation.cfm?id=89851.89891.

[7] Volvo Cars. *Volvo Cars Concepts - 360c*. 2019. URL: https://www.volvocars.com/intl/cars/concepts/360c.

[8] Waymo. *Waymo*. 2018. URL: https://waymo.com/.

[9] Volvo Trucks. *Vera - The future of autonomous transports*. Oct. 2019. URL: https://www.volvotrucks.com/en-en/about-us/automation/vera.html.

[10] NVIDIA Danny Shapiro. "Around the Valley in 80 Kilometers: NVIDIA Autonomous Test Vehicle Completes Fully Driverless Highway Loop". In: Oct. 2018. URL: https://blogs.nvidia.com/blog/2018/10/10/self-driving-highway-loop/.

[11] National Highway Traffic Safety Administration. "USDOT Releases 2016 Fatal Traffic Crash Data". In: NHTSA, Oct. 2017. URL: https://www.nhtsa.gov/press-releases/usdot-releases-2016-fatal-traffic-crash-data.

[12] Stanford Eric Roberts. “Neural Networks: History: The 1940’s to the 1970’s”. In: 2000. URL: https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/History/history1.html.

[13] C. O. S. Sorzano, J. Vargas, and A. Pascual Montano. “A survey of dimensionality reduction techniques”. In: *arXiv e-prints*, arXiv:1403.2877 (Mar. 2014), arXiv:1403.2877. arXiv: 1403.2877 [stat.ML].

[14] Federal Highway Administration. “Intersection Safety”. In: FHWA, July 2018. URL: https://highways.dot.gov/research-programs/safety/intersection-safety.

[15] CARLA Team. “CARLA Simulator”. In: CARLA, 2019. URL: http://carla.org/.

[16] Mariusz Bojarski et al. “End to End Learning for Self-Driving Cars”. In: *CoRR* abs/1604.07316 (2016). arXiv: 1604.07316. URL: http://arxiv.org/abs/1604.07316.

[17] Felipe Codevilla et al. “End-to-end Driving via Conditional Imitation Learning”. In: *CoRR* abs/1710.02410 (2017). arXiv: 1710.02410. URL: http://arxiv.org/abs/1710.02410.

[18] Michael Kelly et al. “HG-DAgger: Interactive Imitation Learning with Human Experts”. In: *CoRR* abs/1810.02890 (2018). arXiv: 1810.02890. URL: http://arxiv.org/abs/1810.02890.

[19] William Curran et al. “Using PCA to Efficiently Represent State Spaces”. In: *CoRR* abs/1505.00322 (2015). arXiv: 1505.00322. URL: http://arxiv.org/abs/1505.00322.

[20] Hongyu Shen et al. “Denoising Gravitational Waves with Enhanced Deep Recurrent Denoising Auto-Encoders”. In: (2019). arXiv: 1903.03105 [astro-ph.CO].

[21] Yasi Wang, Hongxun Yao, and Sicheng Zhao. “Auto-encoder based dimensionality reduction”. In: *Neurocomputing* 184 (2016). RoLoD: Robust Local Descriptors for Computer Vision 2014, pp. 232–242. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2015.08.104. URL: http://www.sciencedirect.com/science/article/pii/S0925231215017671.

[22] Saeid Asgari Taghanaki et al. “Pareto-optimal multi-objective dimensionality reduction deep auto-encoder for mammography classification”. In: *Computer Methods and Programs in Biomedicine* 145 (2017), pp. 85–93. ISSN: 0169-2607. DOI: https://doi.org/10.1016/j.cmpb.2017.04.012. URL: http://www.sciencedirect.com/science/article/pii/S0169260716309269.

[23] B. Mehlig. “Artificial Neural Networks”. In: *CoRR* abs/1901.05639 (2019). arXiv: 1901.05639. URL: http://arxiv.org/abs/1901.05639.

[24] Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. “Generalization in Deep Learning”. In: *arXiv e-prints*, arXiv:1710.05468 (Oct. 2017), arXiv:1710.05468. arXiv: 1710.05468 [stat.ML].

[25] Diederik P. Kingma and Jimmy Lei Ba. “Adam : A method for stochastic optimization”. In: (2014). arXiv: 1412.6980v9.

[26] Guangquan Lu et al. “Multi-task learning using variational auto-encoder for sentiment classification”. In: *Pattern Recognition Letters* (2018). ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2018.06.027. URL: http://www.sciencedirect.com/science/article/pii/S0167865518302769.

[27] Junfeng Zhang and Haifeng Hu. “Exemplar-based Cascaded Stacked Auto-Encoder Networks for robust face alignment”. In: *Computer Vision and Image Understanding* 171 (2018), pp. 95–103. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2018.05.002. URL: http://www.sciencedirect.com/science/article/pii/S1077314218300687.

[28] CARLA Team. “CARLA Simulator”. In: Screenshot by author. CARLA, 2019. URL: http://carla.org/.

[29] Rajeshwar. Govindarajan et al. “Microarray and its applications”. In: *Journal of Pharmacy And Bioallied Sciences* 4.6 (2012), pp. 310–312. DOI: 10.4103/0975-7406.100283. URL: http://www.jpbsonline.org/article.asp?issn=0975-7406;year=2012;volume=4;issue=6;spage=310;epage=312;aulast=Govindarajan;t=6.

A

Reproducibility of Results

In the following sections, the results of all the recreated models are presented and compared to the results displayed in Chapter 4, which will hereby be referred to as the original models. The recreated models were created under the same conditions as the original models and the metrics were measured the same way, with the exception that the recreated models were evaluated for 300 episodes instead of 500. Firstly, the training accuracy is presented. Secondly, the metrics related to the models' driving performance are given. Lastly, the corresponding feature and weight analysis is presented.

A.1 Training and Accuracy

Table A.1 shows the accuracy of the recreated networks. As can be seen in the table, the accuracy did not differ more than 2.4% for any models compared to what was achieved for the original models as presented in Table 4.1. The Reason models, PCA 6 and AE 6 differed the least by 0.1% and the reference and IE 10 also displayed small differences of 0.2% and 0.7% respectively. The remaining three models: PCA 10, AE 10 and IE 6, differed the most with 1.1%, 2.0% and 2.4%.

Table A.1: The accuracies of the recreated models and the corresponding absolute differences to what was presented in Table 4.1 for the original models; Reason, PCA, AE and IE using either fourteen, ten or six features. The accuracy was computed after the training was completed and defined as the fraction of how many times the network chose to perform the same action as the autopilot.

	Reference	Reason		PCA		AE		IE	
	14	10	6	10	6	10	6	10	6
Accuracy [%]	94.7	94.8	93.5	92.2	88.9	90.3	86.0	84.3	80.0
Abs. Diff.	-0.2	+0.1	+0.1	-1.1	-0.1	+2.0	-0.1	+0.7	-2.4

A.2 Driving Performance

Figure A.1 shows how the episodes ended for the recreated models. As can be seen when comparing to the results in Figure 4.1, the reference, Reason and AE 10 models

all behaved similarly as the original models, where approximately the same fraction of the vehicles reached the end destination. The remaining models, on the other hand, demonstrated a different behaviour and managed to complete the route more often. The largest increase was made by IE 10 which improved its ability to reach the end by 77%.

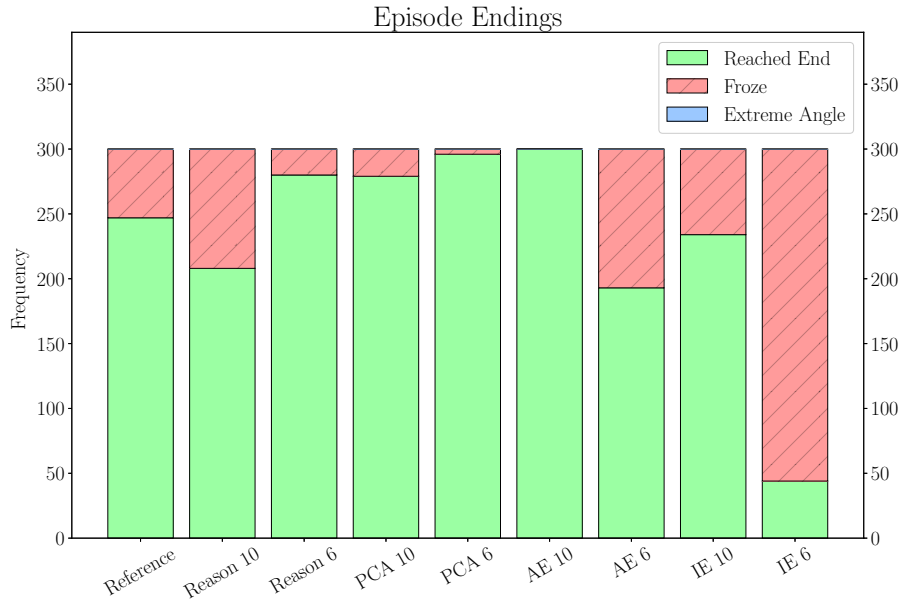


Figure A.1: The way each episode ended for the reproduced models. The lower green bars represent how often the car reached the end destination and the dashed red bars show how often a vehicle froze. Note that the reproduced models never experienced any extreme angles, so no blue bar is visible in this plot.

Figures A.2 and A.3 show the reproduced models' abilities to handle the traffic light scenarios. The first figure, Figure A.2, displays how often the vehicles passed the traffic light at green or red light. When comparing the passings to Figure 4.2, it can be seen that several models showed a similar behaviour, more specifically the Reason models, AE 10 and IE 10. Out of the remaining models: the reference, PCA 10, AE 6 ran more red lights than previously while the last two, i.e. PCA 6 and IE 6, displayed a better behaviour and passed at green more often.

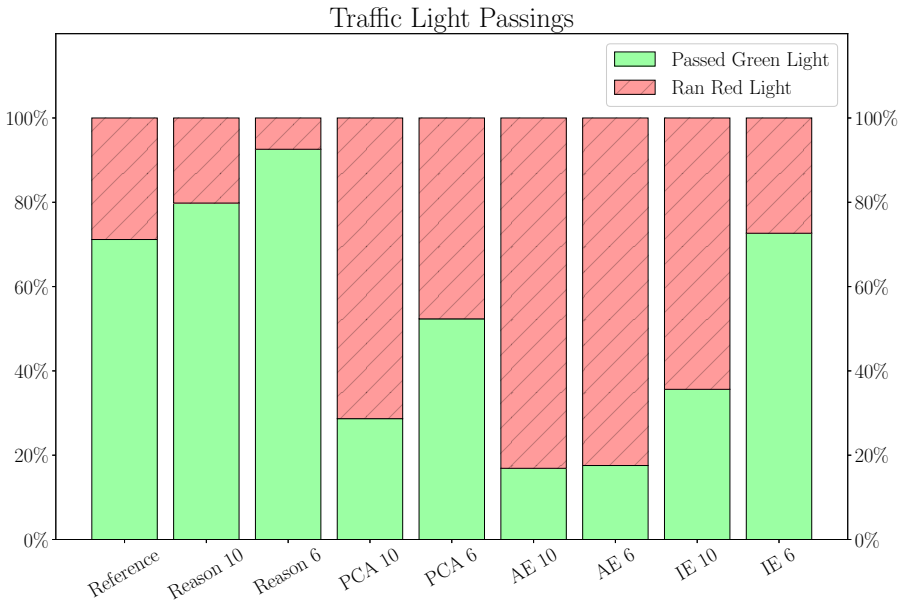


Figure A.2: The proportion of how often the vehicles passed the intersections when the traffic light was either green or red for the reproduced models. The lower green bars show how often it drove when the light was green and the upper dashed red bars show when it ran a red light.

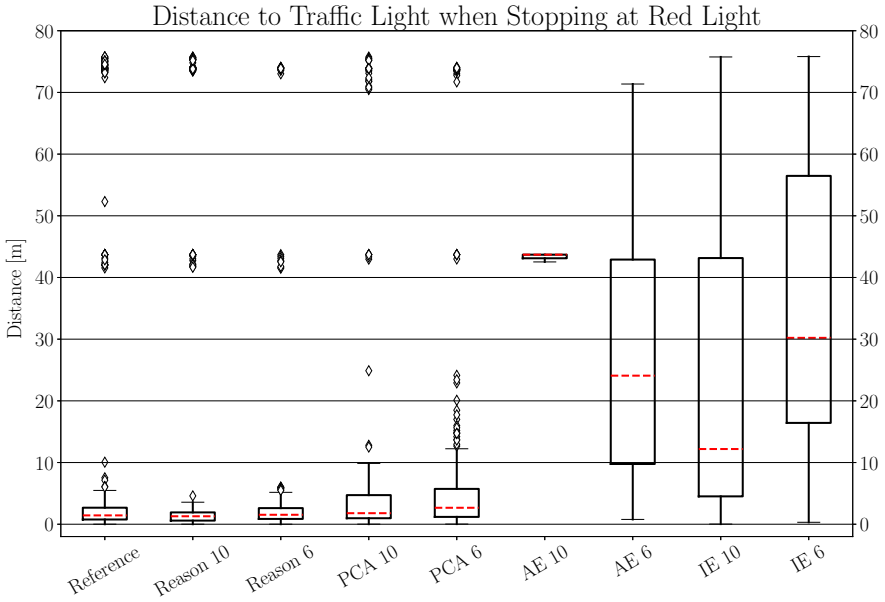


Figure A.3: How far from an intersection the vehicles of the reproduced models stopped when the traffic light was red. The red dashed lines show the median value. The boxes span the first to the third quartile and the whiskers start at the box edges with a length of 1.5 times the interquartile range. The diamond markers represent the outliers.

Figure A.3 shows how far from the intersection the vehicles stopped when the lights were red. The reference and Reason models maintained their ability to stop close to the intersection while the PCA models and AE 6 improved their competence slightly with fewer outliers and a lower median distance. The integrated encoders also improved their performance and stopped more than 10 metres closer to the intersections, though the median distance remained relatively big, at 12 and 30 metres respectively. AE 10 was the only model that performed worse than the original models and stopped more than 40 metres away from the intersections.

Table A.2 depicts the average maximum speed, standard deviation and their respective differences for all reproduced models compared to what was displayed in Table 4.2. As can be seen in the table, the maximum speed did not change significantly, where PCA 10 showed the largest difference and drove 2.5 km/h slower than the original models.

Table A.2: The average maximum speed (Mean), standard deviation (Std) and their respective differences (Diff. Mean and Diff. Std) for all reproduced models compared to the original models, measured in km/h. The maximum speed was noted in each episode and the mean and standard deviation were calculated using all 300 data points for each model.

	Reference	Reason		PCA		AE		IE	
Speed [km/h]	14	10	6	10	6	10	6	10	6
Mean	18.8	18.5	18.8	18.9	19.4	19.0	19.1	18.9	18.2
Diff. Mean	-1.2	-0.2	-0.4	-2.5	+0.8	+0.5	-0.9	+0.2	-0.1
Std	6.0	6.2	6.2	6.4	6.3	6.3	6.5	6.0	6.5
Diff. Std	-0.3	-0.1	+0.1	-0.2	+0.1	+0.1	+0.6	-0.4	0.0

Figure A.4 shows how far from the centre of the lane the reproduced vehicles drove on average. Most models showed a similar behaviour to the original models and managed to keep close to the centre of the lane, namely the reference, Reason, PCA and AE 10 models. Contrarily, both the AE 6 and IE 10 models showed an improvement compared to the original models, whereas IE 6 increased its interquartile range, meaning that it tended to drive further away from the centre.

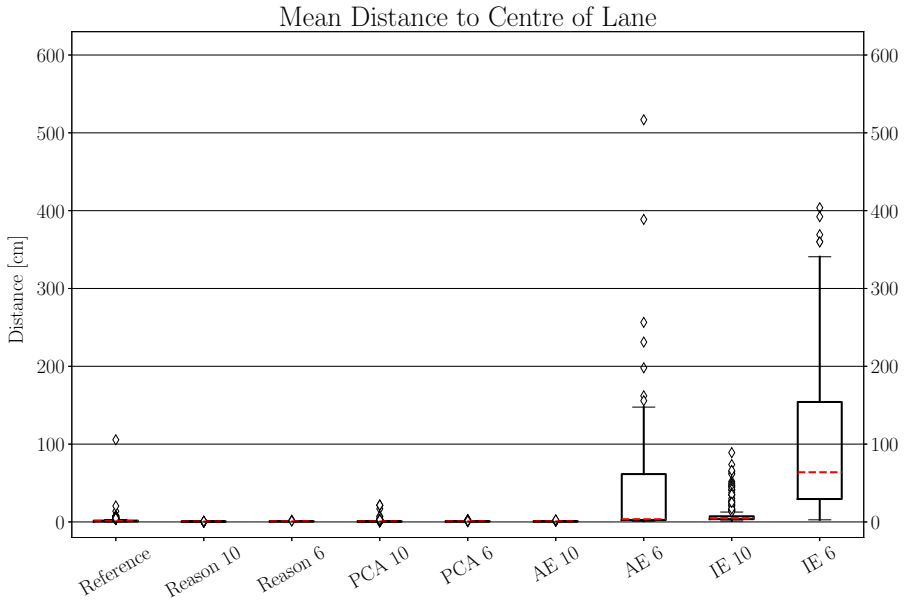


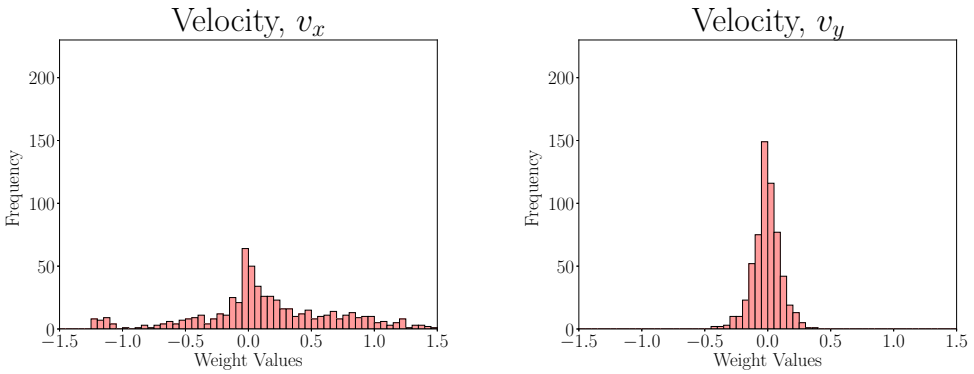
Figure A.4: How close to the centre of the lane a car was on average per episode for the reproduced models. The red dashed lines show the median value. The boxes span the first to the third quartile and the whiskers start at the box edges with a length of 1.5 times the interquartile range. The diamond markers represent the outliers.

A.2.1 Summary

From all the results presented above in Section A.2, it can be concluded that the reference and Reason models' behaviour were reproducible for all metrics. The PCA, AE and IE models, on the other hand were not as stable even though the networks achieved approximately the same accuracy with the same data set, as seen in Table 4.1. The unstable models were able to maintain their ability to stay close to the centre of the road at approximately the same maximum speed, but showed differences to the original models in their capability to handle traffic light scenarios.

A.3 Feature and Weight Analysis

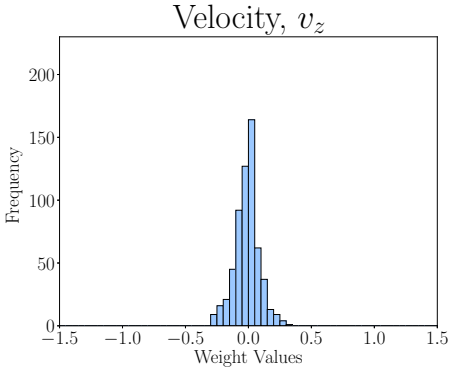
Figure A.5 displays the weight distributions for the reproduced reference model. When comparing the weights that were displayed in Figure 4.5, it can be seen that the distributions look similar in shape. This can also be verified by comparing the Collective Weight Sum (CWS) difference in Table A.3. As can be seen in the table, the differences are on average small, but the reproduced weights tend to be slightly larger which implies that the distributions are more spread and not as peaked around zero.


Velocity, v_x
Frequency
Weight Values

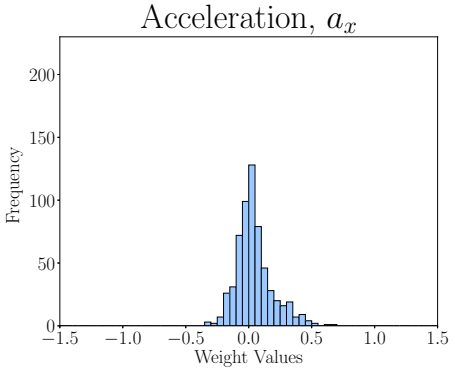
(a)

Velocity, v_y
Frequency
Weight Values

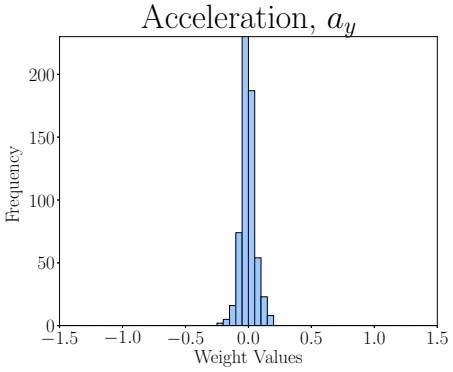
(b)


Velocity, v_z
Frequency
Weight Values

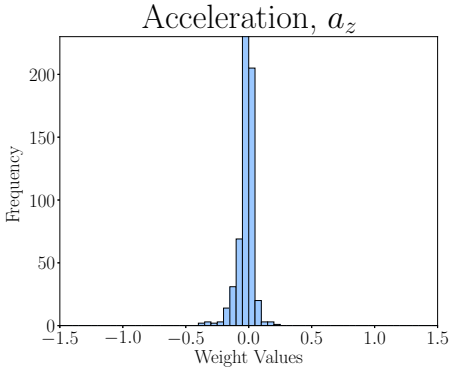
(c)


Acceleration, a_x
Frequency
Weight Values

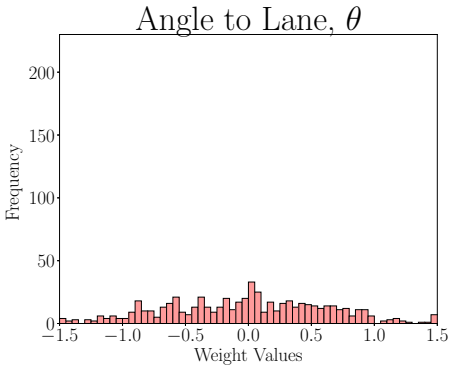
(d)


Acceleration, a_y
Frequency
Weight Values

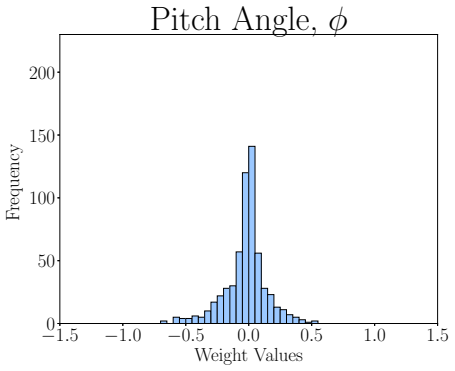
(e)


Acceleration, a_z
Frequency
Weight Values

(f)


Angle to Lane, θ
Frequency
Weight Values

(g)


Pitch Angle, ϕ
Frequency
Weight Values

(h)

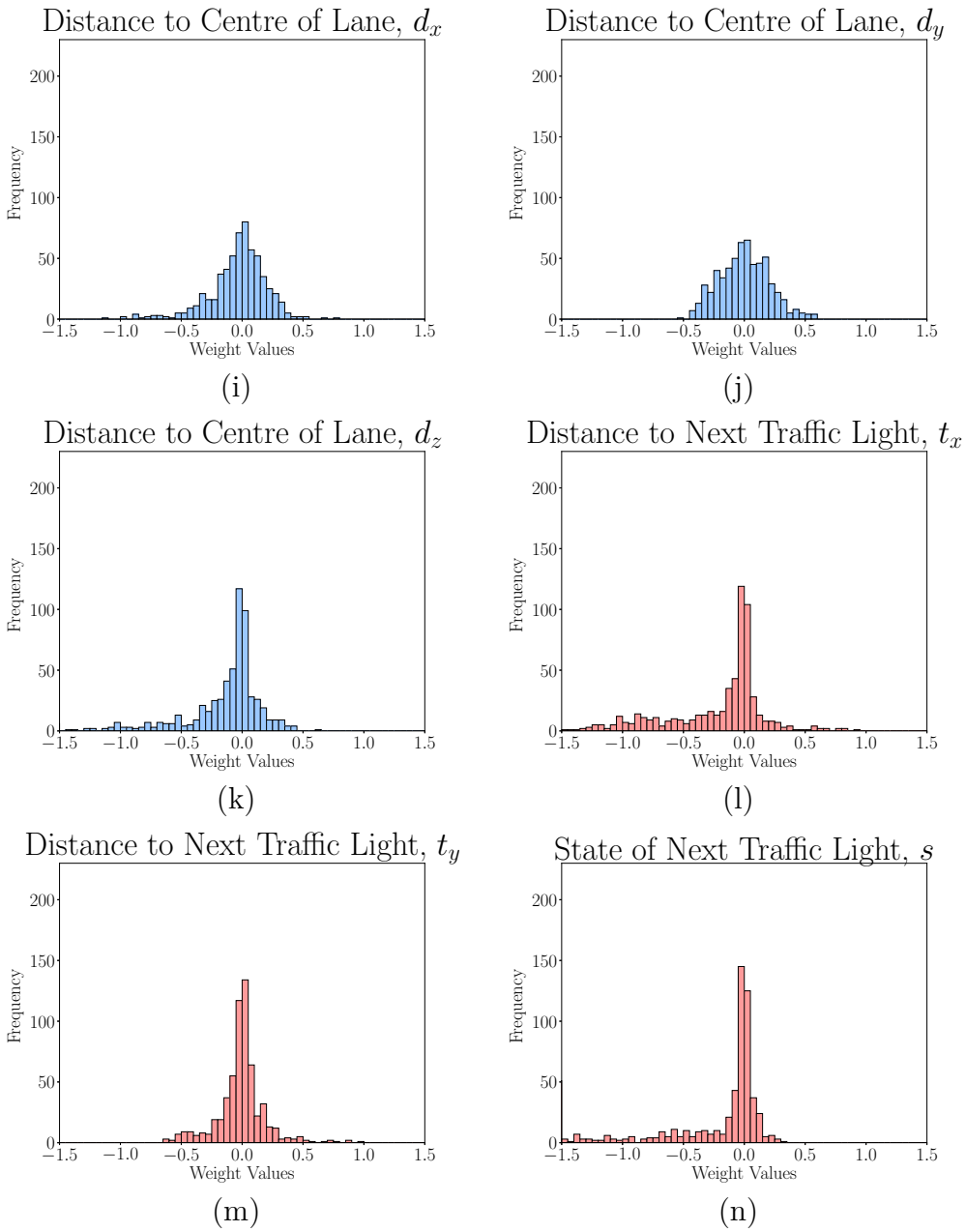


Figure A.5: A comparison of the weight distributions for all 14 features in the fully trained reproduced reference model. Each histogram contains the 600 weights for each feature, indicated by the title, that connect the input layer with the first hidden layer. Only weights that lie between -1.5 and 1.5, i.e. the majority, are shown in the figures. The histograms that are coloured red represents the features that were chosen for the Reason model that used six features.

Table A.3: The Collective Weight Sum (CWS) for all features of the reproduced reference model. The sum was taken over all weights corresponding to each feature's weight distribution in Figure A.5. The third column, Rel. Change, shows the relative change of the CWS compared to the firstly created reference model's CWS, which are listed in Table 4.3.

Features		CWS	Rel. Change [%]
Velocity	v_x	243.0	-9.4
...	v_y	47.6	+1.8
...	v_z	41.6	-4.4
Acceleration	a_x	62.8	-0.4
...	a_y	25.2	+0.2
...	a_z	24.3	-10.9
Angle to lane	θ	309.6	+2.5
Pitch angle	ϕ	68.2	-6.7
Distance to centre of lane	d_x	99.8	+0.4
...	d_y	98.1	+4.7
...	d_z	128.6	-7.1
Distance to next traffic light	t_x	168.0	-11.9
...	t_y	75.6	-10.0
State of next traffic light	s	227.5	-11.2