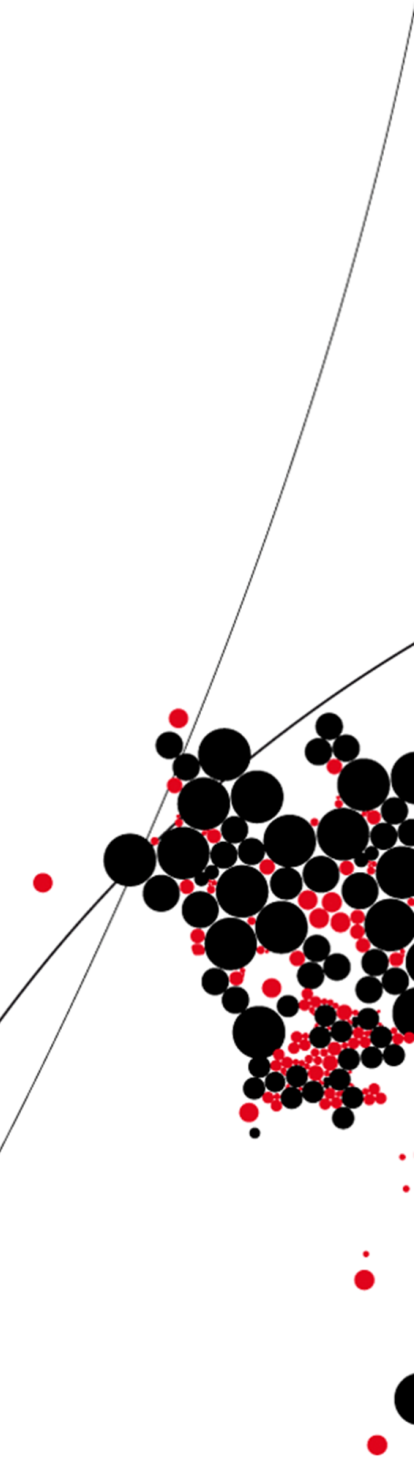
UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering Mathematics and Computer Science



Machine Learning for Cooperative Automated Driving

Aashik Chandramohan

M.Sc. Thesis
August 2018

Supervisors:
prof. dr. ir. Geert Heijenk
Dr. Mannes Poel
Bernd Meijerink. MSc

Design and Analysis of Communication Systems
Faculty of Electrical Engineering Mathematics and Computer Science,
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Summary

Most of the research in autonomous driving currently involves using the on-board sensors on the vehicle to collect the information of the surroundings and using that information for controlling the autonomous vehicle. This research investigates how Machine Learning can be used with cooperative driving for self-driving vehicles. Here the self-driving vehicles can use the vehicle information of the surrounding vehicles to manoeuvre around them. Therefore, it does not need to rely on just its on-board sensors to navigate through the traffic.

In this research Reinforcement learning is used for designing a driving algorithm to control a self-driving vehicle in a highway environment which is simulated using SUMO (Simulation of Urban Mobility). This research focuses on the method to design the driving algorithm which involves choosing the correct input features and actions for the driving agent and the design of the reward structure. It also explains how the performance of the driving algorithm is effected with change in the reinforcement learning parameters. Except the self-driving vehicle, the other vehicles are controlled by SUMO itself. The aim is to check if the self-driving vehicle is able to manoeuvre through the traffic in the highway by overtaking other vehicles and when needed allowing the faster moving vehicles to overtake it.

The driving algorithm is trained and tested in a 2 lane highway environment and a three lane highway environment. It was found that having collision detection as just a part of the rewarding policy did not give the desired results, as the collision percentage was found to be in the confidence interval between 20.5% to 28.4% in a 2 lane highway. Hence collision detection and avoidance was done by a separate entity outside the learning algorithm. This helped in reducing the collision percentage to be in the confidence interval of 0.015 and 0.048 for a two lane system and between 0.08 and 0.14 for a 3 lane system. The research also analyses the performance of the driving algorithm in case of packet loss and change in communication range. A method to cope with the packet loss is also discussed in this report.

This research can be used as a basis for using cooperative driving for self-driving vehicles, but further research needs to be conducted to make the self-driving vehicles safer and reliable as the collision rates achieved using this method is still significantly high.

Acknowledgement

I would first of all like to thank my thesis supervisor, Prof. dr. ir. Geert Heijenk from the Design and Analysis of Communication Systems (DACS) research group of University of Twente for giving me an opportunity to work on this topic. His guidance throughout my research has helped in finishing my thesis on time.

Besides my thesis supervisor, I would also like to thank the rest of my thesis committee members, Dr. Mannes Poel from Human Media Interaction (HMI) research group in University of Twente and Bernd Meijerink. MSc from the DACS research group for their insightful comments and feedback about the research work which has helped me improving this thesis report.

I would like to thank ir. E. Molenkamp, program mentor for the Master program in Embedded Systems in University of Twente for his guidance throughout my Master program.

I would also like the thank the University of Twente and the members of its international office, for admitting me in its Master Program in Embedded Systems and for helping an international student in getting settled in the Netherlands.

I would also like to thank my family and friends for their unconditional support and continuous encouragement during my entire Master program. This accomplishment would not have been possible without them.

Thank you

Table of Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

SAE	Society of Automotive Engineers
LIDAR	Light Detection and Ranging
RADAR	Radio Detection and Ranging
GPS	Global Positioning System
SUMO	Simulation of Urban Mobility
RL	Reinforcement Learning
MDP	Markov Decision Process
DQN	Deep Q Network
DLR	Deutsches Zentrum für Luft- und Raumfahrt
TraCI	Traffic Control Interface
RGB	Red Green Blue
API	Application Programming Interface
GUI	Graphical User Interface
ReLU	Rectified Linear Unit
SD	Standard Deviation

List of Symbols

t	Time Step
S_t	Input state to the agent at time step t
$A(S_t)$	Set of possible action for the agent at input state S_t
A_t	Action taken by the agent at time step t
S	Set of possible input states
R_t	Reward obtained at time step t
S_{t+1}	New state due to action A_t
$Q(s_t,a_t)$	Q value for action a_t to occur when state is s_t at time t
α	Learning rate
Υ	Discount factor
ε	Exploration rate
d_1	Distance between the autonomous vehicle and the vehicle in front in the driving lane for a 2 lane environment. For a generic environment, it is the distance between the autonomous vehicle and the vehicle in front in the current lane. [m]
d_2	Distance between the autonomous vehicle and the vehicle behind in the driving lane for a 2 lane environment. For a generic environment, it is the distance between the autonomous vehicle and the vehicle behind in the current lane. [m]
d_3	Distance between the autonomous vehicle and the vehicle in front in the overtaking lane for a 2 lane environment. For a generic environment, it is the distance between the autonomous vehicle and the vehicle in front in the left lane. [m]
d_4	Distance between the autonomous vehicle and the vehicle behind in the overtaking lane for a 2 lane environment. For a generic environment, it is the distance between the autonomous vehicle and the vehicle behind in the left lane. [m]
d_5	For a generic environment, it is the distance between the autonomous vehicle and the vehicle in front in the right lane. [m]
d_6	For a generic environment, it is the distance between the autonomous vehicle and the vehicle behind in the right lane. [m]
v_1	Velocity of the vehicle in front of the autonomous vehicle in the driving lane for a 2 lane environment. For a generic environment, it is the velocity of the vehicle in front of the autonomous vehicle in the same lane. [m/s]
v_2	Velocity of the vehicle behind the autonomous vehicle in the driving lane for a 2 lane environment. For a generic environment, it is the velocity of the vehicle behind the autonomous vehicle in the same lane. [m/s]

v_3	Velocity of the vehicle in front of the autonomous vehicle in the overtaking lane for a 2 lane environment. For a generic environment, it is the velocity of the vehicle in front of the autonomous vehicle in the left lane. [m/s]
v_4	Velocity of the vehicle behind the autonomous vehicle in the overtaking lane for a 2 lane environment. For a generic environment, it is the velocity of the vehicle behind the autonomous vehicle in the left lane. [m/s]
v_5	For a generic environment, it is the velocity of the vehicle in front of the autonomous vehicle in the right lane. [m/s]
v_6	For a generic environment, it is the velocity of the vehicle behind the autonomous vehicle in the right lane. [m/s]
v_a	Velocity of the autonomous vehicle. [m/s]
v_{acc}	Acceleration rate of the autonomous vehicle. [m/s^2]
L_a	Lane index of the autonomous vehicle
π	Learning Policy
ϑ_0	Initial weights of the neural network
v_f	Maximum allowed velocity of the autonomous vehicle [m/s]
v_d	Difference between the current velocity and the velocity in the previous time step. [m/s]
$R_{positive}$	Constant positive reward
$R_{maxNegative}$	Maximum negative reward
$R_{negativeVariable}$	Variable negative reward
$R_{Collision}$	Reward for collision
$R_{Standstill}$	Reward when $v_a = 0$
R_{near}	Reward when autonomous vehicle is near the vehicle in front
$R_{LaneChange}$	Reward for changing to overtaking lane
R_{Lane1}	Reward for remaining in overtaking lane
$R_{Lane1ApproachingVehicle}$	Reward for being close to a vehicle in front in the overtaking lane
$R_{OverSpeeding}$	Reward for crossing the Speed limit
$R_{Speeding}$	Reward for accelerating
$R_{MaintainSpeedLimit}$	Reward for being in the speed limit.
d	Distance considered as close to the surrounding vehicle [m]
L_{max}	Maximum Lane index in the given highway
r	Communication range [m]
Z	Communication Error Probability

1. Introduction

1.1 Motivation

With the current developments in technology, self-driving vehicles also known as Autonomous vehicles, have become a reality. Autonomous vehicles are the type of vehicles that are capable of sensing its environment and navigating without human input [1] [2].

There are different levels of autonomous driving defined by the Society of Automotive Engineers (SAE). These levels are shown in Figure 1. Until level 3, the vehicle is not fully automated and it still requires a driver for taking certain decisions. There are already vehicles in market that support level 2 automation and Audi have also been able to integrate level 3 automation in their new A8 sedan [3]. Intensive research is being done to bring full automation in driving and it won't be long before we have a level 5 autonomous vehicle in the market. This research is related to level 4 and level 5 automation.

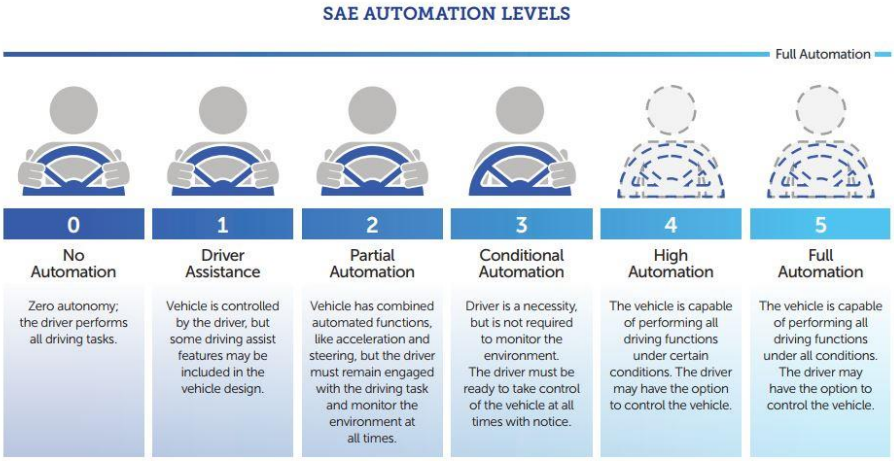


Figure 1 Different Automation Levels [4]

Most of the research being done for level 4 and level 5 automation is done using several sensors for finding the information of the surrounding environment as shown in Figure 2.

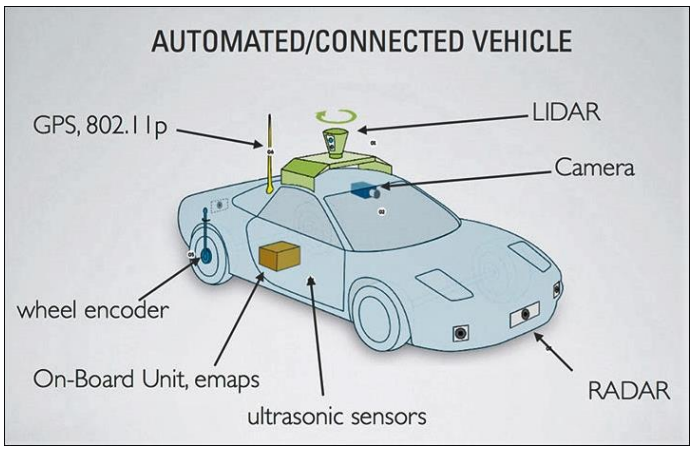


Figure 2 Sensors used in an autonomous vehicle [5]

In this research a different approach would be analysed for fully autonomous vehicles. Here machine learning would be used with cooperative driving. In cooperative driving, the vehicles would be able to communicate with each other and the environment around it as shown in Figure 3. Hence with this approach, the vehicle parameters of the surrounding vehicles can be attained. These vehicle parameters could consist of the vehicle's position and their velocity, which can be used by the machine learning algorithm of the autonomous vehicle to manoeuvre it through the traffic. Here the self-driving vehicle takes its decision based on the information it received from the vehicles around it.

Figure 3 Cooperative Driving. Image taken from Sproul Company [6]

Using cooperative driving would involve more reliance on the sensors required for the inter vehicle communication and the communication network and less reliance on the on board cameras and other sensors of the autonomous vehicle. A learning system involving cooperative driving would be more advantageous, as us-

ing this the vehicle can get the information of all the vehicles within the communication range. Hence the autonomous vehicle can make better decisions than what is possible by using just the sensors in the vehicle that give the information of just the immediate surrounding. To illustrate this point here is an example. Imagine a single lane road. In cooperative driving the autonomous vehicle will not only get the information of the vehicle just in front of it, but also of the vehicle which is way ahead of it provided it is within the communication range. Hence when that vehicle starts to brake the autonomous vehicle will know that it would also need to decrease its speed in some time and hence can already start preparing to gradually reduce its speed. Now if the autonomous vehicle was not using cooperative driving, then it would have to rely on its on board cameras and other sensors for the information of the surrounding vehicles. In this case the information that it would get would be just the information of the immediate vehicle in the front. Hence only when it starts to brake will the autonomous vehicle also have the knowledge that a braking action needs to be performed. Hence the reaction time it has to do this action is less than what it had with cooperative driving. Another advantage of using cooperative driving, is that the vehicles can communicate with each other before taking an action. For example, if two vehicle want to change to the same lane which might cause a collision, then they can negotiate with each other using the communication network and decide who can change to that lane.

Hence using cooperative driving can drastically improve driving efficiency by reducing the number of accidents.

1.2 Research Goals and Contributions

The aims of this research are:

1) To investigate to what extent reinforcement learning using Q learning can be used with cooperative driving to manoeuvre an autonomous vehicle in a multi-lane highway.
2) To investigate the influence with change in the number of hidden layers and the number of nodes in a deep Q network on the performance of the driving algorithm by monitoring the number of collisions initiated by the autonomous vehicle and the average speed it travelled in, with these settings.
3) To investigate the role of communication range on the performance of the driving algorithm by again monitoring the number of collisions with the autonomous vehicle.
4) To investigate the effect of packet loss due to communication error on the performance of the driving algorithm. The performance is again estimated based on the number of collisions caused by the autonomous vehicle.

The contributions done to achieve the above mentioned goals are:

1) A method to create a driving algorithm is designed using which a driving algorithm is developed based on Q learning technique, a type of reinforced machine learning technique for a self-driving vehicle in a highway environment. The algorithm was developed for a two lane highway with one driving lane and an overtaking lane. The algorithm was later extended to also support multi-lane highway environment with more than one overtaking lane.
2) The establishment of an environment for training and testing cooperative driving algorithms, based on SUMO (Simulation of Urban Mobility) [7] and Q learning algorithm.
3) An error concealment technique was designed for the driving algorithm to follow in case of packet loss due to communication error.

The entire research is conducted with respect to the basic driving rules. Hence the vehicles keep to the right lane and overtaking is done through the left side.

1.3 Structure of the Report

To achieve the various goals presented in Section 1.2, a systematic procedure was followed. Hence at the start, previous works were studied to find which type of machine learning technique can be used for autonomous driving and why reinforcement learning is the best type of machine learning technique for this research. Previous research on using cooperative driving for self-driving vehicles were also studied, which gave a basis for this research. These are explained in Chapter 2.

The methodology followed for designing and evaluating the driving algorithm in this research is explained in Chapter 3. It gives a general idea of how the overall research is conducted.

An important part for this research is to be able to test the developed driving algorithm. As using a real vehicle and traffic scenario would prove costly and require more time, it was decided to use simulations to prove the efficiency of the driving algorithms designed in this research. Chapter 4 introduces the simulation environment (SUMO) used in this research and how it is linked with the driving algorithm that is developed using the Python programming language.

Based on the knowledge gained from previous works stated in Chapter 2, different approaches were analysed to create the driving algorithm to manoeuvre a self-driving vehicle in a 2 lane environment. Analysis was done on how to choose the input features, the actions and the rewards for the driving algorithm. These are discussed in Chapter 5 . This chapter also discusses the various measures that can be used to determine the performance of the driving algorithm, as the aim of the research is to develop an efficient driving algorithm.

Based on the analysis in Chapter 5, a driving algorithm was designed and implemented. This was then tested in SUMO with different Deep Q learning configurations and reward values. The results of these simulations gave an understanding of how the various Q learning parameters and reward configurations influence the performance of the driving algorithm. These results were used to further improve the driving algorithm. Chapter 6 describes these initial and intermediate designs and how it affected the performance of the driving algorithm. Hence one of the research goals which is to find the influence of various Q learning parameters on the performance of the driving algorithm is discussed in this chapter.

The Final design and implementation is described in Chapter 7. This was designed based on the knowledge gained from the intermediate iterations of designs explained in Chapter 6. Chapter 5 only described the driving algorithm in a 2 lane highway environment. Chapter 6 extends this to a multilane highway environment so the driving algorithm developed is a generic algorithm that can be used in a highway with more than 2 lanes. This chapter also talks about the performance of the driving algorithm in case of loss of data due to communication error and how this problem can be overcome.

Chapter 8 describes the results of the final implementation given in Chapter 7, based on which conclusions are drawn and recommendation given in Chapter 9.

2. Background and Related Work

This chapter gives a short description of different types of machine learning techniques and gives an introduction about reinforcement learning and gives the reason why this is the preferred learning method for this research. An introduction about Q learning, a type of reinforcement learning is also given in this chapter.

It also talks about the approach taken by two previous research papers by Shota Ishikawa and Sachiyo Arai [8] and Xin Li, Xin Xu and Lei Zuo [9], which used reinforcement learning with cooperative driving.

2.1 Machine Learning

Machine learning can be mainly classified into three types [10], namely:

1) Supervised learning
2) Unsupervised learning
3) Reinforcement learning

In supervised learning, an algorithm is created based on training data. There is already existing data with inputs and its corresponding outputs. This data is used to create the learning algorithm, which can then be used on new inputs to predict the output. An example of the application of supervised learning is in spam filtering of emails, where the algorithm can be first trained with the emails that are correctly marked as spam or not and later use the algorithm on the new emails to classify them as spam or not. In case of driving, there is no definite outputs for each scenario and also it is not possible to train the driving algorithm for each driving scenario that can occur. Hence supervised learning cannot be used for designing the driving algorithm.

In unsupervised learning, the output of the system is not known. The algorithm uses the inputs available and group them based on their similarities or differences to produce the outputs. An example would be to differentiate between cats and dogs. If a learning algorithm is given random pictures of cats and dogs, it would be possible to differentiate the pictures into two groups of cats and dogs based on the similarity in the pictures of cats and that of dogs [11]. Unsupervised learning is quite unpredictable in complex tasks as it does not have any knowledge of the expected output and relies completely on the inputs to find the output. Hence it is not a good learning algorithm to be used in autonomous driving.

Reinforcement learning [8] is a type of learning algorithm that adapts the learning algorithm based on changes in the environment. Therefore, the autonomous vehicles can adapt to different traffic scenarios. Here, a specific output is not known

beforehand for any input, but goal of the algorithm is predefined and the learning algorithm uses rewards to decide which output to choose such that the goal is reached. In the case of an autonomous vehicle, one of the important goals is to avoid collision, hence this can be used in reinforcement learning such that the output produced from the driving algorithm does not lead to collisions. In a similar way other general requirements of driving can be used for designing a driving algorithm based on reinforcement learning [12]. Reinforcement learning is explained in more detail in the next section.

Hence most of the research papers currently published like by Shota Ishikawa and Sachiyo Arai [8] and Xin Li, Xin Xu and Lei Zuo [9] use reinforcement learning for autonomous driving.

Reinforcement Learning

Reinforcement learning involves learning what to do and how to map situations to actions so as to maximize a numerical reward signal. [13] This can be modelled as a closed loop system shown in Figure 4. Here the Agent is the learner and the decision maker and the Environment is the thing it interacts with. The interaction between the agent and the environment happen at each of a sequence of discreet time steps, t= 0, 1, 2, 3, … [13]. The Agent selects the action (A_t) from the set of possible actions,$A_t \in A(S_t)$, where $A(S_t)$ is the set of actions available in state S_t. S_t, $S_t \in S$ (S *is set of possible states*) is the state of the environment at time t, which is the input to the agent. This action leads to a reward (R_t) which is used by the agent to learn about the most effective action to take for that particular input state. Also the action taken by the agent will influence the environment and change its state to a new state (S_{t+1}), which would be the new input to the agent. The aim of the agent is to maximise the reward it gains over time.

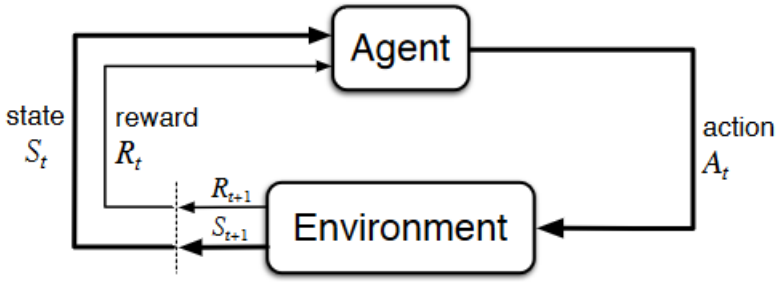


Figure 4 Agent environment interaction in reinforcement learning taken from [13]

Rewards

Rewards are one of the most important parts of reinforcement learning. The agent decides the action it should take based on the reward it gets it. At each time

step, the reward is a simple number, $R_t \in \mathbb{R}$. As mentioned in the previous section, the goal of the agent is to maximise the overall reward in the long run and not just to select the action which gives a large reward in the current time step [13]. For example, an agent could take an action with a lower reward to reach a new state, which in turn opens up new possibility of action with higher rewards such that the total reward at the end is higher than if it had chosen the action with the higher reward at the start.

Choosing a good rewarding scheme is critical for a good learning algorithm. For example, in a learning algorithm which controls the movements of a robot trying to escape from a maze, the reward is often -1 for every time step it remains inside the maze, so that the learning algorithm is encouraged to get the robot out of the maze faster [13] and a large reward when it finally gets out of the maze. It is important to give rewards such that the goal of the agent is finally achieved. The agent always tries to maximize its total rewards, hence if positive rewards are given for even intermediate steps, then it could be possible that the agent finds a way to maximize its reward without actually reaching its goal, which in the previous example is to get out of the maze. Hence the reward scheme is a way of communicating to the agent what needs to be achieved and not how it has to be achieved [13].

Model of the Environment

The model of the environment is one of the elements of Model based reinforcement learning. This allows the agent to know how the environment would behave for a particular action even before the action is taken. That is given a state and action, the model can predict the resultant next state and the reward. Hence they are used for planning a course of action by considering possible future situations before they are actually taken [13].

Reinforcement learning methods that use Models are called model based methods and the reinforcement learning methods that do not use Model but uses trial and error for learning are called model free methods.

Model based methods requires another entity called the transition probability function where $T(s_{t+1}|s_t, a_t)$ specifies the probability of reaching state s_{t+1} from state s_t when action a_t is taken. The problem though is that the transition probability and the reward values for each state action pair is not known beforehand. Hence the agent needs to first learn the model of how the environment works by observing the different states possible and the actions which leads to these states and also the rewards that are attained by these actions. Using this the agent can create the transition probability function and the reward function which can then be used as a planning algorithm to take the best possible decisions so as to get maximum overall rewards.

In autonomous driving, the number of states of the environment is not fixed as this is related to the positions of the surrounding vehicles and also their velocities. Hence there would be a large number of input states possible. Hence it would be difficult to learn the transition probability function for such an environment. Hence model free learning methods are used in such cases. One such model free learning method is Q learning which is discussed in the next section.

Q Learning

Q learning is a model free based reinforcement learning. It provides agents with the capability of learning to act optimally by experiencing the consequences of actions, without requiring them to build maps of the environment domain [14]. In Q learning a particular action is taken based on Q values. The formula for finding the Q value is given in Equation (1) below taken from [15].

$$Q(s_t, a_t) \leftarrow (1-\alpha) \times Q(s_t, a_t) + \alpha \times (r_t + \gamma \times \max Q(s_{t+1}, a)) \quad (1)$$

In the above equation, s_t is the current input state to the agent and a_t is the action taken by the agent. $Q(s_t,a_t)$ is the Q value for the action a_t to occur when state is s_t at time t. At the start the Q values are initialized as random values. During the learning stages the Q values are updated based on Equation (1). In the equation α is the learning rate which determines to what extent the newly acquired information overrides the old information. Hence α could determine how quickly the Q learning converges. Q learning is said to have converged if each state action pair is performed infinitely often [16]. If the learning rate is low, then according to Equation (1), the Q values would change slowly over each iteration as the current Q value would be given more prominence. Hence for the Q learning to converge it would take a lot of training. However, the advantage of having a low learning rate is that at the end the Q learning will surely converge as the Q values vary really less during each iteration. With a large learning rate, it is possible that the new Q value overshoots the convergence point, such that the Q learning does not converge [17]. r_t is the reward that is got for performing the action a_t from state s_t. γ is the discount factor. It determines the importance of the future rewards. A lower discount factor means the algorithm would focus on current rewards as $\gamma \times \max Q(s_{t+1}, a)$ will become negligible. With increase in γ, it would try for higher long term rewards. The term max $Q(s_{t+1}, a)$ signifies the maximum Q value from the next state for all possible actions. Hence $Q(s_t,a_t)$ is the expected discounted reward for executing action a_t from state s_t, which maximizes the total discounted expected reward.

Using the given Equation (1), the Q values are updated with change in the environment and the action is taken based on the action contributing to the highest Q value from the current input state. At the start the learning algorithm is uninformed. Hence it chooses actions randomly and as it gets educated, the actions

would be chosen based on the Q values. Hence for this the exploration rate (ε) is used. A higher ε allows the algorithm to choose the actions randomly. The reason for using the exploration rate is explained in detail in [18].

Deep Q Learning

In the Q learning algorithm, the Q values are used for prescribing the action to be taken by the learning algorithm. For example, if there are 4 possible input states and 2 actions. Then the Q value matrix will be a 4×2 matrix such that there will be a total of 8 Q values. Depending on the current state the action which contains the maximum Q value is selected.

In complex learning algorithms, the input states can have different values. Hence a simple Q table based on the input state and action pair cannot be created. This is where Neural networks are used. In this method, the look up table of Q values is replaced with a neural network. In a neural network, the input features are taken as the input to the neural network and the output actions are taken as the output of the neural network. Hence to determine an action to be taken, the learning algorithm would run the network once for every action and select the action that gives the highest output of the network from the given input state based on the weights of nodes in the neural network [19].

At the start of the learning process the neural networks are assigned with random weights. The feedback from the environment which in this case would be in the form of rewards is compared with the expected reward to adjust its weights and improve its interpretation of state action pairs [20].

Experience Replay

Experience replay can be used with Deep Q learning for improving the efficiency of the learning algorithm. In a standard learning algorithm, the knowledge gained by a certain action is discarded as soon as the action is performed [21]. The experience from each time step like the input state to the agent (s_t), action performed (a_t), reward achieved (r_t) and the new state (s_{t+1}) got due to the action (a_t) can be stored and this information can again be used to retrain the learning algorithm. This will allow the learning algorithm to learn better during the training period, as it does not just learn from actions taken during the simulation but also from the results of the previous simulations.

A pseudo code of how experience replay works for Q learning [21] is shown below in Algorithm 1. A memory buffer is created with size N which stores the current state, action, reward and the next achieved state during execution of each time step. If the buffer is full, the data inside is overwritten, in a first in first out format. That is if N is 100. Then at any time it can hold information of the last 100 time

steps. For experience replay, m number of experiences are taken at random from the buffer and this is used to again retrain the neural network.

Input: *memory size N, minibatch size m, learning rate α, discount factor γ, Total steps T, initial weights ϑ_0, update policy π_ϑ.*

Initialize replay memory buffer with capacity N

Observe initial state s_0

For *t=1 to T* ***do***

    Take action a_t based on $\pi_\vartheta(s_t)$

    Observe r_t and s_{t+1}

    Store transition (s_t, a_t, r_t, s_{t+1}) in the memory buffer

    For *j=1 to m* ***do***

        Sample a transition (s_i, a_i, r_i, s_{i+1}) randomly from the memory buffer

        Retrain the neural network and update its weights

    End for

End for

Algorithm 1 Reinforcement learning with experience replay based on [19]

2.2 Previous Research Work

In the researched papers [8] and [9], a driving algorithm is developed to manoeuvre through traffic in a one way environment. In the paper by Shota Ishikawa and Sachiyo Arai [8], this is done with the autonomous vehicle applying brake whenever required. Xin Li, Xin Xu and Lei Zuo gets the autonomous vehicle to change lanes and overtake the slower moving vehicles in their paper [9]. Both these papers have similar input features, which consists of:

1) The velocity of autonomous vehicle
2) The distance with the vehicle in front

In addition to these two features Xin Li et al. also uses, few more input features which is required for the vehicle to safely overtake. They are:

1) Velocity of the other vehicles around it
2) Maximum allowed velocity for the autonomous vehicle
3) Distance with the other vehicles
4) Current lane, the autonomous vehicle is in.

The other vehicles are the vehicles directly in front and behind of the autonomous vehicle in the driving lane and the overtaking lane as shown in Figure 5. Here d_1, d_2, d_3 and d_4 are the distance of the autonomous vehicle with the immediate vehicles in front and behind it and v_1, v_2, v_3 and v_4 are their respective velocities

and v_a is the velocity of the autonomous vehicle being controlled by the driving algorithm and v_f is its maximum allowed speed. The current lane the autonomous vehicle is in is represented by l.

Hence the current state $s_t = \{v_{a,t}, v_{f,t}, v_{i,t}, d_{i,t}, l_t\}$, i= 1,2,3,4

The possible input states for this driving agent is $S = \{s_t \mid t=0,1,2, 3...\}$

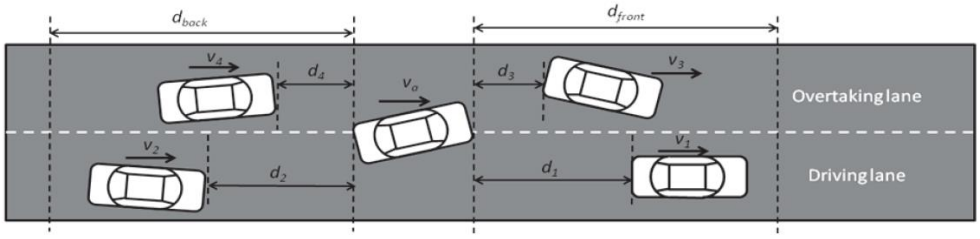


Figure 5 Vehicle position layout used in [9]

In [8], the action is to brake or not to brake. Hence here the autonomous vehicle tries to follow the vehicle in front without colliding. In [9] though, as the aim is to overtake the vehicle in front, the action set is a little larger. It consists of 4 actions:

1) Move to Overtaking lane
2) Move to Driving lane
3) Stay in Overtaking lane
4) Stay in Driving lane

The reward is given in such a way that the maximum negative reward is given to the most undesirable result and the reward value increases as results become desirable. The reward system used by Xin Li et al. in [9] is shown as an example in Table 1.

Reward Value	Condition
-300	c=1
-150	$c = 0$ & $v_d > 10$
$-0.1v_d$	Else & $v_a > v_f$
$v_a - v_f - 0.1v_d$	Else & $l = 2$
$v_a - v_f - 0.1v_d - (d_1 - v_1)$	Else & $l = 1$

Table 1 Reward function used in [9]

Here a reward of -300 is given if a collision occurs. A reward of -150 is given if the velocity changes abruptly by more than or equal to 10 m/s between two time steps. v_d is the velocity difference between the current moment and the last moment. v_a is the velocity of the autonomous vehicle, v_f is the maximum allowed velocity of the autonomous vehicle, d_1 is the distance between the autonomous vehicle and the vehicle in front and v_1 is the velocity of the vehicle in front. l=1 signifies that the

autonomous vehicle is the driving lane and l =2 signifies that it is in the overtaking lane [12].

The performance of the driving algorithm using machine learning was compared with the traditional method without using machine learning. In the traditional method, the action to be done is fixed based on the values of the distances between the vehicles and their velocities as shown in Table 2.

The Current Lane	Action	Condition
1	Move to Lane 2	$(d_l < 50 \mid v_l\text{-}v_f < -3)$ & $d_4 > 80$ &$(d_3 - d_1 > 10)$ & $(v_3\text{-}v_1 > 1)$
	Keep in Lane 1	Else
2	Keep in Lane 2	$((d_1 > 150$ & $v_1 - v_f > 1) \mid v_1 - v_3)$ & $d_2 > 60$
	Move to Lane 1	Else

Table 2 Decision making without using Q learning [9]

It was found that the performance of the algorithm was better using machine learning than by using the traditional method.

In this research a combination of the researches done by Shota ishikawa [8] and Xin Li [9] is performed. The autonomous vehicle will be able to change lanes accelerate and also decelerate to manoeuvre through the traffic in a highway environment. Also in this research SUMO [22] is used as the simulation environment and the learning algorithm is developed in Python.

Detailed analysis of the related work was done during the research phase and is given in [12].

3. Methodology

This chapters gives the methodology followed for designing and evaluating the driving algorithm to control an autonomous vehicle in a highway environment using the vehicle parameters of the immediate surrounding vehicles.

3.1 Reinforcement learning

Based on Chapter 2.1, Reinforcement learning using Deep Q-learning is used for designing the driving algorithm. Two types of driving algorithm are designed. The first one is designed to manoeuvre a vehicle in a 2 lane highway. This is then modified support a generic highway environment where the number of lanes are not fixed.

In this research the driving algorithm consists of the Driving agent and the environment as shown in Figure 4 in Chapter 2.1. This environment should not be confused with the highway environment where the simulation takes place. The environment of the driving algorithm is also a part of the autonomous vehicle. It senses the surrounding positions of the vehicles and determines the current state of the autonomous vehicle. It is also responsible for performing the actions given by the driving agent and calculating the resulting rewards. Hence for this research Figure 4 is modified as shown in Figure 6 .

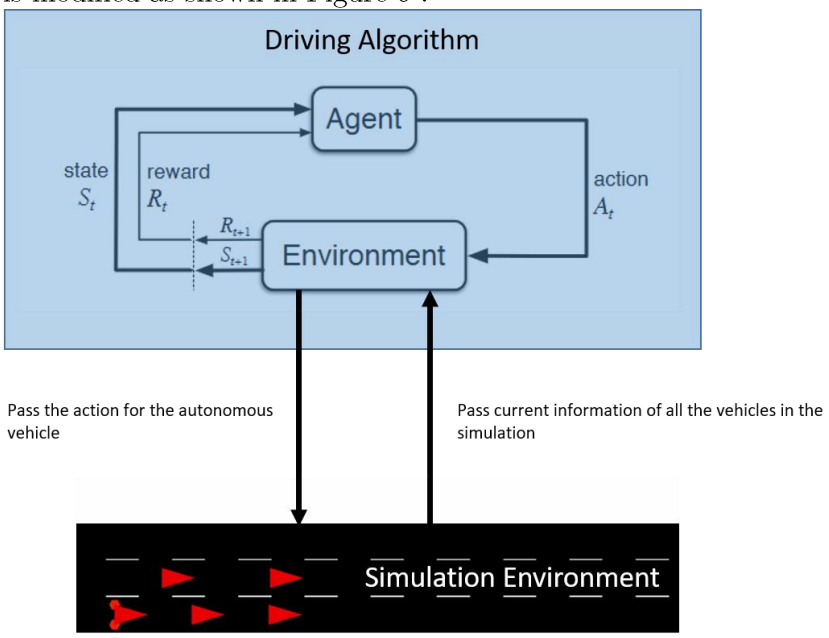


Figure 6 Interaction between the driving algorithm and the simulation environment

Here the simulation environment is the actual surrounding for the vehicle. In this research it is a highway environment. The driving algorithm is part of the autonomous vehicle. The environment of the driving algorithm would include the

communication receiver in the vehicle which receives the information of all the surrounding vehicles within the communication range and the electronic control unit which can use the received information to create the input state for the agent. It also takes the output action from the agent for controlling the vehicle accordingly and calculates the reward.

Rewarding Scheme

The goal of the autonomous vehicle is to move forward in a good speed on the highway without colliding with other vehicles. Hence the rewarding scheme is designed such that this goal is attained. Details regarding the rewarding scheme used are given in Chapter 5.3. Two types of rewarding scheme are discussed, one which gives importance to short term goals like changing lanes, overtaking etc. and the other one gives importance to overall goals like maintaining speed limit, etc. Finally, the rewarding scheme which gives importance to overall goals was implemented as it can be easily modified to support the autonomous vehicle in a multi-lane environment.

Input State

The input state set of the driving algorithm should contain the current features of the autonomous vehicle and the vehicles immediately surrounding it, as this information is required by the agent to make the decision on how the autonomous vehicle should move so as to avoid collision with its neighbour. Hence the input state used is similar to the one discussed by Xin Li et al. which is discussed in Chapter 2.2. The input state set used in this research for a two lane highway is discussed in Chapter 5.1 and for a multi-lane highway is discussed in Chapter 7.1.

Actions

The major actions of a vehicle when in a highway is to either change the lanes or to accelerate or brake. Hence the possible actions by the driving agent are also the same. The action set consists of 4 features which include moving to the overtaking lane, moving to the driving lane, accelerate and decelerate, for a two lane highway environment. The selection of action features for a 2 lane highway is given in more detail in Chapter 5.2. For a multi-lane highway, the action state features will change as the number of lanes are not limited. Hence the possible actions will now be among changing to the left lane, changing to the right lane, accelerate, decelerate or idle action which means no action is performed. This is explained in Chapter 7.2.

Training and Testing Period

The agent needs to first learn about the environment and find out which actions are best suited for each state. Based on this the neural network is trained such that the optimal action is selected for each state. Hence Training period is used for training the neural network. At the start the agent selects actions at random and as the training continues, it starts taking less and less random actions and more actions based on the calculated Q values for each action from the corresponding state. This is where the exploration rate ($\boldsymbol{\varepsilon}$) comes into play. Higher the $\boldsymbol{\varepsilon}$ higher the chance of actions being taken in random and as the exploration rate decreases the action starts being taken based on the Q values.

In this research the training of the neural network is done using experience replay explained in Chapter 2.1. The Q learning algorithm used in this research is given in Algorithm 2 and is based on [23]. For learning the experiments are done in the form of episodes. One episode is one full simulation of the autonomous vehicle. An episode finishes when either the number of time steps reaches T or the autonomous vehicle collides with another vehicle. The learning happens during each time step. As already explained in Chapter 2.1, the current state, action, reward and new state is stored in a memory buffer during each time step. In this algorithm another parameter is also used which tells if the episode has ended or not. As already mentioned, an episode ends if there is a collision or the time step reaches its maximum which is T in Algorithm 2. This is represented by the done variable. From this memory, random number of elements are sampled which are used to calculate the Q values. If the simulation had ended in that time step (done = true), then there won't be any next state possible hence $\max_{valid\ a} Q(s_{j+1}, a) = 0$. Therefore in such cases $Q_{target}(s_j,a_j)$ = the reward gained in that state. In other cases, the Q_{target} value needs to be calculated and using this the neural network is trained. After each episode as the neural network is getting trained the exploration rate is decreased by a small factor to decrease the random selection of actions by the agent. Once the algorithm is full trained the selection of the action will be based on the Q values.

Once the neural network is trained it can be tested on new simulations. The testing algorithm is similar to the training algorithm shown in Algorithm 2, except that here the actions are always selected based on the maximum Q value and as the algorithm is already trained, the neural network need not be trained during testing.

For *episode = 1 to N*

For *time step t = 1 to T*

With probability ε select a random valid action a_t

Otherwise select the valid action a_t *with the maximum Q value*

Get the reward r_t *and state* s_{t+1} *and done status*

Store $(s_t, a_t, r_t, s_{t+1}, done)$ *in memory buffer*

For *memory*$(s_j, a_j, r_j, s_{j+1}, done)$ *in sampled minibatch*

Perform gradient descent on $Q(s_j, a_j)$ *using the target value*

$$Q_{target}(s_j, a_j) = \begin{cases} r_j & done\ is\ true \\ r_j + \gamma \max\limits_{valid\ a} Q(s_{j+1}, a) & otherwise \end{cases}$$

Retrain the Deep Q network

End for

End for

Decrease ε

End for

Algorithm 2 Deep Q learning algorithm adapted from [20]

3.2 Testing the Driving Algorithm

Testing of the driving algorithm requires a simulator where the highway environment can be modelled. In this research SUMO (Simulation of Urban Mobility) [22] is used for modelling a highway environment to run the traffic simulation. The highway consists of some traffic which includes a number of slow moving vehicles and a number of fast moving vehicles, so that the autonomous vehicle will be able to learn to overtake and also at the same time will learn to allow the faster moving vehicles to overtake it by not unnecessarily occupying an overtaking lane. To test the performance of the driving algorithm in an ideal traffic situation, all other vehicles are considered to be following the traffic rules. That is, they maintain a certain safe distance between the vehicle in front of them and only change lanes for overtaking and also maintain the speed limit. This means that none of the collisions in the simulation would be initiated by the other vehicles and if any collision were to occur, it would be due to a mistake by the driving algorithm of the autonomous vehicle.

The frequency of generation of the traffic needs to be random so that the agent can learn for different conditions. If the traffic is not random, then each episode will be exactly the same, and hence the driving algorithm will not learn for all the conditions. With traffic generation being random, it is possible that the autonomous vehicle is inserted into the simulation just at the next time step after insertion of another vehicle. Hence at such times it is possible for a collision to occur as the autonomous vehicle will not have enough time to react. These types of situation

should be avoided and in this research if the collision occurred due to such a situation then that result is not considered.

In cooperative driving the vehicle parameters of all the vehicles within the communication range is received by the autonomous vehicle. The driving agent though only requires the vehicle parameters of the immediate surrounding vehicle to the autonomous vehicle as its input features. Hence an algorithm to find the immediate surrounding vehicles from all the vehicle information is designed. This is discussed in detail in Chapter 6.1.

One of the goals in this research is to test the behaviour of the driving algorithm with change in communication range and during packet loss. As a simulation environment is used for testing the driving algorithm, these scenarios need to be implemented separately. The simulation environment gives information of all the vehicles active in the simulation. Hence to simulate unavailability of vehicle information of the vehicles outside the communication range, the environment in the driving algorithm only considers the positions of the surrounding vehicles within the communication range for the input state of the driving agent. For simulating packet loss. A random function is used in the environment of the driving algorithm that feeds the default values as the features for the input state of the driving agent. These functions are discussed in detail in Chapter 7.4.

3.3 Evaluation of the performance of the Driving algorithm

The performance of the driving algorithm is assessed based on how well the autonomous vehicle manoeuvres on the highway. As the main goal of the algorithm is to avoid collision with other vehicles, collisions are taken as one of performance criteria for the driving algorithm. Other than this the average speed of the autonomous vehicle is also taken as another performance criterion. Measuring the number of overtakes by the autonomous vehicle is difficult with the experiment setup in this research as SUMO does not give the count of the overtook vehicles. Also writing an algorithm for measuring this would be quite complex as it requires to keep track of the vehicles in front of the autonomous vehicle and record when that particular vehicle goes behind the autonomous vehicle. Also it is not necessary that each time the autonomous vehicle changes to the overtaking lane, it overtakes a vehicle. Hence lane changes cannot be used for measuring the number of overtakes. It is however possible to tell if the vehicle has done any overtakes from its average speed. If the average speed of the vehicle is higher than the maximum speed of the slower moving vehicle in the simulation, then it means that the autonomous vehicle would have overtaken a few slower moving vehicles. For this the traffic density should be such that there are slower moving already in the highway before the autonomous vehicle is inserted. Another performance criterion to assess the driving

agent is to observe the total rewards achieved in each episode. The aim of the learning agent is to maximize the total reward.

Details of how to evaluate the performance of the driving algorithm is given in Chapter 5.4.

4. Experiment Setup

To test the driving algorithm in a highway environment, SUMO (Simulation of Urban Mobility) [7] is used. This chapter gives an introduction about SUMO and describes how the simulation environment is set up using SUMO.

The driving agent is designed in Python using Keras [24] and Tensorflow [25]. An introduction about these Machine learning libraries are also given in this chapter.

Finally, the interface between the simulation environment and Learning algorithm is described.

4.1 SUMO - Simulation of Urban Mobility

SUMO is a free and open traffic simulation suite by DLR [26] Institute of Transportation Systems [7]. It allows design of different types of road networks and traffic scenarios. This section describes how the highway environment is designed and configured using SUMO.

Design of the Road Layout

The highway environment is designed in SUMO using the tool NetEdit [27]. NetEdit allows the user to design road layouts and set rules to be maintained while driving on them. Figure 7 shows the highway layout designed in NetEdit.

The left side pane in Figure 7 shows the properties of the designed path. Paths are called as edges in SUMO. 'Lane' is the id of this edge. Ids are required for modifying the parameters through TraCI (Traffic Control Interface) and also for designating the routes the vehicles should take. Networks of roads are created using series of junctions. Junctions are the end points of the edge. As in this research a straight road is used, just one edge is required and hence two junctions are used which are denoted by 'gneJ0' and 'gneJ1' and their positions are (0,0) and (5000,0) respectively in the x, y axis. The speed parameter denotes the maximum speed to be maintained (speed limit) in this edge. For this research the maximum speed is set to 22.22m/s (80 Km/hr). The priority parameter denotes the priority of the vehicles going through the particular edge. This is used in junctions connecting more than one edge for denoting the right of way for the vehicles. Higher the value, more the priority. In this research as only one edge is used, the priority value is 'no value'. The 'numLanes' denotes the number of lanes for the edge. Initially this is set to 2 for having a driving lane and an overtaking lane. The 'allow' and 'disallow' fields are used to regulate the type of vehicles entering the edge. In this research the only type of vehicles used are passenger cars.

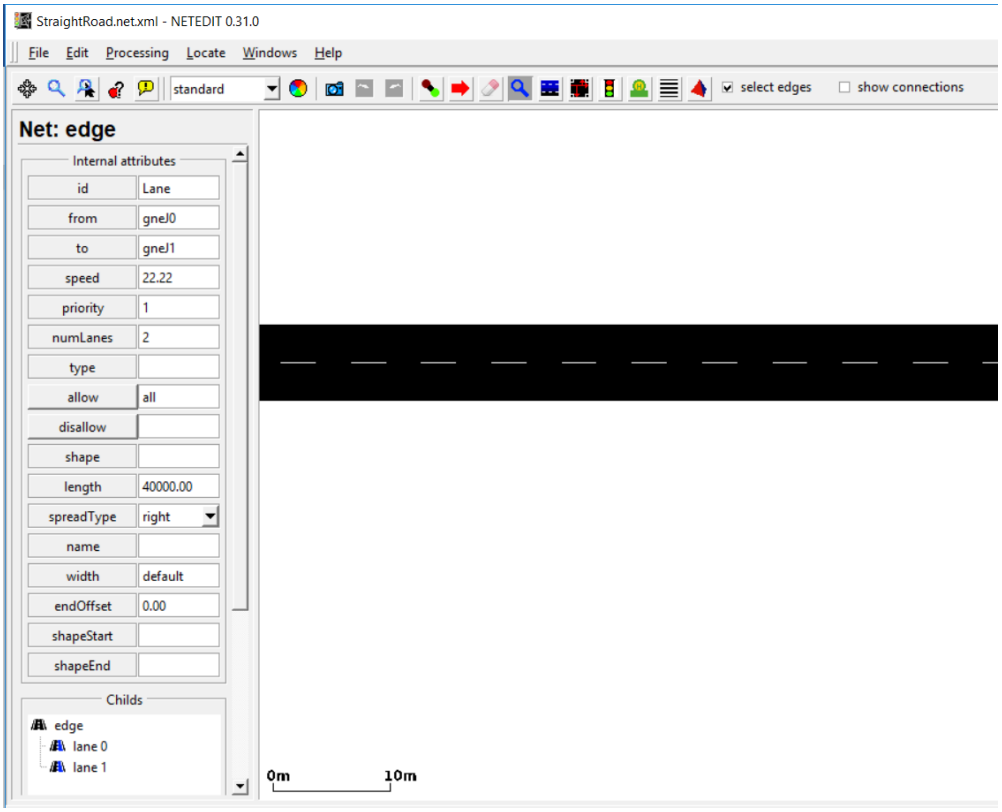


Figure 7 Road layout designed using NetEdit

'Shape' allows the user to specify the geometry of the edge [28]. The length specifies the distance covered by the edge in meters. In this research the length has been set such that the autonomous vehicle does not go past the end of the edge within the simulation time. Hence the length is set to be 40000 meters (40 km). As the positions of the junctions are (0,0) and (5000,0), each distance unit of the road layout corresponds to 8 meters. 'spreadType' is again related to the geometry of the edge. It describes how the lanes should be spread [29]. 'name' is the street name for this edge. 'width' determines the width of the lanes in meters. In this research the default width prescribed by SUMO is used. The 'endOffset' field determines if the edge should end before the actual endpoint, allowing the intersection with the other edge to be bigger [29]. 'startShape' and 'endShape' are used to specify the starting point and the ending point of the edges if 'shape' is used.

The designed road network is saved as .net.xml file so that it can be used by the simulation environment. In this research the road network is named as StraightRoad.net.xml and is shown in Appendix A: Configuration Files.

Design of vehicle routes

Before the simulation, the routes the vehicle should take must be specified. This is specified in a .rou.xml file. The snapshot of the route file used in the initial design is shown in Figure 8 and the complete file is given in Appendix A: Configuration Files.

```
<routes>
<vType id= "Auto" accel="30.000000" decel="30.000000" sigma="0.500000" maxSpeed = "55.550000" length="3.000000" color="0,0,255" vClass="passenger" minGap="0" tau="0.1"/>
<vType id= "Car" accel="30.000000" decel="30.000000" sigma="0.500000" maxSpeed = "11.100" length="3.000000" color="255,0,0" vClass="passenger" minGap="10" tau="0.1"/>
<vType id= "FastCar" lcKeepRight="100" accel="30.000000" decel="30.000000" sigma="0.500000" maxSpeed = "55.550000" length="3.000000" color="255,0,255" vClass="passenger" minGap="0" tau="0.1"/>
<route id="Straight" edges= "Lane"/>
<flow id="SlowCar" color="255,0,0"  begin="0" end= "200" probability="0.1" type="Car">
    <route edges="Lane"/>
</flow>
 <flow id="FastCar" color="255,0,255"  begin="0" end= "200" probability="0.01" type="FastCar">
    <route edges="Lane"/>
</flow>
<vehicle id="Auto" color="0,0,255" depart="60"  route="Straight" type="Auto"/>
</routes>
```

Figure 8 Route file for initial design

The route file should contain the characteristics of the vehicles used in the simulation and the route followed by it. In this research as the idea is to simulate vehicle overtaking scenarios on a highway, three types of passenger vehicles are used.

The vehicle with id as 'Auto' is the autonomous vehicle that is going to be controlled by the driving algorithm. The 'accel' and 'decel' are the maximum acceleration and deceleration capability of the vehicle. These value are used when SUMO controls the speed of the vehicles in the simulation. 'sigma' represents the value for the car following model denoting the driver imperfection [30]. Its default value is 0.5. Car following model prevents the vehicles from colliding with each other as the vehicle will automatically brake if it comes close to another vehicle. In this design, the car following model is disabled for the autonomous vehicle through TraCI which would be explained in a later section. Disabling the car following model will enable the autonomous vehicle to collide with the vehicle in front of it or the vehicle beside it during lane change. 'maxSpeed' is the maximum speed the vehicle can achieve. 'length' and 'colour' are for representation purposes during the simulation. 'length' specifies the vehicle length in meters and 'color' specifies the RGB (red, green, blue) code for the vehicle colour. The type of vehicle is determined by the 'vClass'. In this research all vehicles are from the passenger class. 'minGap' is the minimum gap a vehicle should maintain from the vehicle in front. A zero value of minimum gap means that the vehicle behind can be really close to the vehicle in front. 'tau' gives the minimum time headway between the rear of the vehicle in front and the front of the vehicle [30]. Minimum time headway is the difference between the time when the front of a vehicle arrives at a point on the highway and the time the front of the next vehicle arrives at the same point [31].

In a similar way the parameters of the other two type of vehicles are also set. The vehicle with id 'Car' is a slower moving vehicle with a maximum speed of 11.1m/s (40 Km/hr) and the other vehicle with id 'FastCar' is a faster moving vehicle with a maximum speed of 55.55m/s (200Km/hr). For the fast moving vehicle, the 'lcKeepRight' parameter is set to a high value. This ensures that the

vehicle will move back to the driving lane as soon as possible [32]. This is not required for the slower moving vehicle as it would mostly stay in the driving lane as it is the slowest moving vehicle in the simulation. The lane change mechanism for the autonomous vehicle is controlled by the driving algorithm. Hence the 'lcKeepRight' parameter needn't be used with the vehicle id 'Auto'. The car following model of both these types of vehicles are turned on so that no collisions would be triggered by them. The car following model used is the Krauß-model [33]. The reason for having a slower moving vehicle is so that the overtaking scenarios can be created easily. The velocities of the vehicles are taken to be quite low to help in debugging, as the movement of the vehicle in each time step would be less. Time step of 1 second is used for the simulation.

To randomize the traffic consisting of the manually driven vehicles, flows [34] are used. Flows allow the same type of vehicles to be to emitted into a simulation with a given probability each second [35]. The begin and end time determines within what time period should the emission of the vehicles be considered. The probability of emission of the slower moving vehicle is set to 0.1 and the faster moving vehicle is set to 0.01. Hence there would be enough vehicles for the autonomous vehicle to overtake and also there would be some vehicles which would overtake the autonomous vehicle as well. The route 'edges' state the edges the vehicle should take. As there is only one edge in this simulation, the id of that edge is given for the route.

There will be only one autonomous vehicle which will start at the 60^{th} second, so that by the time it starts there will already be some vehicles in the route.

The route file for this simulation is saved as StraightRoad.rou.xml.

SUMO Configuration

SUMO configuration file [36] is a .sumocfg file. This file combines the route file and the network file with the simulation settings for running the simulation. An example of the configuration file used is shown in Figure 9. The location paths of the route file and network file are mentioned in the input section. In this case all these files are in the same location. Hence just the file names needed to be specified. Other than the file paths, the action to be taken during a collision needs to be specified. The actions that can be performed in case of a collision are [37]:

- Teleport: The follower vehicle is teleported to the next edge in the route. This is the default setting.
- Warn: a warning is issued.
- None: no action is taken.
- Remove: The vehicles involved in the collision are removed from the simulation.

Remove is used as this will allow the learning algorithm in detecting the collision. The driving algorithm will monitor the vehicle information of the autonomous vehicle and if the vehicle information is missing before the end of the simulation then the driving algorithm can deduce that an accident has occurred.

```
<configuration>

    <input>
        <net-file value="StraightRoad.net.xml"/>
        <route-files value="StraightRoad.rou.xml"/>
        <collision.action value="remove"/>
    </input>

    <time>
        <begin value="0"/>
    </time>

    <report>
        <verbose value="true"/>
        <no-step-log value="true"/>
    </report>
    <gui_only>
        <quit-on-end value="true" />
        <start value="true" />
    </gui_only>
    <random_numberType>
        <random value="true" />
    </random_numberType>
    <processing>
        <lanechange.overtake-right value="false" />
    </processing>
</configuration>
```

Figure 9 Example of a SUMO configuration file

The begin value in the time section specifies when the simulation should begin [38]. In this case the simulation will begin as soon as it is invoked. The report section is used for specifying how the progress of the simulation should be reported. SUMO simulation can be invoked in two ways, with the GUI, where the user can visualise the simulation or through command line which puts less load on host's processing unit. Reports are important to know the progress of the simulation when it is run through the command line. 'gui_only' section specifies the settings while using the SUMO GUI for simulation. The 'start' value and the 'quite on end' value will start the simulation on loading of the configuration and quit the GUI after the simulation ends respectively.

In the earlier section it was explained that flows are used to generate the traffic randomly. For this the 'random value' needs to be set to true, else each simulation would be exactly identical, which is not useful for the driving algorithm to learn.

The 'lanechange.overtake-right' value specifies if overtaking through the right side is permitted. By default, it is set to false, hence it is not necessary to specify it in the configuration file.

The full set of the settings that can be used in the configuration file is given in [22]. The SUMO configuration file used in this research is shown in Appendix A: Configuration Files.

Another advantage of SUMO is that it can be enhanced with custom models and provides various APIs to remotely control the simulation. One such API is TraCI [39] which is explained in Chapter 4.4.

4.2 TensorFlow

TensorFlow [25] is an open source software library based on Python used for numerical computation using dataflow graphs. It contains wide range of functionalities for machine learning mainly related to deep neural networks. In this research the APIs from TensorFlow are used by Keras [24]. [12]

4.3 Keras

Keras [24] is a high level neural networks API capable of running on top of TensorFlow. It provides simple APIs for most of the machine learning tasks like learning and model creation and modification of neural networks. The list of APIs used from Keras for this research is given in Appendix B: External APIs used.

4.4 Interaction between the Simulator and the driving algorithm

Figure 10 describes the interaction between the driving algorithm developed in Python and the simulation environment in SUMO.

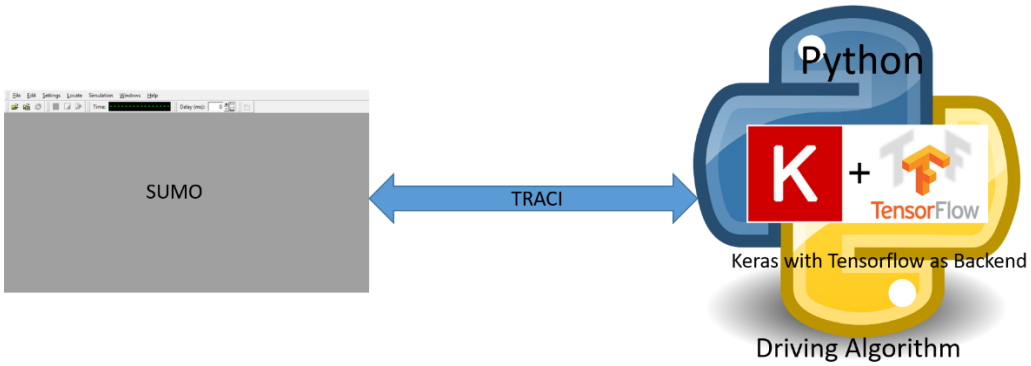


Figure 10 Block diagram describing the experiment setup reused from [12]

TraCI allows other applications to access the functionalities inside SUMO. These functionalities could involve modifying the simulation environment or the vehicle behaviour. In this research TraCI is used for controlling the autonomous vehicle by the driving algorithm. The APIs used from TraCI is given in Appendix B: External APIs used.

5. Approach for defining the elements of reinforcement learning

As discussed in Chapter 3, reinforcement learning using Deep Q learning is used for developing the driving agent. For using reinforcement learning, input features, output actions and the reward mechanism need to be established. This chapter explains how these are selected and also discusses the evaluation criteria for the performance of the driving algorithm in a 2 lane highway environment.

5.1 Selection of input features of the input state

In a two lane highway environment, vehicles can travel in either the first lane also called as the driving lane or the second lane, which is called as the overtaking lane. Hence at any point in time, there could be a vehicle in front or behind the vehicle being controlled and also in front and behind the autonomous vehicle in the other lane as shown in Figure 11. Here d_1, d_2, d_3 and d_4 are the distances of the immediate surrounding vehicles of the autonomous car (maroon coloured car). The distances are taken from the centre of the front bumper of the vehicles [40]. Only the distances with the immediate surrounding vehicles are taken as this information is enough for the autonomous vehicle to decide what action to take. If the distance with the vehicle in front is decreasing, then the autonomous vehicle could change to the overtaking lane provided that lane is free which can be found from the value of d_3 and d_4 or it can reduce its speed. In the same way if d_2 is decreasing, that is the vehicle behind is approaching the autonomous vehicle, then if the path in front is clear and the vehicle is under the speed limit, it can speed up.

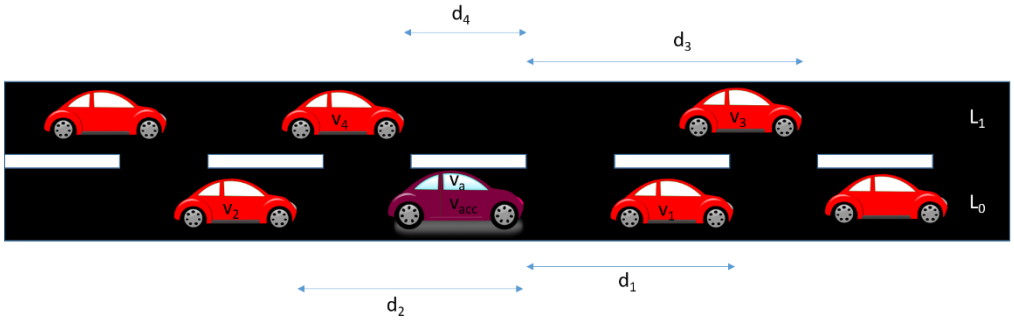


Figure 11 Positioning of vehicles in a two lane road

The information of the velocities of the surrounding vehicles are also useful. In Figure 11, these are labelled as v_1, v_2, v_3 and v_4. The velocity of the autonomous vehicle is labelled as v_a. Knowledge of the current velocity of the vehicles will allow the driving algorithm to make better decisions. For example, if the vehicle in front is close by, i.e. d_1 is small, but v_1 is greater than v_a, then this means that the vehicle in front has started going faster than the autonomous vehicle. Hence there is no

point for the autonomous vehicle to move to the overtaking lane. Also, as controlling the vehicle would also involve accelerating or decelerating it when required, the current acceleration rate of the autonomous vehicle (v_{acc}) is also required as an input to the driving agent. Another important input state is the current lane the autonomous vehicle is occupying. In different lanes the vehicle has to behave differently. The vehicle should only try to go to the overtaking lane which is shown as L_1 in Figure 11 to overtake and then should come back to the driving lane represented by L_0 in Figure 11. Hence the lane position is also vital for the driving algorithm. As the length of all the vehicles are the same as shown in Chapter 4.1, it does not influence the measurement of the distance between the vehicles. Hence the input features for this research was fixed to be:

1) Current velocity of the autonomous vehicle (v_a).
2) Current velocity of the vehicle in front in the driving lane (v_1).
3) Distance the autonomous vehicle is from the vehicle in front in the driving lane (d_1).
4) Current velocity of the vehicle behind in the driving lane (v_2).
5) Distance the autonomous vehicle is from the vehicle behind in the driving lane (d_2).
6) Current velocity of the vehicle in front in the overtaking lane (v_3).
7) Distance the autonomous vehicle is from the vehicle in front in the overtaking lane (d_3).
8) Current velocity of the vehicle behind in the overtaking lane (v_4).
9) Distance the autonomous vehicle is from the vehicle behind in the overtaking lane (d_4).
10) Current lane the autonomous vehicle is occupying (L=0 or L=1).
11) Current Acceleration rate of the autonomous vehicle (v_{acc}).

If there are no vehicles in the corresponding position, for example if there is no vehicle in front in the driving lane, then it can be considered as there is a vehicle far away in front, which would not interfere with the driving algorithm. Hence in such a condition the corresponding distance is given a high value and the velocity is given as 0. Hence if there is no vehicle in front in the driving lane, then d_1 will be a high value and v_1 will be 0. The values that are given and its units will be discussed in detail during design in Chapter 6.

The possible input states for this driving agent will include the features {v_a, v_1, d_1, v_2, d_2, v_3, d_3, v_4, d_4, L, v_{acc}}.

5.2 Selection of Output actions

In a highway environment, the basic actions a vehicle can do are to change lanes or to change its velocity. Hence based on these the actions states would be:

1) Move to overtaking lane.
2) Move to driving lane.
3) Increase velocity of the autonomous vehicle.
4) Decrease velocity of the autonomous vehicle.

As there are only two lanes in this environment, if the autonomous vehicle is already in the driving lane, then action 2 will have no effect on the state of the vehicle. In the same way if the autonomous vehicle is in the overtaking lane, then action 1 will not have any effect on the state of the vehicle. Hence an idle action for the autonomous vehicle to stay in the same lane is not used. Actions 3 and 4 were used in the initial implementations, where the vehicle maximum speed were quite low (around 10 m/s). If action 3 was selected, then the velocity of the autonomous vehicle would be increased by a small constant. In the same way, if action 4 was selected, the velocity of the autonomous vehicle would be decreased by a small constant. The problem with this approach is that, though it works well with vehicles with low max speed, when the maximum speed is a large value, around 55 m/s, then the time required to get the velocity to this value would be quite high. Hence in later iteration, actions 3 and 4 were modified to acceleration and deceleration. In this vehicle will either accelerate or decelerate with a constant acceleration or deceleration rate. This constant will vary based on the next immediate action. If there are continuous actions for acceleration, then the acceleration rate will also increase and in the same way, if there are continuous actions for deceleration, then the deceleration rate will increase. The principle used for changing the acceleration rate and the deceleration rate is explained in the Actions section of Chapter 6.

5.3 Reward System

The reward system is really important in a reinforcement based learning algorithm. A good reward system will enable the learning algorithm to learn faster and produce favourable outputs as explained in Chapter 2.1.

In this research, the reward system was altered from time to time to get the best performance for the driving algorithm.

Two types of reward systems were mainly considered. One was a lower level reward system where the reward was given for almost all the possible conditions. Hence the vehicle was almost directed to do as expected. An overview of this reward system is given in Table 3. The main aim for having the reward structure as shown in Table 3 is to have total control in the way the autonomous vehicle moves.

Motivation for Lower Level Reward System

Avoiding collisions is the most important part of the driving agent. Hence no actions taken by the agent should lead to an accident. Therefore, the maximum negative reward is given if the action taken leads to a collision. This reward is denoted by $R_{maxNegetive}$.

The autonomous vehicle should only move to the overtaking lane if there is a vehicle moving nearby at front in the driving lane. The distance the autonomous vehicle needs to consider before moving to the overtaking lane is given by 'd'. Hence if there is a vehicle at a distance of d in front of the autonomous vehicle and if it is moving slower than the autonomous vehicle, then it means that the autonomous vehicle should try to overtake it if the overtaking lane is free or if the overtaking lane is not free then it should try to reduce its speed so that it does not come very close to the vehicle in front. This is depicted in the conditions in row 2, 6, 7 and 8 of Table 3. In row 2, $L_a = L_0$ to L_1 means that the autonomous vehicle is changing to the overtaking lane. Hence if this is done because the vehicle in front in the driving lane is nearer and slower than the autonomous vehicle and also while changing the overtaking lane is free, i.e. there are no vehicles in the proximity of the autonomous vehicle in the overtaking lane, then a positive reward is given as represented by $R_{positive}$. Similarly, in Row 6, for the same situation if the autonomous vehicle does not move to the overtaking lane, then a negative reward is given as this might lead to a collision. The reward given is not a constant but a variable which depends on the distance with the vehicle in front in the driving lane. If the vehicle in front is very near, then a high negative reward is given and if it is not that near then a lower negative reward is given. Row 7 and 8 depicts when a positive reward is given in such a condition. In row 7, even though there is a vehicle nearby in the front, its velocity is higher than the velocity of the autonomous vehicle. Hence there is no way the autonomous vehicle will catch up to the vehicle in front. Hence it is the right decision to stay in the driving lane itself. Therefore, a positive reward is given. In row 8, the possibility of the overtaking lane not being free in such a situation is considered. Hence then the only action the driving agent can do is to reduce its velocity. Hence if the velocity of the autonomous vehicle is lower than its velocity at the previous time step ($v_{a,t-1}$), then a positive reward is given.

S.No	Reward Value	Reward Condition
1	$R_{maxNegative}$	Collision occurrence
2	$R_{positive}$	Else & $L_a = L_0$ to L_1 & $d_1 < d$ & $v_1 < v_a$ & L_{1free}
3	$R_{negative}$	Else & $L_a = L_0$ to L_1 & $d_1 < d$ & $v_1 < v_a$ & L_{1busy}
4	$R_{negativeVariable}$	Else & $L_a = L_0$ to L_1 & $v_1 > va$
5	$R_{negativeVariable}$	Else & $L_a = L_0$ to L_1 & $d_1 > d$
6	$R_{negativeVariable}$	Else & $L_a = L_0$ & $d_1 < d$ & L_{1free} & $v_a > v_1$
7	$R_{positive}$	Else & $L_a = L_0$ & $d_1 < d$ & L_{1free} & $v_a < v_1$
8	$R_{positive}$	Else & $L_a = L_0$ & $d_1 < d$ & L_{1busy} and $v_a < v_{a,t-1}$
9	$R_{negativeVariable}$	Else & $L_a = L_0$ & $d_1 > d$ & ($v_a <$ speed limit | $v_a >$ speed limit)
10	$R_{postive}$	Else & $L_a = L_0$ & $d_1 > d$ & $v_a =$ speed limit
11	$R_{negativeVariable}$	Else & $L_a = L_1$ & $d_3 < d$ & $v_3 < v_a$
12	$R_{negativeVariable}$	Else & $L_a = L_1$ & $d_1 > d$ & $d_2 > d$
13	$R_{positive}$	Else & $L_a = L_1$ & $v_a > v_{a,t-1}$ & $d_3 > d$
14	Zero Reward	Else

Table 3 Lower level Reward system

If the autonomous vehicle moves to the overtaking lane unnecessarily, then a negative reward needs to be given as that is an undesired action. Hence rewards for such scenarios are given in rows 3, 4 and 5 of Table 3. Row 3 considers the scenario, where the autonomous vehicle moves to the overtaking lane if the vehicle in front is slower and nearby, but the overtaking lane is not free. If the vehicle moves to the overtaking lane when there is already a vehicle present nearby in the overtaking lane, then there is a high possibility for the autonomous vehicle to collide with the vehicle in the overtaking lane. Hence a negative reward is given in such a case, so that the autonomous vehicle will only move to the overtaking lane when it is free. Row 4 and Row 5 depicts a scenario where the autonomous vehicle change to the overtaking lane when either the vehicle in front in the driving lane is slower or it is far away ($> d$). In both these cases there is no need to move to the overtaking lane as there is no possibility for the autonomous vehicle to be near the vehicle in front and hence it could have just continued with the same velocity in

the driving lane. Therefore, a variable negative reward ($R_{NegativeVariable}$) is given which depends on the distance between the autonomous vehicle and the vehicle in front in the driving lane. If the distance is high, this means that the autonomous vehicle changed lanes very early and hence a higher negative reward is given. If the distance is low but it is still greater than d, then a lower negative reward is given.

Another aim of the autonomous vehicle is to go at a good speed so as to decrease the journey time provided that it follows the speed limit. This is given by the reward scheme in row 9 and 10 of Table 3. In row 10 a positive reward is given if the vehicle maintains the speed limit. The reason for checking if the vehicle in front is far away is only then can the autonomous vehicle move in its desired speed. If there is a vehicle nearby then that would influence the speed of the autonomous vehicle as the vehicle may have to slow down which is again not an undesirable action for that situation. Hence this is only monitored if the vehicle in front is far away. A variable negative reward is given if the vehicle is either slower than the speed limit or exceeds the speed limit. The negative rewards are based on how further the vehicle's velocity is when compared with the speed limit. If the velocity is closer to the speed limit then the negative reward value is lower and it increases with increase in the difference between the vehicle's velocity and the speed limit. This is given in row 10.

The autonomous vehicle should try to be in the driving lane as much as possible and the overtaking lane should only be used for overtaking. Therefore, after overtaking the vehicle should come back to the driving lane. This is taken care by the rewarding scheme given in row 12 of Table 3. If the autonomous vehicle remains unnecessarily in the overtaking lane. That is if there is no vehicle in front in the driving lane for a long distance (d_1>d) to overtake and there is no vehicle behind in the driving lane for a long distance(d2>d). Hence there is no possibility of collision while moving back to the driving lane. Then a negative reward is given if the vehicle continues in the overtaking lane. This reward is based on the distances of the vehicles ahead and behind the autonomous vehicle in the driving lane. Hence if these distances are high, then the negative reward will be high and if it is low then the negative reward will also be low.

There is also a possibility for the autonomous vehicle to encounter a vehicle in front in the overtaking lane while overtaking. In such a case the autonomous vehicle cannot again change to a higher lane as there are no other lanes available. Hence in such a case a negative reward is given if the velocity of the autonomous vehicle is greater than the velocity of the vehicle ahead. This negative reward is again based on the distance between the two vehicles. If the distance is less the negative reward is higher and if the distance is high the negative reward is lower. The condition for giving this reward is shown in row 11 of Table 3. Also the idea is that the overtaking should be done as quickly as possible and that the vehicle should not slow down unnecessarily in the overtaking lane. Hence if the vehicle increases

its speed while in the overtaking lane as shown in row 13, then a positive reward is given.

As can be seen, in this method of assigning reward, rewards are assigned for most of the scenarios. This makes the reward system quite complex as there are quite a lot of scenarios possible. If a new scenario were to occur, it could make the driving algorithm vulnerable to errors. Another problem with this reward system is that, it is very specific to the environment, it would be difficult to generalize the reward system for using it in another highway environment like highways with more than two lanes which is one of the goals of this research.

To tackle this problem a more generalised reward system is used which does not completely control how the vehicle would behave, but mainly takes into account that the vehicle follows the basic road and safety rules. This reward system is shown in Table 4.

Motivation for Generic Reward System

S.No	Reward Value	Reward Condition
1	$R_{Collision}$	Collision occurrence
2	$R_{Standstill}$	Else & $v_a = 0$
3	R_{near}	Else & $L_a = L_0$ & $d_1 < d$
4	$R_{LaneChange}$	Else & $L_a = L_1$ & $d_1<d$ & $v_a > v_{a,t-1}$
5	R_{Lane1}	Else & $L_a = L_1$ & $d_1 > d$
6	$R_{Lane1ApproachVehicle}$	Else & $L_a = L_1$ & $d_3 < d$ & $v_a < v_{a,t-1}$
7	$-R_{Lane1ApproachVehicle}$	Else & $L_a = L_1$ & $d_3 < d$ & $v_a >= v_{a,t-1}$
8	$R_{OverSpeeding}$	Else & $v_a >$ speed limit
9	$R_{Speding}$	Else & $v_a > v_{a,t-1}$
10	$R_{MaintainSpeedLimit}$	Else & $v_a =$ speed limit
11	Zero Reward	Else

Table 4 Generic Reward system

In Table 4, rewards are mainly given for undesirable traffic situations. Again for collision a very high negative reward is given as it is still the most undesirable outcome. This is depicted by $R_{Collision}$.

A large negative reward is also given if the vehicle stops on the highway as this can cause a traffic jam. In this research, it is assumed that all the other vehicles will follow the traffic rules as they are being driven manually. Hence there is no

possibility of a collision with the autonomous vehicle, from the vehicle behind. Hence if the autonomous vehicle stops, then the vehicle behind will have to either stop as well or move to the overtaking lane and overtake. In any case this situation will disrupt the flow of traffic. Hence a large negative reward $R_{Standstill}$ is given when the vehicle stops. This is depicted in the second row of Table 4.

To discourage the vehicle from coming close to the vehicle in front on the driving lane($d1<d$) a negative reward R_{near} is given to avoid chances of the vehicles colliding. This is represented in row 3.

To encourage the autonomous vehicle to go to the overtaking lane when there is a vehicle in front at a close distance, a small positive reward $R_{LaneChange}$ is given on changing to the overtaking lane as depicted in row 4 of Table 4. Here it is not checked if the overtaking lane is free because, if it is not free and a collision does happen then anyway the $R_{collision}$ will be given and not the $R_{LaneChange}$.

To avoid the vehicle from staying unnecessarily in the overtaking lane a negative reward R_{Lane1} is given if there is no vehicle ahead in the vicinity in the driving lane for the autonomous vehicle to overtake. This is given in row 5 of Table 4.

As this implementation is for a 2 lane highway, if the autonomous vehicle is in the overtaking lane and there is a vehicle in front of it in the overtaking lane, it needs to decrease its speed to avoid a collision with that vehicle. Hence in such a scenario, if the autonomous vehicle decreases its speed from what it was in the previous time step, a positive reward $R_{Lane1ApproachVehicle}$ is given as shown in row 6 and if it does not decrease its speed then $-R_{Lane1ApproachVehicle}$ is given as the reward as shown in row 7 of Table 4.

Rows 8, 9 and 10 are the rewarding schemes for the vehicle to obey the speed limit. An efficient driving agent should try to reduce the travel time. This can be done by travelling at high speed. For safety reasons the vehicle needs to maintain the speed limit. Hence travelling at v_a=speed limit would be the fastest way to reach the destination. Hence a high reward $R_{MaintainSpeedLimit}$ is given when the speed of the vehicle is the same as the speed limit as shown in row 10. It is not always possible to travel at the speed limit, hence to encourage the driving agent to accelerate until it reaches the speed limit, a positive reward $R_{Speding}$ lower than $R_{MaintainSpeedLimit}$ is given as depicted in row 9. Again to avoid going over the speed limit, a higher negative reward $R_{OverSpeeding}$ is given as shown in row 8 of Table 4. For this reward scheme there is no need to check for the distance with the vehicle in front as if there is a vehicle in front nearby automatically the reward condition in row 3 is invoked and these reward conditions are not considered.

The advantage of this reward system is that it is a lot more generic than the previous reward system and it only dictates on how the vehicle should behave to avoid potential collisions or traffic disruptions. This reward system can also be used for environments with more than one lane with a small change in conditions from

row 3 to row 7 to handle multi lanes. The reward system for multilane highway is discussed in Chapter 7.3.

5.4 Performance Evaluation of the Driving Algorithm

Efficiency of the driving algorithm depends on how well it is able to manoeuvre the autonomous vehicle through traffic. The performance of the algorithm can be evaluated based on number of times the autonomous vehicle is responsible for an accident. This can be calculated by using a feature in SUMO, where the vehicles involved in a collision are removed from the simulation. This is discussed in the section SUMO Configuration in Chapter 4.1. During the simulation the number of collisions can be plotted over the number of simulations to get the collision ratio. The lesser the collisions, more efficient the driving algorithm is. The problem of using just the collision ratio as an evaluation for the performance is that a vehicle can avoid collision by going really slow. As all other vehicle are assumed to be careful, they will not initiate a collision. Hence even though the collision ratio is low, the driving algorithm may not be behaving properly as the vehicle does not go for overtaking of the slower moving vehicles. Hence another evaluation based on the distance travelled by the vehicle in the simulation is also used as a criterion for evaluating the performance of the driving algorithm. The total distance travelled by the autonomous vehicle in each episode can be plotted. From this the average distance travelled by the autonomous vehicle in each simulation can be calculated. As the simulation time is known, the average speed of the vehicle can be calculated and if this speed is greater than the maximum speed of the slower moving vehicles, then it means that the autonomous vehicle has done some overtaking. As this is used to check if the driving algorithm is performing well in case of no collision, The distance total distance travelled is plotted only for simulations that end without a collision. The total rewards achieved in each simulation is also used as a criterion to check if the driving algorithm is trained correctly. If the vehicle finishes the simulation successfully by following all the rules, then the total reward achieved for that simulation would be higher than when a collision occurs or if the vehicle breaks any rules, for example if it remains in the overtaking lane unnecessarily.

6. Design of the Driving Algorithm for driving in a 2 Lane Highway

Based on the analysis in Chapter 5, a driving algorithm was designed and simulated for controlling a vehicle in a 2 lane highway. This chapter talks about the design of the driving algorithm and results from the simulation of this driving algorithm. It also talks about changes in the reinforcement learning parameters affecting the performance of the driving algorithm.

This chapter is mainly divided into two parts. Design of the Driving algorithm in Python and the results obtained during testing this driving algorithm in the simulation setup described in Chapter 4.1. These results were used to improve the driving algorithm such that it can be used in a generic multilane highway environment.

6.1 Design of The Driving Algorithm

The driving algorithm is responsible of learning based on the actions performed by the autonomous vehicle. It also invokes the communication with SUMO to start and proceed with the simulation. As explained in Chapter 4, Keras is used for designing and implementing the Neural network for the driving algorithm.

For the driving algorithm to communicate with the simulation environment, a new environment based on Gym [41] is used. Gym is a toolkit used for developing the environment used in machine learning. It provides the structure required in the environment that can be used by the learning algorithm [12]. The main structure of the Gym environment is [42]:

1) Initialization state: The environment is initialized. The input states and the actions are initialised.
2) Reset state: This state sets up the environment in sumo and starts the simulation. This state is maintained until the autonomous vehicle has become active in the simulation. The learning algorithm only starts after this state.
3) Step state: Each time an action is taken by the driving algorithm; it leads to a new step. The step state performs the action desired by the driving algorithm, calculates the rewards and gets the new input state.

Figure 12 shows the class diagram of the driving algorithm.

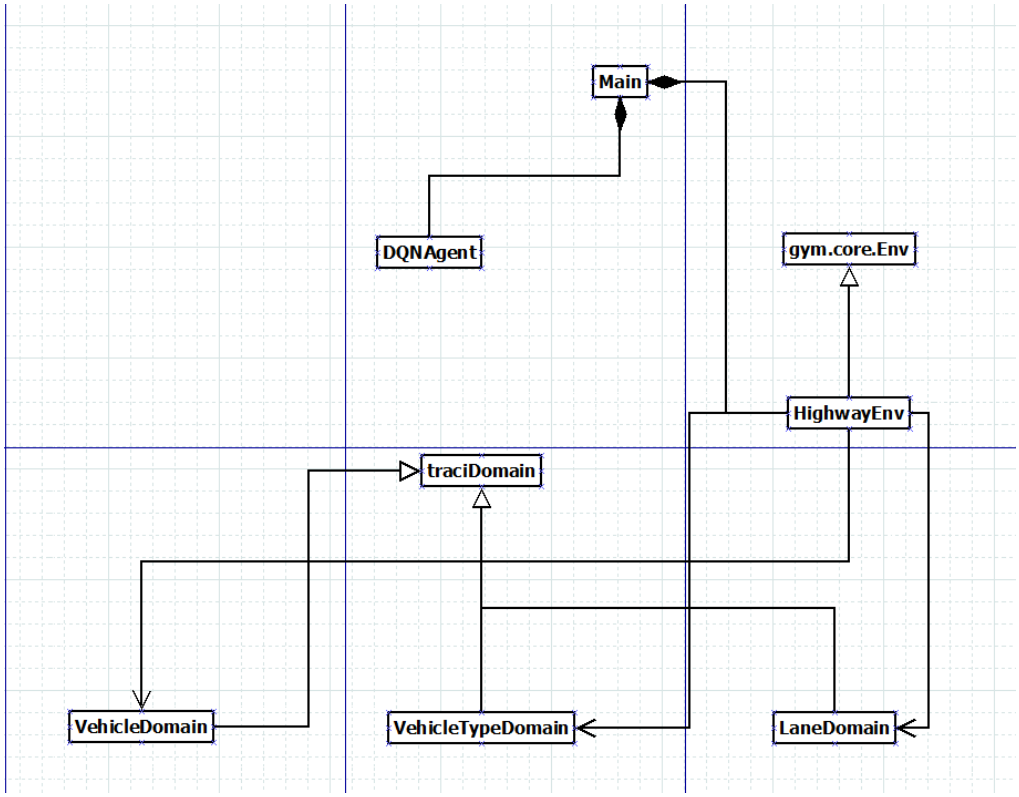


Figure 12 Class Diagram of the Driving algorithm

The design contains three user defined classes, the 'Main' class, 'DQNAgent' class and the 'HighwayEnv' class as shown in Figure 12. The 'Main' class is invoked to start the training process. It uses the 'DQNAgent' class for learning and the 'HighwayEnv' class to realise the results of the driving algorithm and to calculate the rewards. Here the 'DQNAgent' class is the equivalent of agent and the 'HighwayEnv' class is the equivalent of environment in Figure 6 in Chapter 3.1.

The weights of the neural network need to be first trained, as at the start they are all assigned to random values. Hence the training period is used to train the neural network. The resulting neural network model is then saved to be used for testing on new simulations. The training needs to be long enough such that the neural network can learn to the maximum. From simulations it was found that training the algorithm for 7000 episodes was enough as training longer than this did not have a big impact on the performance. Figure 13 shows the collision percentage of the autonomous vehicle with increase in the training episodes.

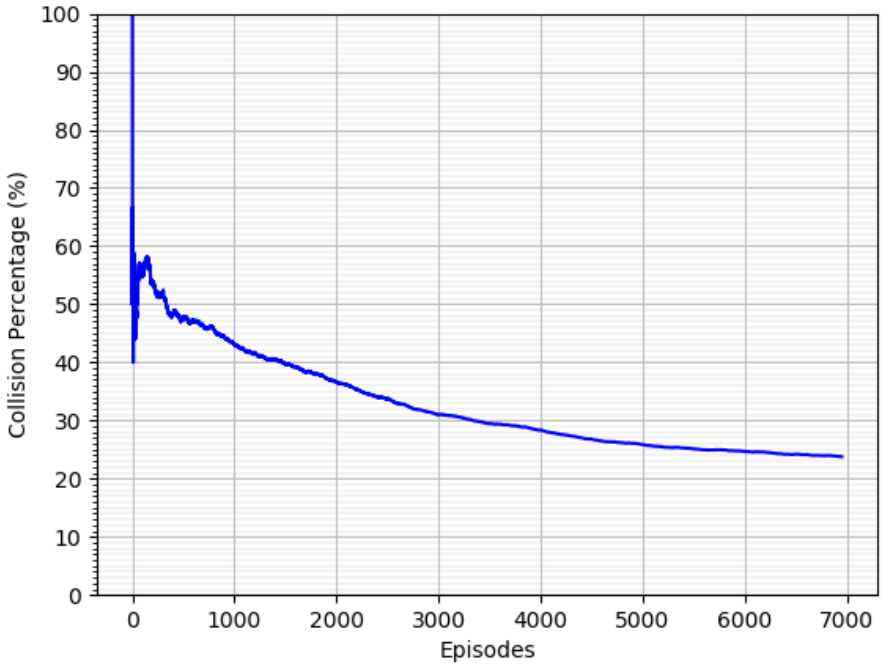


Figure 13 Percentage of collisions of the autonomous vehicles during training

Episode is an entire simulation from when the autonomous vehicle is inserted to when the simulation ends due to either the end of simulation time or due to a collision as explained in Chapter 2.1. It can be seen from the figure that the plot almost converges at around 7000 episodes. For the simulations done in this research, the simulation time is kept to be 100 steps with each step being 1 second long. The driving agent is trained during each time step.

'DQNAgent' class in Figure 12 creates and trains the neural network based on the results from the simulation which is got from the 'HighwayEnv' class and passed on by the 'Main' class. It also contains a memory buffer queue of length 2000 that stores the results from each time step. This means that if an entire episode is completed without a collision, the memory buffer can hold information of 20 full episodes. This information include, the episode index, current input state, action taken, reward achieved, next input state and information about whether the episode ended or not which are required for the training the driving agent using experience replay as explained in Chapter 2.1. A batch of 32 random items from the memory buffer is taken and the neural network is trained again with this data using the algorithm already explained in Algorithm 1. As this is used for training the neural network it is only done during the training period and not during the testing period.

As mentioned in Chapter 2, the actions by the driving algorithm are first taken at random and as the algorithm learns it is taken based on the maximum output

form the neural network. The convergence of the exploration rate is exponential. At the start of the training the exploration rate (ε) is set to 0.9 so that the driving agent can learn the consequence of all the actions. This is then decreased exponentially by a factor of 0.9992 after each episode until it reaches the minimum exploration rate of 0.1. After this the actions are mostly taken purely based on the outputs of the neural network. In this implementation It takes around 2747 episodes for the exploration rate to decrease to 0.1. Hence until then the actions are taken at random from time to time.

The learning rate (α) is chosen to be 0.001 which is a very low value, so that the Q function will have a higher chance of converging to its optima and not go past it as discussed in Chapter 2.1. The discount factor (γ) is chosen to be 0.9 so that the algorithm tries to maximise the long term rewards [12] according to Equation (1) in Chapter 2.1.

Sequential neural network is used for reinforcement learning as explained in Chapter 2.1. In this layers are stacked in front of each other for propagation of data through them [43]. Keras provides the APIs required for creating these models [44]. The first and the last layer of the neural network are the input and the output layer respectively. The layers in between these are called the hidden layers. Each layer contains certain number of nodes. The input layer contains the nodes equal to the number of input features, which in this case is 10 and the output layer contains the number of nodes equal to number of actions which is 4. The nodes in the hidden layer can be of any amount. In this research varying hidden layer configurations from 1 hidden layer to 5 hidden layers with varying nodes between 1 to 2001 were used to check the performance of the learning algorithm. Unfortunately, there is no direct relation between the number of nodes and hidden layers and the performance of the driving algorithm. The nodes in the neural network are activated using an activation function. This activation function is used to get the output from the nodes. The action corresponding to the max output among the nodes from the output layer is chosen as the action of the learning algorithm. The most common activation function used is the Rectified Linear Unit (ReLU) activation function. Here the output is 0 if the input is less than 0 or it is the input if the input is greater than or equal to 0 [45].

'HighwayEnv' class in Figure 12 takes care of incorporating the actions given by the driving algorithm into the simulation in SUMO and returns the new input state and the reward for the action back to the Main class. It inherits the properties of the 'Gym.Core.Env' as shown in Figure 12 and hence implements the definitions for initialization, reset and step. It is also associated with the classes in TraCI for communication with the SUMO environment. The classes associated with 'HighwayEnv' are 'VehicleDomain', 'VehicleTypeDomain' and 'LaneDomain' as shown in Figure 12. 'VehicleDomain' class contains the APIs for accessing and modifying the current vehicle parameters of the vehicles that are in the simulation. These

parameters include, the current velocity, lane number, position and other vehicle parameters [46]. It also contains the APIs for activating and deactivating the car following model that activates collision avoidance. TraCI supports object variable subscription [47]. Using this the vehicle parameters of the vehicle can be subscribed at the start of the simulation and the response of the subscription contains the vehicle parameters of the previous time step. This is used for getting the vehicle parameters which are the position, vehicle speed, lane index and distance travelled of all the vehicles that are currently active in the simulation.

The APIs from the other two classes of 'VehicleTypeDomain' [48] and 'LaneDomain' [49] are used to access the maximum speed of the vehicle and the speed limit in the lane respectively.

The entire list of APIs used in the implementation is given in Appendix B: External APIs used.

Finding the position of the surrounding vehicles

As described in Selection of input features in Chapter 5.1, the driving agent only uses the distance between the autonomous vehicle and the immediate surrounding vehicles and their velocities for learning. TraCI does not contain any API for giving this information directly. Hence the environment in the driving algorithm which is the 'HighwayEnv' class in Figure 12, needs to segregate this information from the received information of all the vehicles in simulation. This section describes how the vehicle information of the immediate surrounding vehicles to the autonomous vehicle is found.

Firstly, the positions and the vehicle ids and the lane number of all the vehicles currently active in the simulation are accessed. The vehicle position returns the x axis and the y axis position. As the highway used here is a horizontal road as shown in Figure 7, only the x axis value is required for knowing the position of the vehicle. This list of information with the vehicle id, x axis position and the lane number are sorted with respect to the x axis position. The corresponding index of the list containing the autonomous vehicle's Id is its positon on the road. The next index with the lane number as 0 is the vehicle in front in L_0, next index with lane number 1 is the vehicle in front in L_1, previous index with lane number 0 is the vehicle behind in L_0 and the previous index with lane number 1 is the vehicle behind in L_1. If either the last index in the list or the first index in the list is reached without finding a vehicle, then it means that there are no vehicles in front or behind the autonomous vehicle in that lane respectively. The value for that distance is given as the maximum distance value for the input state and the velocities for those input states is given as 0 m/s. From the positions of the surrounding vehicles, the x axis distances d_1, d_2, d_3 and d_4 can be found. Using the vehicle Ids their velocities v_1, v_2, v_3 and v_4 can also be found. Also if the distances d_1, d_2, d_3 or d_4 are greater than

the max distance for the input states, then it means that the vehicle is not in the vicinity of the communication range and again the max distance of the input state is given as the distance for that vehicle and its velocity is taken as 0 m/s.

Actions of the driving algorithm

As mentioned in Chapter 5.2, four types of action were chosen. As there are just 2 lanes in this scenario, the vehicle can either move to the overtaking lane or to the driving lane. The other two actions were to increase or decrease the vehicle velocities. The problem with increasing or decreasing the vehicle velocity is that it will take the vehicle some time to reach a high velocity. For example, if the vehicle can only increase by a constant speed of 0.5 m/s, then to reach the speed limit of 22.5m/s it would take around 45 seconds of continuous same action of increasing the vehicle velocity. As a result the vehicle's average velocity was quite low as shown by one of the average distance travelled plots in Figure 14.

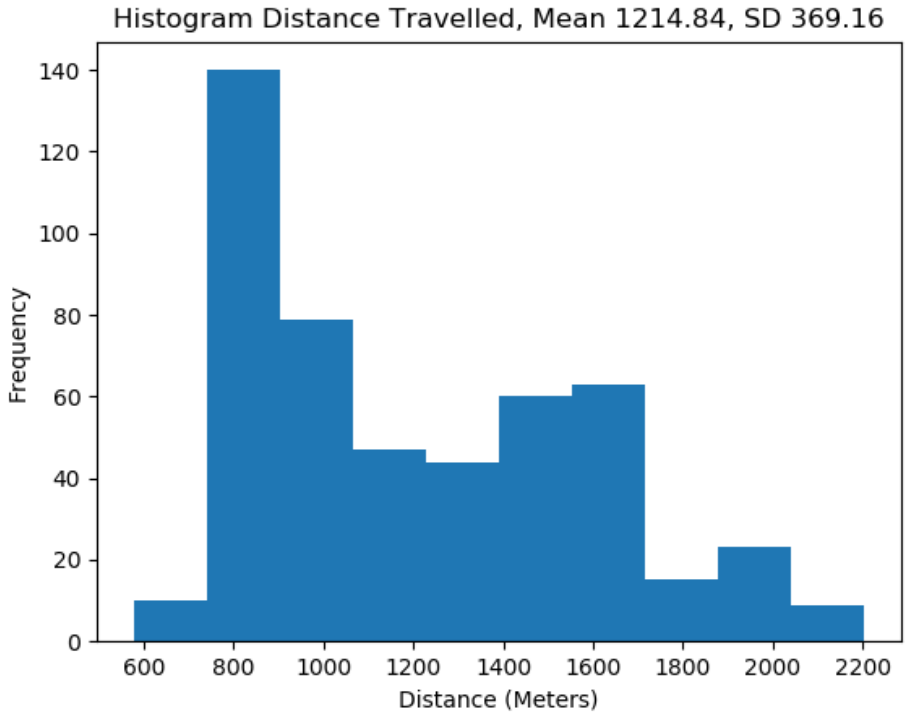


Figure 14 Frequency distribution of distance travelled in a simulation with 500 episodes

As each episode was for 100 seconds the average [50] velocity of the autonomous vehicle was found to be 12.14m/s.

Another method was then used for increasing or decreasing the vehicle velocity. This was in line with what usually happens while driving, which is increasing speed based on the acceleration or deceleration rate. At the start a constant acceleration and deceleration rate is set at 1.26 m/s^2 and 0.63 m/s^2 respectively. This rate is

doubled on immediate actions of acceleration or deceleration. That is, the first time acceleration is selected as the action, the acceleration rate of the vehicle will be 1.26 m/s^2. If the next action is also acceleration, then the new acceleration rate will be $1.26 \times 2 = 2.52 m/s^2$ and if the next action is also acceleration then it will be $1.26 \times 3 = 3.78 m/s^2$. If the next action is deceleration, then the deceleration rate will be taken into account to slow the vehicle and the acceleration rate will again go back to the stating value of 1.26m/s^2.

To calculate the time, the vehicle should accelerate, the time required for the vehicle to reach its maximum speed with the current acceleration rate is calculated. The vehicle will then try to increase to its maximum speed in the calculated time, provided that it does not get an action to decelerate within that time period.

In the same way to change the velocity based on declaration, time taken for the vehicle to come to rest is calculated with the set deceleration rate. The vehicle will then reduce its speed gradually to 0m/s^2 within the given time provided that an action for acceleration does not occur during that time.

Another advantage of this method is that here the vehicle does not directly go to a certain velocity but the velocity increases gradually as it happens in a real vehicle. The distance plot of the autonomous vehicle using this method in the same environment and traffic rate is shown in Figure 15.

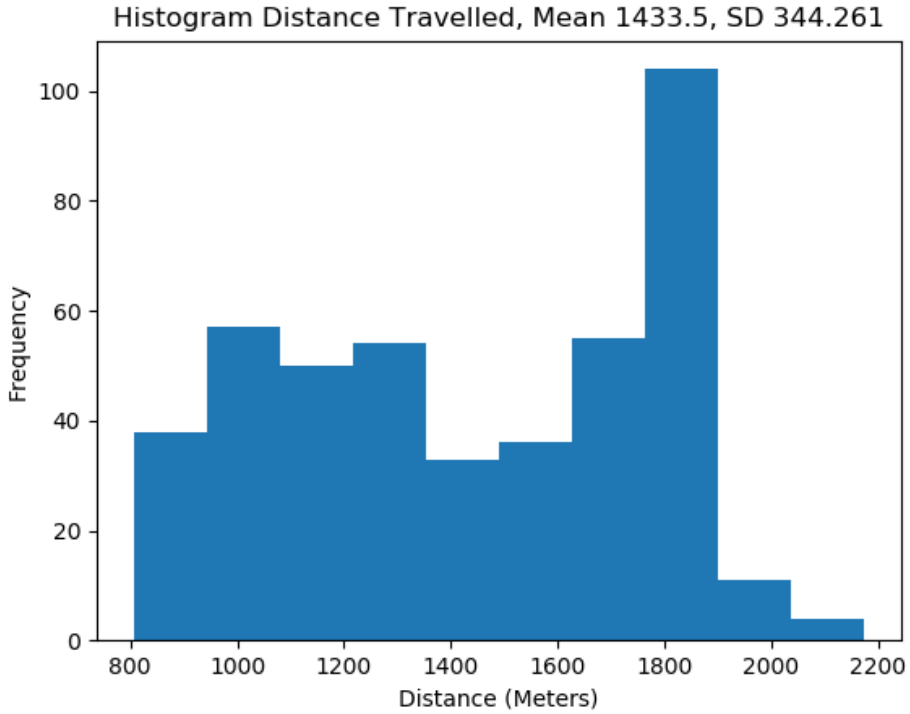


Figure 15 Frequency distribution of distance travelled in a simulation with 500 episodes

The mean velocity of the autonomous vehicle using acceleration and deceleration as actions were found to be 14.33m/s, which is higher than the velocity found in Figure 14.

Rewarding Scheme

Generic reward system explained in Chapter 5.3 in Table 4 is used for calculating the rewards. In this section values are given for the parameters described in Table 4. The values for the rewards were found by testing the driving algorithm with different combinations of reward values and observing its performance as described in Chapter 3.3. The value combinations which gave a low collision rate as well as a good average speed for the autonomous vehicle was selected.

$R_{collision}$, which is reward for collision in Table 4 of Chapter 5.3 is given a value of -101. This is the highest individual negative value as collision is the most undesired outcome for the driving agent. $R_{standstill}$ is given a value of -50 which is less than the negative reward for collision but is still a fairly big reward so that the vehicle does not come to rest on the highway. The reason for this reward is because it was noticed that in some cases to avoid getting the negative rewards, the vehicle would just come to rest in the driving lane. To avoid this scenario $R_{standstill}$ is used.

The distance 'd' which represents the gap to be maintained between the vehicles in Table 4 is given a value of 160m. It is explained in the simulation setup in Chapter 4.1 that the endpoints of the highway in coordinate system are (0,0) and (5000,0) while the length of the highway is 40Km. Hence each unit in the x axis corresponds to 8m. The positions of the vehicles got from SUMO are in coordinate system. Hence the calculated d_1, d_2, d_3, d_4 and d are in coordinate system. Hence 160m is equivalent to 20 units which is used in calculation of the rewards.

R_{near} is given a value of -5, when the autonomous value gets near the vehicle in front (d_1<d). During each time step the vehicle is in this range the negative reward is added up. The idea here is that, once the driving agent senses that it has started getting the negative reward it can either change lanes if possible or try to decrease the velocity of the vehicle such that the vehicle comes out of this range ($d_1 > 160$m).

The next reward in row 4 uses $R_{LaneChange}$ as $\mathbf{50 - d_1}$. This is to make sure that the vehicle finishes the overtaking quickly. As d_1 will decrease during overtake, the reward got will increase during the overtaking period. This also make sure that the vehicle does not brake unnecessarily when it is in the overtaking lane. If when in the overtaking lane there is a vehicle in front (d_3<160m) then the reward value given in row 6 or row 7 comes into play. Here $\boldsymbol{R_{Lane1ApproachingVehicle} = 0.5}$. Hence if the vehicle decreases its speed +0.5 is given as the reward until d_3<160m and if the vehicle does not decrease its speed then -0.5 is given as the reward during each time step.

Also R_{Lane1}= -1.5×d_1 as shown in row 5. This is to avoid the vehicle from staying in the overtaking lane unnecessarily. This becomes active when $d_1 > 160$m. Hence as d_1 increases, R_{Lane1} will also increase negatively.

As already mentioned in Design of the Road Layout in Chapter 4.1, the speed limit in the highway is 22.2m/s. Hence a negative reward needs to be given when the vehicle is breaking the speed limit. Hence $R_{Overspeeding}$=-1. Also as mentioned in Motivation for Generic Reward System in Chapter 5.3, If the vehicle maintains its velocity to be equal to the speed limit, a positive reward is given which is set to be $R_{MaintainSpeedlimit}$=+2 and to encourage the vehicle to get to the speed limit a smaller positive reward of $R_{Speeding}$ = +1 is given when the vehicle accelerates towards the speed limit.

The reward scheme is showed in Table 5.

S.No	Reward Value	Reward Condition
1	-101	Collision occurrence
2	-50	Else & $v_a = 0$
3	-5	Else & L_a=L_0 & $d_1 <$ 160m
4	$50 - d_1$	Else & L_a=L_1 & d_1 <160m & $v_a > v_{at-1}$
5	$-1.5 \times d_1$	Else & L_a=L_1 & $d_1 >$ 160m
6	0.5	Else & L_a=L_1 & $d_3 <$ 160m & $v_a < v_{at-1}$
7	-0.5	Else & L_a=L_1 & $d_3 <$ 160m & $v_a > v_{at-1}$
8	-1	Else & $v_a >$ 22.5m/s
9	1	Else & $v_a > v_{at-1}$
10	2	Else & v_a = 22.5m/s
11	0	Else

Table 5 Calculation of Rewards adapted from Table 4

Changing this combination for the reward values meant that the driving algorithm did not perform as expected. For example, having a higher negative reward of -500 for collision meant that the autonomous vehicle would just stop as soon as it becomes active in the simulation even though it would get a negative reward for not moving, as the reward for collision highly outweighed the reward for stopping.

6.2 Results and Discussions from the Initial Design

The best performance got with this design during the test simulation was a collision percentage in the confidence interval between 20.5% to 28.4% found by Clopper Perason Binomial confidence method [51]. The test simulation consisted of testing the driving algorithm on 500 episodes.

This collision ratio is still very high. To solve this problem, the negative rewards for collision can be increased. This definitely solves the problem of collisions, but it also changes the autonomous vehicle into a car following vehicle in the driving lane which never goes for an overtake. Hence this means that there needs to be an external entity that prevents the collisions, so that the driving algorithm can only focus on manoeuvring through the traffic. To do this the collision avoidance system provided by SUMO was enabled, such that the autonomous vehicle will automatically brake if it comes very close (within 10 meters) to the vehicle in front. Still collisions are possible during lane change. Using this modification, the collision rate was bought down to be between 5.4% and 10.2%.

As explained before it was not possible to establish a direct relation between the configuration of the neural network and the performance of the driving algorithm. To test the neural network configuration best suited for this driving agent, the training and testing was done with different number of hidden layers and different number of nodes in each layer. The collision percentage during training and testing and the average total distance travelled during testing with 2 hidden layer, 3 hidden layer and 5 hidden layers are shown in Figure 16, Figure 17 and Figure 18 respectively. The number of nodes are varied from 1 to 2001 with an increment of 100. Having zero nodes in a hidden layer is equivalent to having no layer at all and hence the number of nodes were selected from 1.

From Figure 16, it can be seen that the driving agent trained with 3 hidden layers have slightly lower collisions than when trained with 2 hidden layers. In case of using 5 layers the fluctuation in the collision percentage with changes in nodes is quite high. The collision percentage when the number of nodes are 1 in each layer is however higher with 3 and 5 hidden layers than with 2 hidden layers. It was however noticed that, having just one node in the layers gave highly unpredictable results during each simulation as with another simulation for the same configuration with 2 hidden layers, collision percentage was found to be around 55% during training.

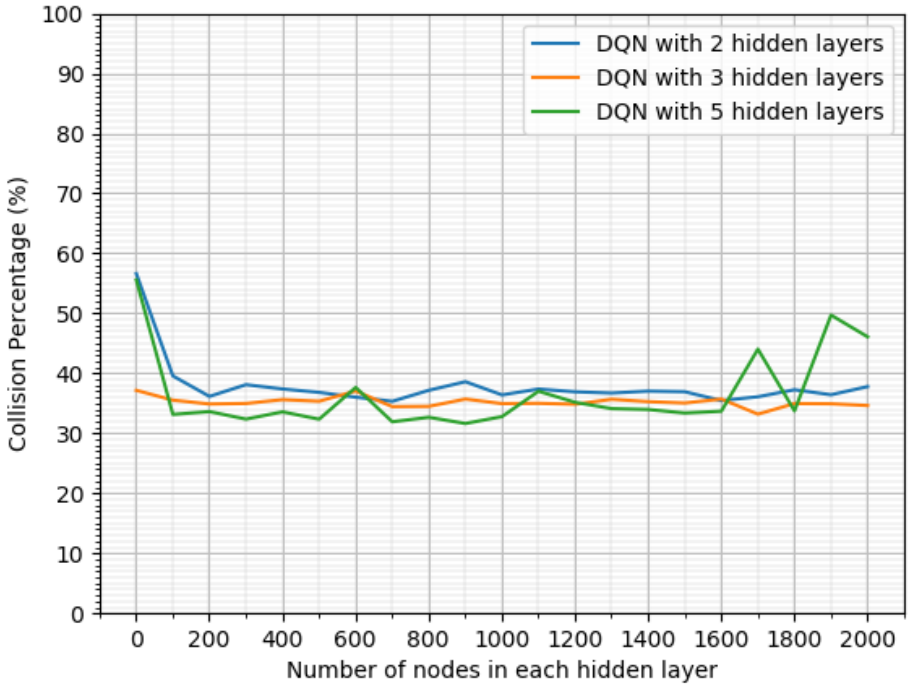


Figure 16 Collision percentage for different number of nodes and hidden layers during training

Figure 17 shows the collision percentage occurring with 2, 3 and 5 hidden layers during testing. The network was trained for 7000 episodes whose collision percentage is given in Figure 16. For testing the simulation was run for 500 episodes. For an ideal driving algorithm, the collision percentage should be very low and at the same time the vehicle should also travel at a good speed such that the travel time is low, as already mentioned in earlier chapters, it is possible to have a low collision percentage by staying in the driving lane itself and not going for any overtakes. To check if this is happening Figure 18 is used. It shows the average distance travelled by the autonomous vehicle during test simulations for each of the DQN (Deep Q Neural Network) layer configurations. The maximum speed of the slower moving vehicles is 11.1m/s. Hence if the average distance in the plot is less than 11100 meters then it means that the autonomous vehicle was behind a slower moving vehicle for a long time and hence did not do any overtaking.

From Figure 17, it can be seen that, with 5 hidden layers, the collision percentage is zero for most of the node configuration, but this is because the vehicle does not do any overtakes as can be seen from the average distance plot in Figure 18 where the average distance travelled by the vehicle during 0% collision with 5 hidden layers is less than 11000 meters. Also for other node configurations with 5 hidden layers where the collisions are lower, the average distance travelled by the vehicle is also quite low when compared with the average distance of the vehicle which used 3 hidden layers for learning.

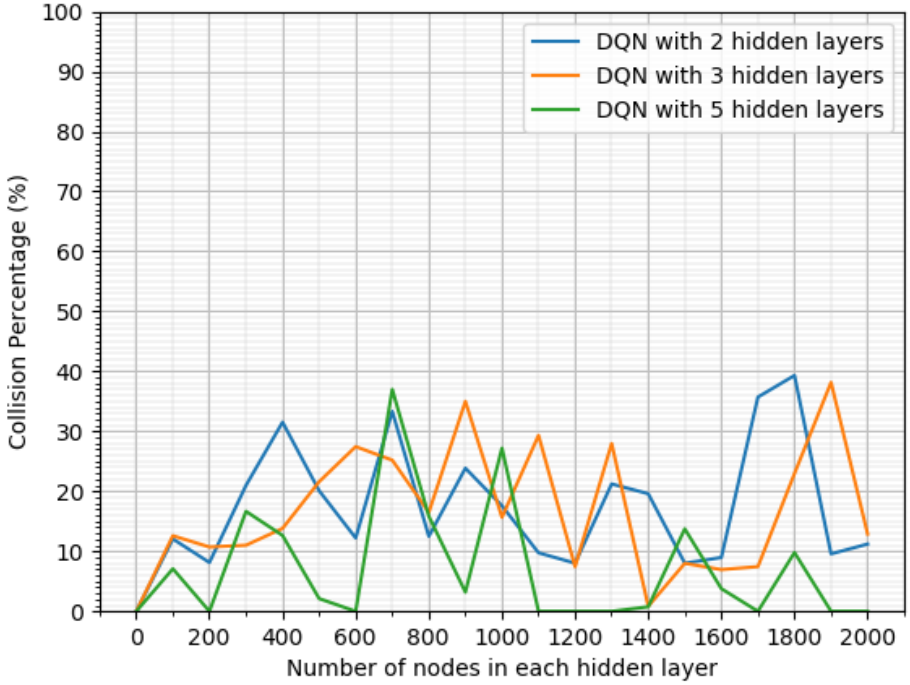


Figure 17 Collision percentage for different number of nodes and hidden layers during testing

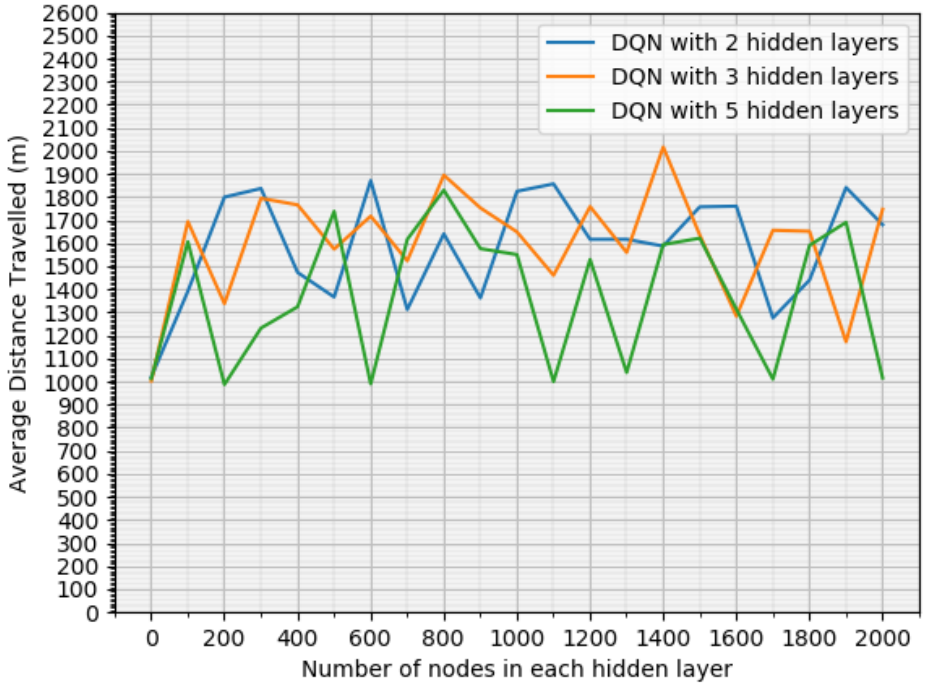


Figure 18 Average distance travelled by the autonomous vehicle for different node and layer configuration during testing

For the simulations with 2 hidden layers and 3 hidden layers, with just one node in each of the hidden layers, the driving algorithm does not learn much and it just learns to avoid the maximum negative reward and hence drives slowly in the driving lane. As the number of nodes increases, comparing these two configurations become difficult as they both show mixed performance for different node configurations in both the number of collisions and the average distance travelled. Hence foe the experiment 3 hidden layers were chosen as it gave a lower collision percentage during training.

Another strange observation in Figure 17 was at 1401 nodes in each hidden layer for a 3 layer DQN, the collision percentage was found to be 0% and the average distance travelled by the autonomous vehicle was also greater than 2000 meters, making it the best configuration for this learning algorithm. However, it was not able to reproduce this performance as again simulating with this configuration and testing it gave a collision percentage of 11%. Using 1500 nodes for each hidden layer for a 3-layer neural network gave a stable performance with the collision rate always being below 10%.

6.3 Conclusions from the Initial Design

The main aim of the initial design was to create a driving algorithm using Reinforcement learning which works on a 2 lane environment. The selection of input features, actions and the reward system in this design can be modified to create a generic driving algorithm for multi-lane highway environment, but the principle of selecting these features are the same.

From the results section in Section 6.2 which compared different layer and node configurations for the Deep Q network. It was found that having 3 hidden layers gave more stable performance during the training period than other configurations as shown in Figure 16. Hence three hidden layers is chosen as the number of layers for the DQN for the final experiment. Deciding the number of nodes for the hidden layers was trickier as there were quite a few node configurations possible which gave low number of collisions, especially between 1400 and 1700 nodes for a 3 layer neural network as shown in Figure 17. Hence further study needs to done to decide exactly how to choose the number of nodes for the hidden layer. For the final experiment 1500 nodes were chosen for each hidden layer, as with this configuration the collision percentage deviation between different simulations were found to be low ($\cong$2.5%).

Another important thing to take in consideration while choosing the number of layers and the number of nodes is that, with increase in the number of nodes or number of hidden layers the time required to train the neural network increases. This is the reason for not considering hidden layers greater than 5 for testing the performance of the driving algorithm.

7. Final Experiment for driving in a Multilane Highway Environment

Based on the results of the initial implementation in Chapter 6, a generic driving algorithm is designed which can be used on a multilane highway environment. For designing this new algorithm changes had to be made in the number of input features and the output actions. The same rewarding scheme that was used in the previous design given in Chapter 6.1 is used even here with slight modifications. The rewarding scheme for the previous design only took into account two lanes whereas here the number of lanes can be more than two. Hence this is incorporated in the rewarding scheme for this design.

This chapter talks about the design and implementation of this driving algorithm. It also talks about the implementation done to test the performance of the driving algorithm in case of packet loss due to communication error and also with change in communication range.

7.1 Input Features

In highways with more than 2 lanes the required information of the surrounding vehicles changes as shown in Figure 19. This is true when the autonomous vehicle is in any of the overtaking lane other than the last lane, which in this case is L_2.

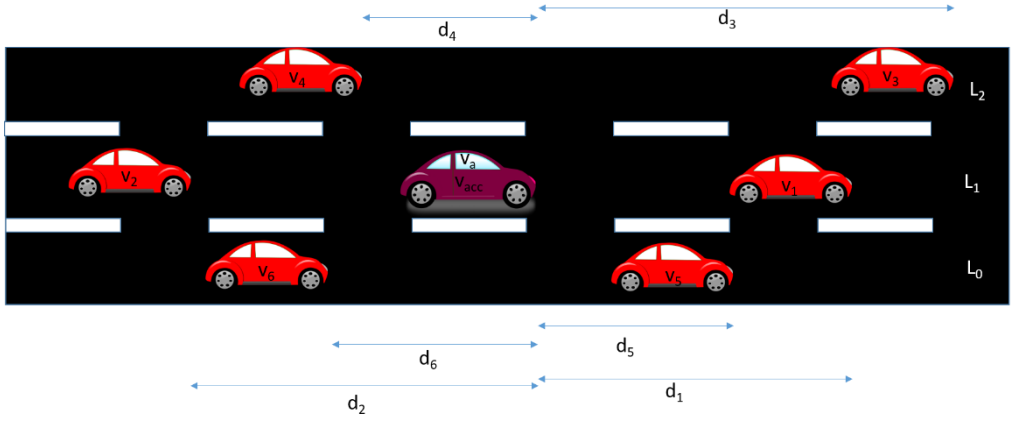


Figure 19 Position of vehicles when the autonomous vehicle is in one of the middle lanes

The earlier design in Chapter 6 used the positions of surrounding vehicles with respect to the lane it was associated with. That is d_1 and d_2 were the gap with the vehicles in lane 0 (driving lane) and d_3 and d_4 were the gaps between the vehicles in lane 1 (overtaking lane). In this design the position of the surrounding vehicle is decided with respect to the lane the autonomous vehicle is in. Hence if the autonomous vehicle is in lane 1 as shown in Figure 19, then d_1 and d_2 are distances with the vehicles in that lane and v_1, v_2 being their respective velocity. d_3 and d_4 are the distance with the vehicles in the lane left to the autonomous vehicle, which will

then be the overtaking lane and v_3 and v_4 will be the velocities of those vehicles. The distances d_5 and d_6 will be for the vehicles in the right lane and v_5 and v_6 will be their velocity.

Now if the vehicle is in the driving lane or in the last overtaking lane then the states will change as shown in Figure 20 and Figure 21. Figure 20 shows when the autonomous vehicle is in L_0 (driving lane). Then as there are no more lanes to its right side, d_5, d_6, v_5, v_6 will not exist. Hence default values are given for these parameters. Usually when d_5 or d_6 is 800m, then it means that there is a vehicle only at a distance of 800m in the left lane and that the autonomous vehicle is in the overtaking lane unnecessarily, but the lane number which is another input state can be used to determine if the vehicle is in any of the overtaking lanes or in the driving lane. This is explained in the reward calculation section.

Figure 21 shows the changes in distance between the vehicle states and vehicle velocity states when the autonomous vehicle is in the left most lane. Here as there won't be any more vehicles to the left side, d_3, d_4, v_3 and v_4 wont exist. Hence they are all given the default values.

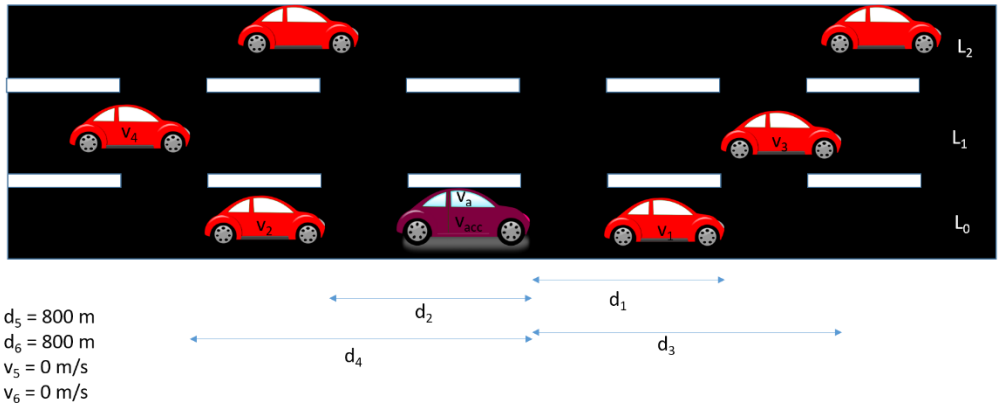


Figure 20 Position of vehicles when the autonomous vehicle is in the driving lane

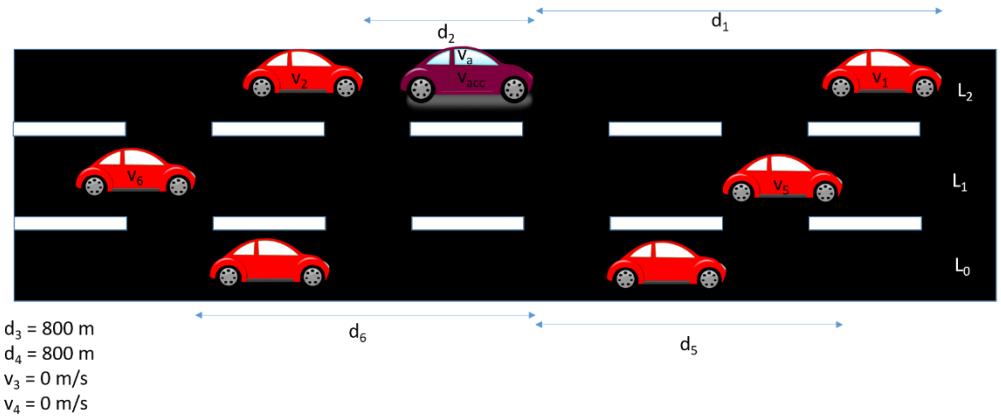


Figure 21 Position of vehicles when the autonomous vehicle is in the last overtaking lane

The new input features will be:

1) Current velocity of the autonomous vehicle (v_a).
2) Current velocity of the vehicle in front in the same lane (v_1).
3) Distance the autonomous vehicle is from the vehicle in front in the same lane (d_1).
4) Current velocity of the vehicle behind in the same lane (v_2).
5) Distance the autonomous vehicle is from the vehicle behind in the same lane (d_2).
6) Current velocity of the vehicle in front in the left lane (v_3).
7) Distance the autonomous vehicle is from the vehicle in front in the left lane (d_3).
8) Current velocity of the vehicle behind in the left lane (v_4).
9) Distance the autonomous vehicle is from the vehicle behind in the left lane (d_4).
10) Current velocity of the vehicle in front in the right lane (v_5).
11) Distance the autonomous vehicle is from the vehicle in front in the right lane (d_5).
12) Current velocity of the vehicle behind in the right lane (v_6).
13) Distance the autonomous vehicle is from the vehicle behind in the right lane (d_6).
14) Current lane the autonomous vehicle is occupying (L=0 to maximum number of lanes).
15) Current acceleration rate of the autonomous vehicle (v_{acc})

The input state s_t at time t will be a collection of $\{v_a, v_1, d_1, v_2, d_2, v_3, d_3, v_4, d_4, v_5, d_5, v_6, d_6, L_a, v_{acc}\}$

7.2 Output Actions of the Driving Agent

The number of action states has been changed to be more generic. The new action states are:

1) Changing to the left lane provided that the vehicle is not already at the left most lane.
2) Changing to the right lane provided that the vehicle is not already at the right most lane.
3) Increase velocity of the autonomous vehicle.
4) Decrease velocity of the autonomous vehicle.
5) Idle action.

An additional idle action is added to remain in the same lane. In the previous action state for two lane highway the number of lanes were fixed. Hence if the vehicle was in Lane 0, then to remain in the same lane the driving agent could choose 'Move to Lane 0' as the action. In this design as the number of lanes are not fixed. The action for Lane change cannot be used for staying in the same lane. Hence an idle action is used for remaining in the same lane.

7.3 Reward System

Rewarding Scheme similar to the one used in Table 5 in Chapter 6.1 is used for this design and is given in Table 6. The differences between both the reward systems are that the reward system in Table 5 considered the highway to have only two lanes. Hence L_0 was the driving lane and L_1 was the overtaking lane. Therefore, if there was a vehicle getting close in front of the autonomous vehicle in the driving lane, a negative reward was given. In this implementation as the number of lanes can be more than two, the driving and the overtaking lane will depend on which lane the autonomous vehicle is driving on. Unless the autonomous vehicle is not at the Max Lane number (L_{max}) which is the left most lane, it can change to an overtaking lane to overtake the vehicle in front. This is the reason in row 3 of Table 6 $L_a \neq L_{max}$ is checked before giving the negative reward.

As explained in Section 7.1 in establishing the input features for this design is different to the ones used in the design in Chapter 6. Here the distance with the vehicle in front in the right lane is given by d_5. The information given by d_5 is the same as what d_1 gave in Table 5. Also all the lanes other than the first lane (L_0) are overtaking lanes. Hence to check that the vehicle finishes the overtaking quickly and also to avoid having the vehicle in the overtaking lane unnecessarily, rewards in row 4 and 5 are used.

As it is not possible to overtake vehicles in the last lane, rewards in row 6 and row 7 are used. In Table 5, the max lane was always 1, but here as this is a generic reward system, $L_a = L_{max}$ is used to define the max lane number.

The values for the rewards are the same as what is given in Table 5.

S.No	Reward Value	Reward Condition
1	-101	Collision occurrence
2	-50	Else & $v_a = 0$
3	-5	Else & $L_a \neq L_{max}$ & $d_1 <$ 160m
4	$50 - d_5$	Else & $L_a \neq L_0$ & d_5 <160m & $v_a > v_{a,t-1}$
5	$-1.5 \times d_5$	Else & $L_a \neq L_0$ & $d_5 >$ 160m
6	0.5	Else & $L_a = L_{max}$ & $d_3 <$ 160m & $v_a < v_{a,t-1}$
7	-0.5	Else & $L_a = L_{max}$ & d3 < 160m & va > $va_{,t-1}$
8	-1	Else & $v_a >$ speed limit (22.5m/s)
9	1	Else & $v_a > v_{a,t-1}$
10	2	Else & $v_a =$ speed limit (22.5m/s)
11	0	Else

Table 6 Reward calculation for multilane Driving algorithm

7.4 Handling Communication Error

A provision is implemented to induce communication error as it is possible during vehicle to vehicle communication. This allows to test the performance of the driving algorithm in case of unavailability of vehicle information required by the learning algorithm. The type of communication error tested in this simulation is shown in Figure 22.

As shown in Figure 22, for the vehicles lying outside the communication range, the probability of successfully receiving its vehicle parameters by the autonomous vehicle is 0. This means that the vehicle information of these vehicles are not received by the autonomous vehicle. Hence the driving algorithm assumes that there is no vehicle in those positions. The default communication range used for this research is 800 meters. For the vehicles lying inside the communication range, the error of receiving the vehicle information by the autonomous vehicle is determined by the error probability. If the error probability is 0.5 then it means that there is a 50% possibility of not receiving the communication packet during each time step. The results of this simulation is shown in Chapter 8.

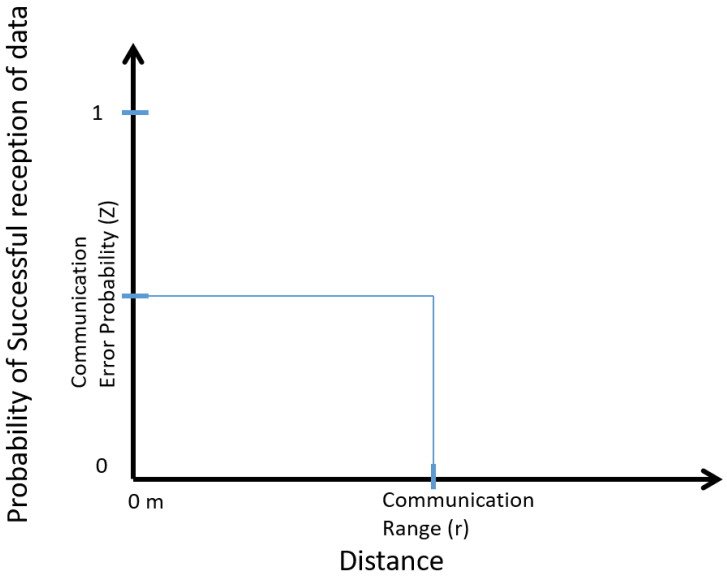


Figure 22 Probability of successful reception with respect to the communication range and error probability

To cope with the communication error, a modification was made in the design of the driving algorithm. A buffer is maintained in the environment side of the driving algorithm (shown in Figure 6) which stores the vehicle parameter received in the last successful communication. If it is observed that a communication error has occurred, then the vehicle information from the last communication is used as the current vehicle information by the driving algorithm. The driving algorithm performed better with this approach which is also discussed in Chapter 8.

8. Results and Discussions

This chapter discusses the results of the implementation of the generic driving algorithm discussed in Chapter 7.

The values of the different parameters which includes the simulation settings and also the machine learning parameters are given in Table 7.

Speed limit in the highway	22.22m/s (80Km/Hr)
Maximum possible speed of the autonomous vehicle	55.55m/s (200Km/Hr)
Maximum possible speed of the slow moving vehicles	11.1m/s (40Km/Hr)
Maximum possible speed of the other vehicles	55.55m/s (200Km/Hr)
Number of input features for the driving agent	14
Number of possible output actions from the driving agent	5
Number of hidden layers in the neural network	3
Number of nodes in each hidden layer	1500
Learning rate (α)	0.001
Discount factor (γ)	0.9
Exploration rate (ε)	0.9
Probability of Traffic rate of slow moving vehicles per second	0.1
Probability of Traffic rate of normal speed vehicles per second	0.01
Training Period	7000 episodes
Testing Period	500 episodes

Table 7 Parameters for Reinforcement learning and SUMO simulation used in the final implementation

8.1 Performance of the driving Algorithm in a 3 lane highway and a 2 lane highway

The collision percentage during the training period trained on a three lane highway is shown in Figure 23. Even though the training was done for 7000 episodes, the plot shows episodes fewer than 7000. This is because as the vehicles are inserted in the simulation randomly, there is a possibility that the autonomous vehicle collides with another vehicle within the first time step as explained in Chapter 3.3. These episodes are not taken into account for evaluating the performance of the driving algorithm. Hence the number of episodes in Figure 23 is lower than 7000.

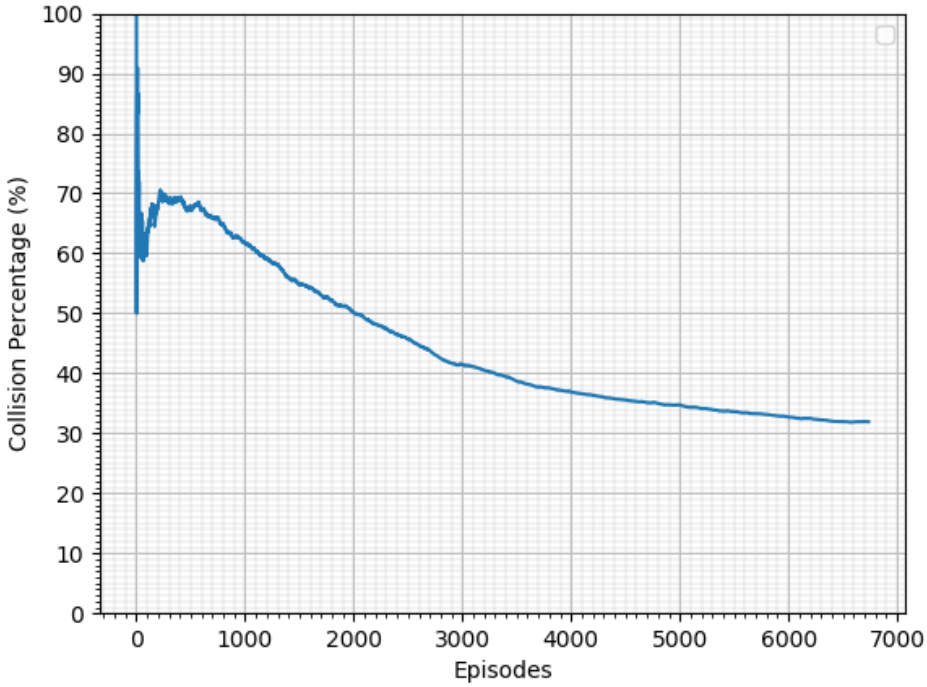


Figure 23 change in collision percentage during training period

As mentioned earlier in Chapter 6, an exploration rate of 0.9 is used at the start which is then decreased as the simulation progresses. When the exploration rate is 0.9, 90% of the actions that are taken are in random and not based on the Q values. For this value to reduce to 0.5, it takes close to 735 episodes. Hence for the first 735 episodes the actions are mainly taken in random, that is the reason for the curve to have a spike and high collisions until that number of episodes. After that, as the actions are taken mainly based on the learning done by the driving algorithm until 2746 episodes. There is a steady decrease in the collisions. By 6500 episodes, the curve has started decreasing very slowly, meaning that the performance is not improving much beyond this point. The training was also done for 10000 episodes, but the results found were similar to what is shown in Figure 23.

The total reward attained during each episode while training the driving agent is shown in Figure 24. It can be observed that until around 2000 episodes, the total rewards obtained are mostly negative. This is again because during this time period the driving agent is trying to learn by choosing random actions based on its exploration rate. With progression of episodes, the frequency of positive rewards increases, but there are still cases where a low reward is earned. This is because there might have been scenarios where the vehicle would not have had an available action that gave a positive reward, as the reward system is designed mainly to discourage the driving agent in making the wrong decisions. Hence most of the rewarding scheme deals with giving negative rewards as shown in Table 6 and positive rewards

are given only for few conditions. One example that could lead to a negative reward without the vehicle making any mistake is if the autonomous vehicle is close to the vehicle in front and the overtaking lane is not free. As the vehicle is close to the vehicle in front, a negative reward of -5 (based on row 3 of Table 6) is given to the driving agent during each time step the vehicle is below 160m to the vehicle in front. The vehicle cannot also move to the overtaking lane as that would give a higher negative reward of -101 (in case of collision). Hence to avoid getting the negative reward it needs to slow down, such that when the vehicle in front is at a distance greater than 160m, it will start getting a reward of 0. Hence it avoids getting the negative reward but the already attained negative reward is not compensated with positive reward. Hence the total reward in the episode might still be negative even though the driving agent behaved as expected.

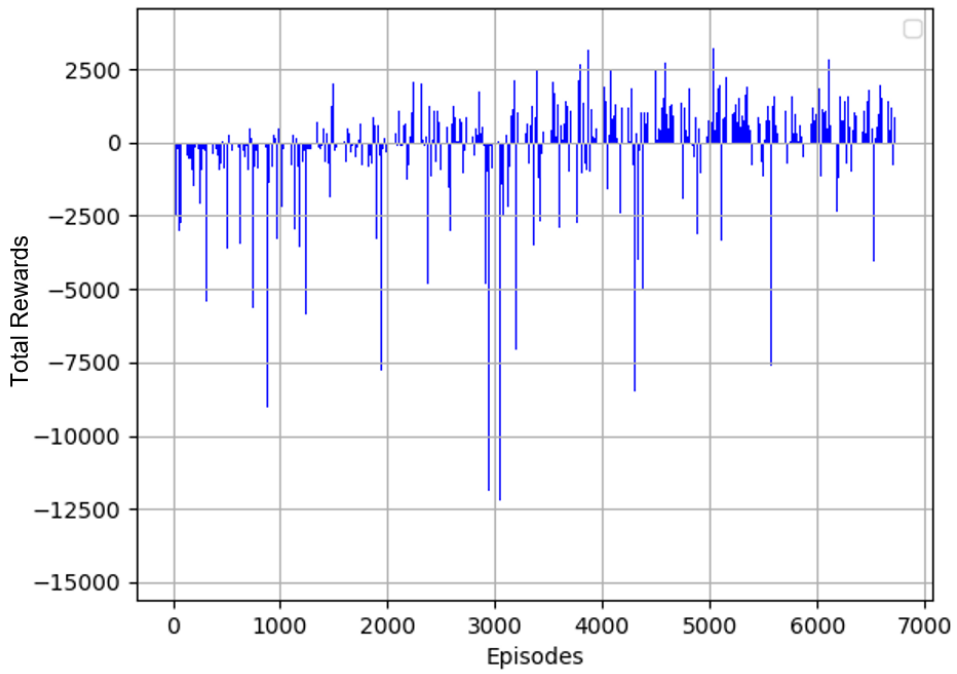


Figure 24 Total Rewards attained by the driving agent in each episode

The snapshots of the simulation output showing the overtaking procedure in a 3 lane highway environment is shown in Figure 25 to Figure 30.

In Figure 25, the autonomous vehicle(blue coloured vehicle) is just inserted in the driving lane. Hence the driving algorithm will base its decision only based on the vehicle parameters of the vehicles in front and in the left lane. As there is a vehicle close by in front, the action taken by the vehicle is to move to the overtaking lane. Which is the lane to the left as shown in Figure 26.

Figure 25 The autonomous vehicle (blue) is inserted into the simulation

Once in the second lane, the driving algorithm uses the vehicle parameters of the surrounding vehicle in the first lane, second lane and also the third lane to take the appropriate action. As there is a vehicle close by in front and the third lane is free, the driving algorithm decides to change to the third lane as shown in Figure 27.

Figure 26 The autonomous vehicle (blue) moves to the 2 lane for overtake

Figure 27 The autonomous vehicle (blue) moves to the third lane for overtake

As the third lane is free, the autonomous vehicle increases its velocity and overtakes the vehicles in the second lane as shown in Figure 28. Once it has finished overtaking and the second lane becomes free, the autonomous vehicle returns to the second lane to avoid accumulating negative rewards as shown in Figure 29. Once back in the second lane the driving algorithm again uses the vehicle parameters of the surrounding vehicles of all the three lanes and it finds out that even lane 1 is free. Hence the vehicle is moved back to lane 1 (driving lane) to avoid accumulating negative rewards for being in an overtaking lane unnecessarily. This is shown in Figure 30.

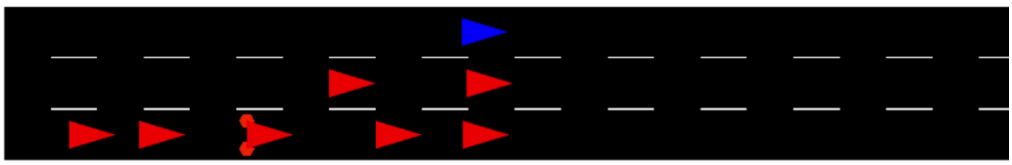


Figure 28 Overtaking the vehicle in second lane

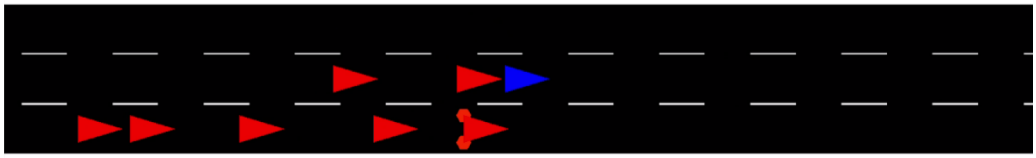


Figure 29 Autonomous vehicle (blue) back in the second lane after overtake

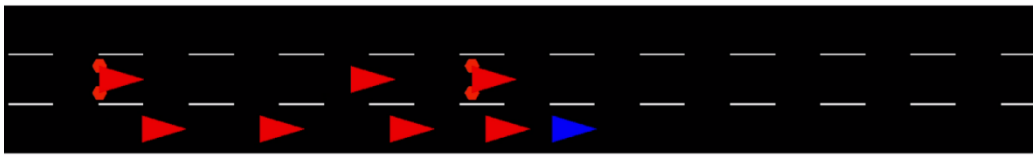


Figure 30 Autonomous vehicle (blue) back in the driving lane

The probability of collision with this trained driving algorithm on test simulations is was found to be between the confidence interval of 0.08 and 0.14 using the Clopper Pearson Binomial confidence method. Figure 31 shows the frequency distribution of the distance travelled by the autonomous vehicle during the test simulation in a 3 lane highway environment. The average distance travelled by the vehicle was 1842m, which means that its average speed was 18.42m/s.

The performance of the same driving algorithm on a 2 lane highway environment with the same traffic density was also tested and the probability of collision was found to be between 0.015 and 0.048 in a simulation of 500 episodes. Figure 32 shows the frequency distribution of the distance travelled by the autonomous vehicle during the test simulation in the 2 lane highway environment. The average distance travelled was 1340.1m, which means that the average speed of the autonomous vehicle was 13.4m/s.

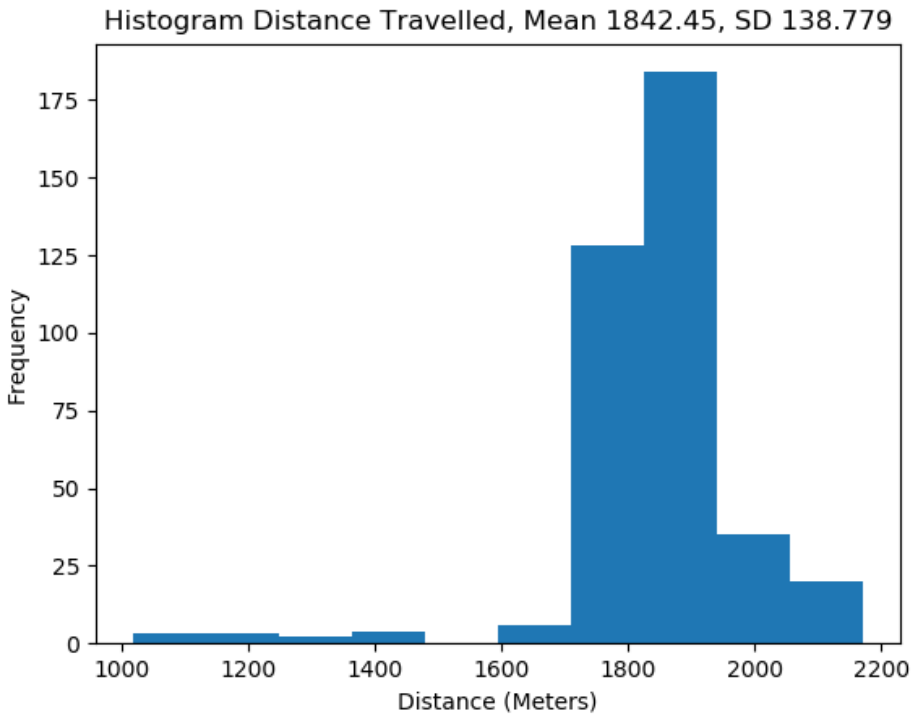


Figure 31 Frequency distribution of distance travelled in a simulation with 500 episodes

It can be observed from the collision probabilities that the probability of collision is lower in a two lane system than in a 3 lane system with this driving algorithm. The reason for this is because, the number of overtaking possible in the 3 lane system is higher than in a 2 lane system, as the traffic density is the same. Hence in a 2 lane system, the vehicle would have to follow the vehicle in front more than overtaking it. This can be shown by the average distance travelled by the autonomous vehicle in both these environments shown in Figure 31 for a 3 lane system and Figure 32 for a 2 lane system.

In Figure 31, the mean distance travelled by the autonomous vehicle is 1842.45 meters which means the average speed of the vehicle in that simulation was 18.42 m/s (as the simulation was for 100 seconds) which is a lot higher than the average velocity calculated from Figure 32 which comes to be 13.4 m/s. Hence in the 2 lane system, the vehicle is not able to do a lot of overtakes as in a 3 lane system due to lower number of lanes for the same traffic density.

In the current driving algorithm as the collisions are possible only during lane change and as the number of lane changes are higher than the lane changes in a 2 lane environment, the collisions in a 3 lane environment is higher.

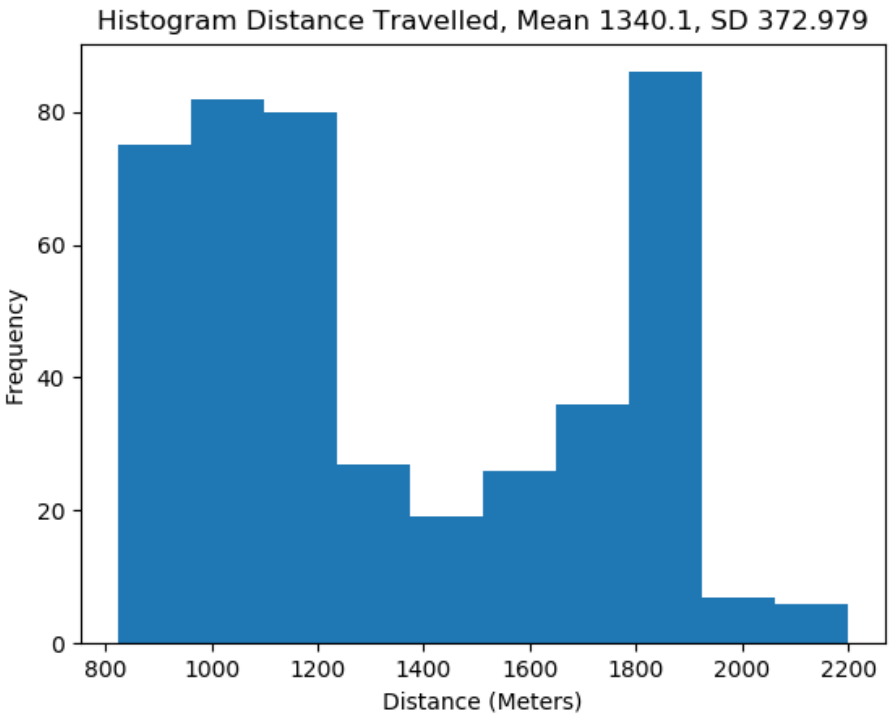


Figure 32 Frequency distribution of distance travelled in a simulation with 500 episodes

It can also be observed that the probability of collision in the two lane highway with this method is way lower than the probability of collision measured with the initial design which gave a collision percentage between 5.4% and 10.2%. This might be because of the extra output action in the generic implementation which tells the vehicle to maintain the same state. This was not present in the initial design as there were only two lanes and hence it was assumed that if the vehicle had to remain in the same state, then the action corresponding to moving to that lane can be selected. This means there are two actions that is related to staying in the same state and the driving algorithm needed to select the correct one based on the current input state. This might have reduced the performance of the driving algorithm which resulted in higher number of collisions.

8.2 Performance of the driving algorithm during communication error and limited communication range

As discussed in Section 7.4, The performance of the driving algorithm was tested with limited communication range and during loss of communication packets. The testing was done by inducing a probability of packet loss for different communication ranges in a 2 lane highway environment. It was observed that, by training the driving agent with packet loss, the agent did not learn much as its input states were not accurate due to packet loss. Hence the agent learnt to be cautious and

drive the autonomous vehicle at very low speed below 9m/s so as to avoid any collisions. This also prevented the vehicle from overtaking any other vehicle in the highway. Therefore, training the driving algorithm was done with no packet loss so as to test how a fully learnt driving agent can cope with packet loss.

Figure 33 shows the percentage of collisions for 3 scenarios of packet loss with increase in communication range and the average distance travelled by the autonomous vehicle during these simulations is given in Figure 34. It can be seen that when the communication range is 0, then it means that the vehicle does not get the information of any surrounding vehicles, but the collision percentage of all the three plots are 0%. This is because, as the driving algorithm assumes there are no vehicles in the vicinity, the autonomous vehicle will only travel in the driving lane. As it approaches a vehicle in front, the collision avoidance system in the autonomous vehicle will automatically detect the vehicle in front and slow down the autonomous vehicle. This can be seen in Figure 34 where the average distance travelled by the autonomous vehicle for all the three plots when the communication range is 0, is very low at 900m. The average speed of the autonomous vehicle was 9m/s which is slower than the velocity of all the other vehicles in the simulation. When there is no packet loss, then until the communication range is 200 meters the average collision percentage is quite high especially until 30 meters where the collision percentage is around 61%. Above the communication range of 200 meters the collision percentage is quite close to the normal collision percentage achieved during this research which is below 10%. Hence this means that the driving algorithm should at least get the vehicle parameters of the vehicles within a range of 200 meters from the autonomous vehicle to make correct and safe decisions even when there is no possibility of packet loss.

The number of collisions increases drastically when there is loss of packets as shown by the plot corresponding to collision percentage in case of 50% packet loss. As data is missing at random times, the driving agent senses that a vehicle is not present even when it is there. Most of the collisions that happen here is when the autonomous vehicle tries to move back to the driving lane from the overtaking lane, when the driving agent mistakenly senses that there is no vehicle present in the driving lane to overtake. For example, suppose the autonomous vehicle is in the driving lane and there is a vehicle close to it in the front. If the communication takes place in this step, then the driving agent knows that a vehicle is present in the front and it decides to move the autonomous vehicle to the overtaking lane. Now while overtaking if there is an error in communication and data doesn't come through, then the autonomous vehicle will have no clue of the position of the surrounding vehicles. Hence the driving agent will assume that the autonomous vehicle is in the overtaking lane unnecessarily and will move it back to the driving lane based on the current reward mechanism. In reality though, as there is already a

vehicle in the driving lane, there is a high possibility of collision with this vehicle while moving back to the driving lane. Hence the high amount of collisions.

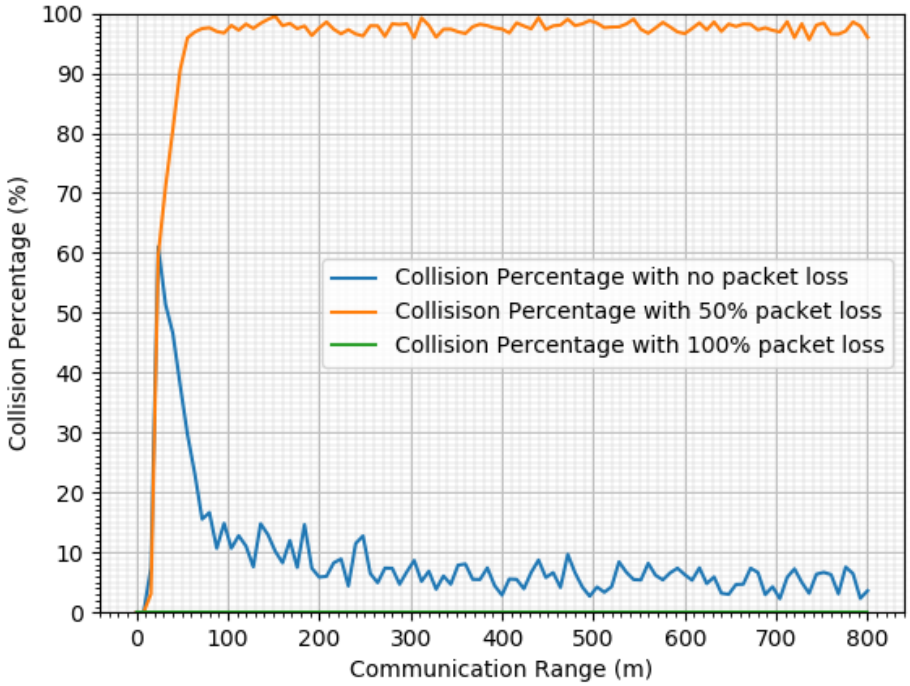


Figure 33 Collision percentage with increase in communication range for different packet loss

From Figure 34, it can be observed that there is a large variation in the plot corresponding to the average distance travelled by the autonomous vehicle during 50% packet loss. The reason for this is still not clear, but one of the reasons for this might be that the average distance is calculated on the episodes where the autonomous vehicle successfully finishes its episode without a collision. When the packet loss is 50%, the collision percentage was around 98% as can be seen from Figure 33. As testing is done on 500 episodes, this means only distances covered in 10 episodes are taken for calculating the average distance. Hence as the average distance is calculated in less number of episodes, there might be a big deviation in the average distance travelled by the autonomous vehicle.

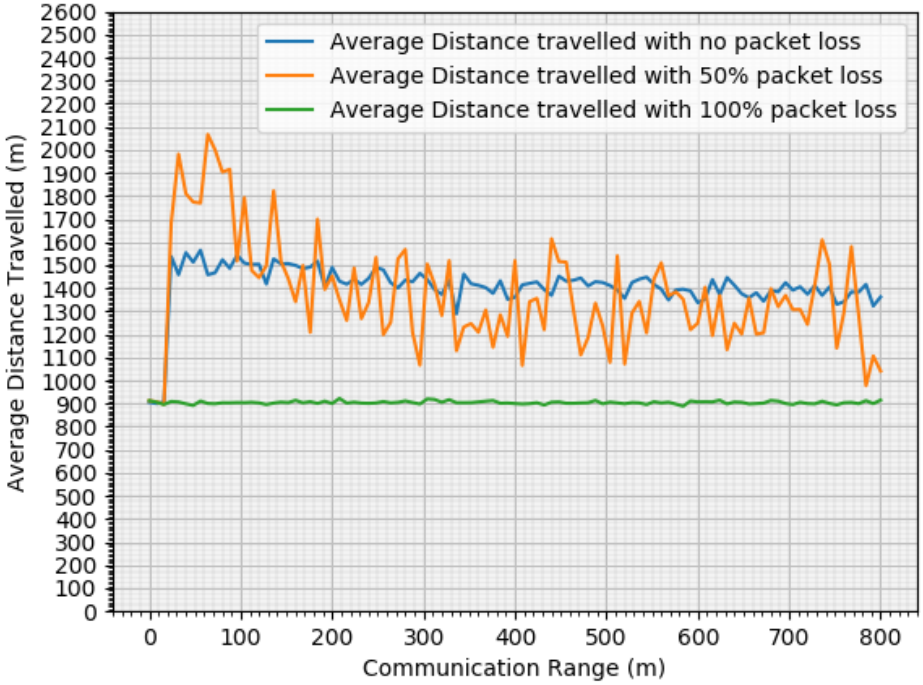


Figure 34 Average distance travelled with increase in communication range for different packet loss

When there is packet loss at all time, then the scenario is similar to when the communication range is 0m. As the driving agent will not have any idea of the distances with the vehicles around it, it would assume that there are no vehicle surrounding it and the autonomous vehicle will be driven only in the driving lane. Hence the collision avoidance system on the vehicle will prevent collisions with the vehicle in front. As a result the collision percentage during 100% packet loss is 0% as shown in Figure 33. Also as the vehicle only moves in the driving lane, it does not do any overtakes and hence the average distance travelled by the autonomous vehicle is also quite low at around 900m as shown in Figure 34.

The change in collision percentage with increase in packet loss is shown in Figure 35. It can be seen that as the probability of communication error increases, the collision percentage also increases and finally when the communication error probability is 1, that is the autonomous vehicle does not receive any information of the vehicles around it, it moves only in the driving lane and as a result the collision avoidance system of the vehicle takes care that it doesn't collide with the vehicle in front. Hence the collision percentage becomes zero when the communication error probability is 1.

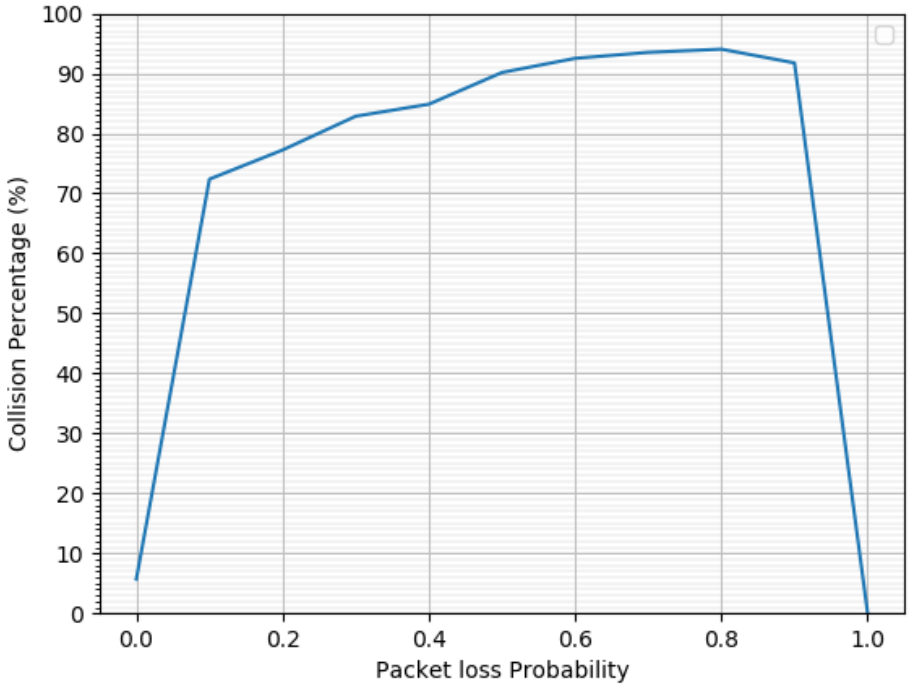


Figure 35 Collision Percentage with change in Communication Error Probability for communication range of 800m

8.3 Performance of the Driving algorithm with Error Concealment Technique

In Chapter 7.4, to handle packet loss, an error concealment technique was designed. The performance of the driving algorithm with this error concealment technique is given in Figure 36 and Figure 37. Figure 36 gives the collision percentage of the autonomous vehicle for different communication range and three types of packet loss. It can be seen that when there is no packet loss or when there is 100% packet loss during the entire episode, then the collision plots are similar to the one in Figure 33. This means that the error concealment technique does not affect the performance of the driving agent in case of zero packet loss. When there is 100% packet loss, this technique is not useful as it works based on the previous successful reception of the surrounding vehicle information. Hence when packet loss is 100%, the vehicle information of the surrounding vehicles is never got during the entire episode and hence the algorithm works as it did without the error concealment technique.

When there is a packet loss of 50% during the episode, the collision percentage is 0% when the communication range is 0m. This is again because the autonomous vehicle keeps to the driving lane at all times and hence the collision avoidance algorithm of the autonomous vehicle prevents it from colliding with the vehicle in

front resulting in the autonomous vehicles moving at a slower velocity than the slowest vehicle in the simulation as can be calculated from the average distance plot given in Figure 37. Until around 30m of communication range, the collisions are still higher, close to 70%. The percentage of collisions though is less than what was observed in Figure 33 where it was above 90%. This is because, now even if there is a communication error, the last known position of the vehicle is used as the current position and hence the vehicle can still react based on that. As the average speed of the autonomous vehicles is around 14m/s. The reaction time it has when the surrounding vehicle is within 30m, is around 2.14s. This means 2 time steps. Hence if there is a gap in communication for two consecutive, time step and there is a vehicle nearby, within 30m then the possibility of collision is very high. With increase in communication range the collisions decreases, as the autonomous vehicle has more time to react using the last correct communicated information.

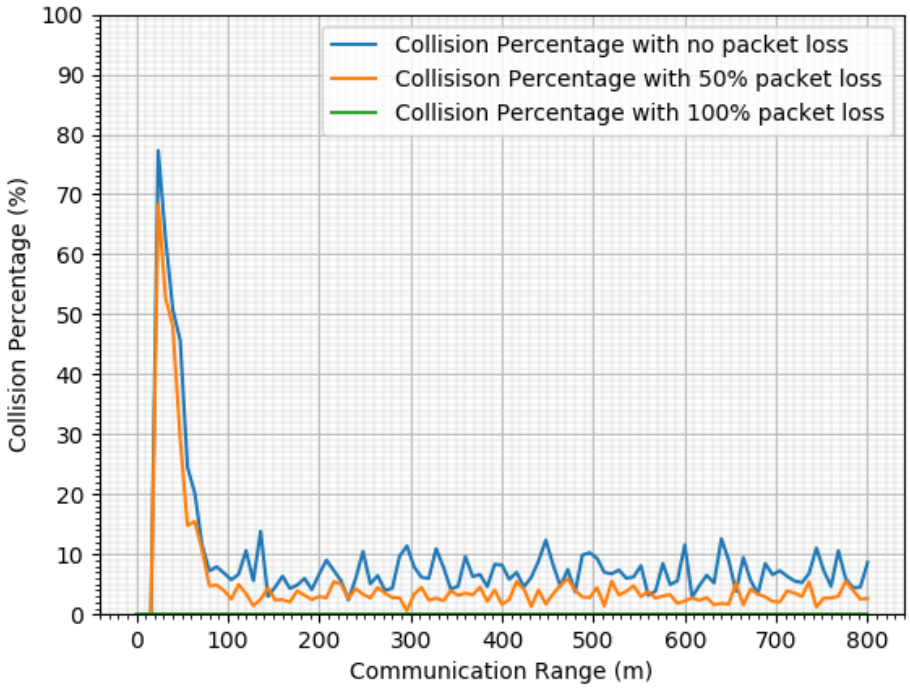


Figure 36 Collision percentage for different communication ranges and different packet loss probabilities and with implementation of error concealment technique

It can be observed from Figure 36 that the collision percentage is slightly lower in case of 50% packet loss than when there is no packet loss. The reason for this might be as the vehicles all travel in the same direction, taking an action based on the last position of the vehicle may not create a scenario for collision. As already explained, in this research collisions are only possible during lane change. Hence a collision can happen if the autonomous vehicle tries to change from the driving lane to the overtaking lane or when the autonomous vehicle tries to change from the

overtaking lane to the driving lane. Suppose there is a loss of packet during a time step when the autonomous vehicle is behind another vehicle in the driving lane. To decide whether to move to the overtaking lane or not, the driving agent will use the previous distances with the surrounding vehicles. By this time the vehicle in front would have already moved further and hence using the previous distances will not cause a collision with that vehicle while coming back to the driving lane after overtaking. However, there is a possibility of colliding with a vehicle while changing to the overtaking lane. This is possible if there was a vehicle further behind in the overtaking lane while there was no loss in communication packets, but due to packet loss at the time of overtaking, that distance is used by the agent to decide to move to the overtaking lane. This might lead to a collision for the autonomous vehicle and the vehicle behind in the overtaking lane. As can be calculated from Figure 37 the average speed of the autonomous vehicle during 50% packet loss is between 12 and 15m/s. Hence for a vehicle to overtake the autonomous vehicle, it should have a maximum speed greater than 12m/s. As can be seen from Table 7, only one type of vehicle have a maximum speed great than 14m/s. Hence only they are capable of overtaking the autonomous vehicle. As the probability of insertion of this vehicle into the simulation is quite low (=0.01), the chance of this condition occurring is also quite low. This might be the reason for lower number of collisions. Still more study and testing needs to be done with different traffic rates to find if this is the correct reason for this behaviour.

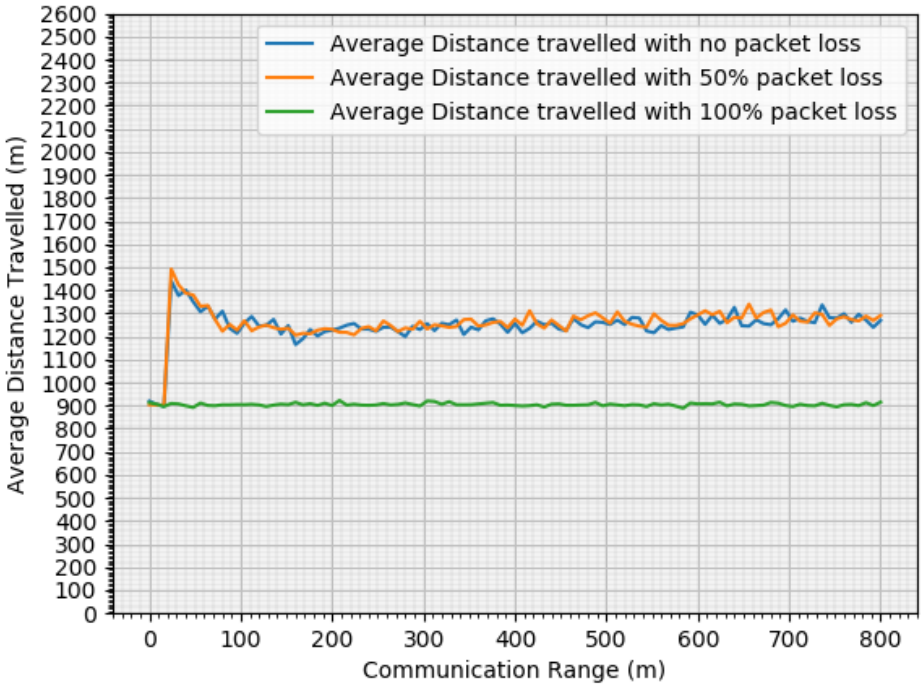


Figure 37 Average distance travelled by autonomous vehicle for different communication range and packet loss probabilities

The change in collision percentage for the driving algorithm with the implementation for error concealment with respect to increase in communication error probability is shown in Figure 38. It can be seen that the collision percentage is very low when compared with the one got in Figure 35 and is within 6%. Hence it is possible to use certain techniques in the driving algorithm to cope with communication packet loss.

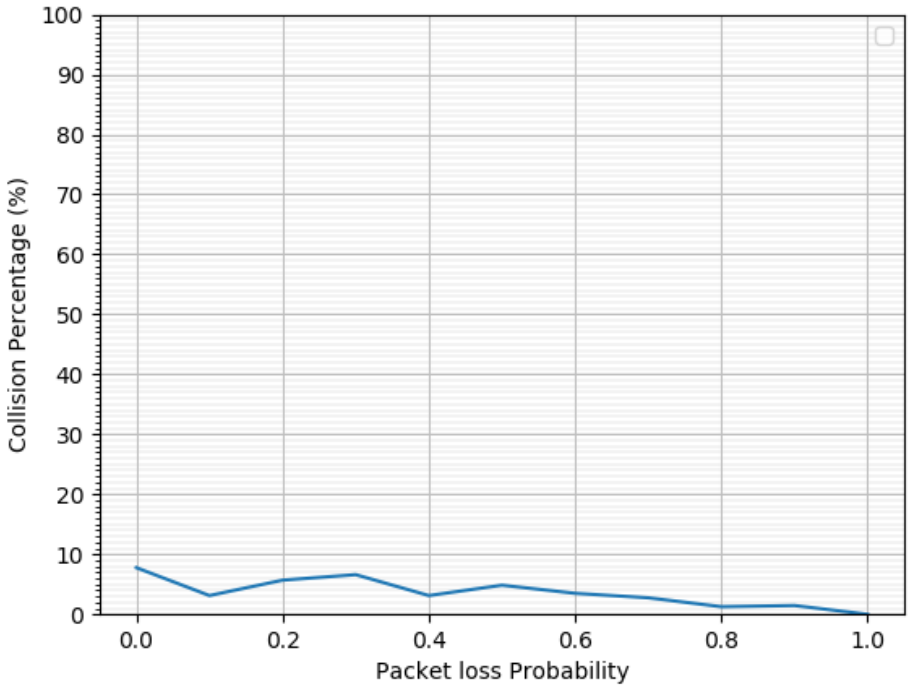


Figure 38 Collision Percentage with change in Communication Error Probability for communication range of 400m

9. Conclusion and Recommendation

Four research goals were introduced in Section 1.2. This chapter gives the conclusions reached on these goals and also give certain recommendations that can be used for further improving and continuing with this research.

9.1 Conclusion

The first goal of this research was to investigate to what extent reinforcement learning using Q learning can be used with cooperative driving to manoeuvre an autonomous vehicle in a multi-lane highway. For this a driving algorithm was designed with a driving agent using Deep Q Neural Networks for controlling a vehicle in a highway environment by changing the lanes or by accelerating or decelerating depending on the vehicle information got from the surrounding vehicles. The simulation of the driving algorithm produced around 20.5% to 28.4% of collisions in 500 episodes, even though collision avoidance was the main criteria in the reward system. Hence it is better to have collision avoidance as an external entity to the driving agent, such that the action taken by the driving agent is only executed by the driving algorithm if it does not lead to a collision. An implementation similar to this where the collision avoidance entity takes care of not causing a rear end collision with the vehicle in front was tested and this bought the collision percentage to below 10% as now the collisions that happen are only during lane changes. Hence if an intelligent collision avoidance system can be designed which can take the output from the driving agent and check if that could cause a collision or not before the vehicle performs that action, then the collisions can completely be avoided. Other than the high collisions, the driving algorithm worked as expected, with the autonomous vehicle being able to overtake the slow moving vehicles and also allowing the faster moving vehicles to overtake it by getting back to the driving lane after overtaking. Hence reinforcement learning can be used with cooperative driving to control the vehicle effectively, but another entity is required to prevent the vehicle from colliding with the surrounding vehicles.

The second goal of this research was to investigate the influence with change in the number of hidden layers and the number of nodes in a deep Q network on the performance of the driving algorithm. It was observed in this research based on the results given in Chapter 6.2, that there are no direct relations between the number of nodes and number of hidden layers of the neural network and the performance of the driving algorithm. In this research the best performance was achieved by using 3 hidden layers and between 1400 to 1700 nodes in each of the hidden layers. Having low number of nodes in each hidden layer gave a bad performance, with high number of collisions or the driving algorithm only trying to avoid collisions by not making any overtakes and travelling slowly.

The third goal of this research was to investigate the role of communication range on the performance of the driving algorithm. From the results in Chapter 8.2 it is found that the minimum communication range should be at least 200m, for the driving agent to take the correct decisions as for the communication range below this the collisions taking place were a lot higher.

The final goal of this research was to investigate the effect of packet loss due to communication error on the performance of the driving algorithm. Again based on the results in Chapter 8.2, loss of communication packets have a large negative influence on the performance of the driving algorithm. The collision percentage when the probability of packet loss was 0.5 during each time step was found to be more than 95% even when the communication range was greater than 200m. To solve this problem an error concealment technique was designed that used the information from the previous successfully received packet as the current packet in case of packet loss. By doing so the collision percentage was again brought down to be below 10% which is within the normal performance achieved in this research.

Having a higher level reward system for the driving algorithm helped in making the driving algorithm more generic which can be used in multilane highway environment as here the rewards are mainly based on the road rules like following the speed limit, avoiding an accident, overtaking from the left side and maintaining a certain gap from the vehicle in front, which are the same in all the highways.

Finally, it can be concluded that reinforcement learning using cooperative driving is a prospective approach for autonomous driving, but more research needs to be conducted to make sure that the actions resulting from the learning agent is does not cause any undesirable consequences. Reinforcement learning is mainly based on the reward system, as actions are taken to maximize the rewards. Hence having a good reward system is crucial for using reinforcement learning in autonomous driving.

9.2 Recommendations

In this research the rear end collision is avoided by SUMO. Hence no implementation for that had to be done in the driving algorithm. In the real scenario though a mechanism needs to be designed outside the driving algorithm that can monitor for possibilities of collisions and give outputs to the driving algorithm such that it can take the appropriate action to avoid the collision.

The driving algorithm can be made more efficient with the experience replay running parallel to the actual simulations. This would mean that the learning would be a lot faster, but achieving this design would be complex as both the experience replay and the simulations would modify the neural network. This would mean there would be a high chance of data sharing between the two threads which could lead to data corruption.

In this research the communication error was considered in the receiver side such that in case of error in communication, the autonomous vehicle would not receive information of all the surrounding vehicles. Another possibility is if there is a communication error in the transmission side, such that the autonomous vehicle would receive information of certain vehicles but not of all the vehicles surrounding it. Another problem related to communication could be, if the message being transmitted is corrupted or if there is a delay in receiving the message. The performance of the driving algorithm for these cases also needs to be tested.

The handling of communication error in this research was done by using the previous positions of the surrounding vehicles and using this positon as the new position of these vehicles in case of communication error. A better approach would be to calculate the new position of the surrounding vehicles based on their older positions and their current velocity and acceleration rates. As the communication is considered to happen during every time step, which is in every second. If the time when the successful communication occurred is known, the new position can be found based on the time elapsed. This would give more accurate information of the surrounding vehicle's position which is an important detail for avoiding collisions. This method of course would assume that the vehicle's velocity and its rate of acceleration are still the same. Hence this is useful, if the communication error is only for a few seconds as within this time there won't be a big difference in its positon.

The driving algorithm developed in this research only used the vehicle parameters of the immediate surrounding vehicles. Having the information of the vehicles in the next level, that is the surrounding vehicles of the surrounding vehicles to the autonomous vehicle might help the driving algorithm to get more knowledge about what is about to happen in the environment.

This research assumed that other surrounding vehicles are all manually driven and that the drivers drive perfectly without making any errors. Testing also needs to be done on how the driving algorithm behaves on imperfect driving of the other vehicles and how the performance would be if all vehicles in the simulation were also autonomous vehicles.

10. References

[1] "Autonomous Car," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Autonomous_car. [Accessed 19 06 2018].

[2] S. K. Gehrig and F. J. Stein, " Dead reckoning and cartography using stereo vision for an autonomous car," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Kyongju, 1999.

[3] A. Goodwin, "Audi AI, autonomy-ready," cnet, 14 05 2018. [Online]. Available: https://www.cnet.com/roadshow/auto/2019-audi-a8/preview/. [Accessed 19 06 2018].

[4] C. Fortuna, "Autonomous Driving Levels 0–5 + Implications," Clean Technica , 02 12 2017. [Online]. Available: https://cleantechnica.com/2017/12/02/autonomous-driving-levels-0-5-implications/. [Accessed 19 06 2018].

[5] "Functional block diagram of a typical self-driving car," O'REILLY, [Online]. Available: https://www.safaribooksonline.com/library/view/ros-robotics-projects/9781783554713/ch10s02.html. [Accessed 19 06 2018].

[6] Sproul, "Massachusetts has a BETTER & SAFER way!," Sproul Company, 27 03 2018. [Online]. Available: https://sproulco.com/2018/03/27/massachusetts-has-a-better-safer-way/. [Accessed 19 06 2018].

[7] "SUMO - Simulation of Urban MObility," DLR Institue of transportation Systems, [Online]. Available: http://www.dlr.de/ts/en/desktopdefault.aspx/tabid-9883/16931_read-41000/.

[8] S. A. Shota Ishikawa, "Cooperative Learning of a Driving Strategy to Suppress Phantom Traffic Jams," in *IEEE International Conference on Agents*, 2016.

[9] X. X. L. Z. Xin Li, "Reinforcment Learning Based Overtaking Decision-Making for Highway Autonomous Driving," in *Sixth International Conference on Intelligent Control and Information Processing*, Wuhan, 2015.

[10] S. Raschka, "Three Different types of Machine learning," KDnuggets, [Online]. Available: https://www.kdnuggets.com/2017/11/3-different-types-machine-learning.html. [Accessed 26 06 2018].

[11] "Unsupervised Learning," TechTarget, [Online]. Available: http://whatis.techtarget.com/definition/unsupervised-learning.

[12] A. Chandramohan, "Machine learning for Autonomous Driving (Research topics for Master Thesis)," 2018.

[13] A. G. Barto and R. S. Sutton, Reinforcement Learning: An Introduction, London: Bradford, 2015.

[14] C. J. Watkins and P. Dayan, *Technical Note Q-Learning,* Kluwer Academic Publishers, 1992.

[15] "Q learning," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Q-learning.

[16] E. E. Dar and Y. Mansour, "Learning rates for Q-learning," *Journal for Machine Learning Research,* no. 5, 2003.

[17] "Estimating the optimal Learning rate for a deep neural network," Towards Data Science, 12 11 2017. [Online]. Available: https://towardsdatascience.com/estimating-optimal-learning-rate-for-a-deep-neural-network-ce32f2556ce0. [Accessed 05 08 2018].

[18] "Reinforcement learning part 1: Q-learning and exploration," STUDYWOLF, 25 November 2012. [Online]. Available: https://studywolf.wordpress.com/2012/11/25/reinforcement-learning-q-learning-and-exploration/.

[19] S. D, "Teaching a Neural Network to play a game using Q-learning," Practical Artificial Intelligence, 04 09 2017. [Online]. Available: https://www.practicalai.io/teaching-a-neural-network-to-play-a-game-with-q-learning/. [Accessed 02 07 2018].

[20] "A beginner's Guide to Deep Reinforcement Learning," Skymind, [Online]. Available: https://skymind.ai/wiki/deep-reinforcement-learning#neural. [Accessed 2018 07 2018].

[21] R. Liu and J. Zou, "The Effects of Memory Replay in Reinforcement Learning".

[22] "SUMO," SUMO/DLR, [Online]. Available: http://sumo.dlr.de/wiki/SUMO. [Accessed 03 07 2018].

[23] X. Gao, *Deep reinforcement learning for time series: playing idealized trading games.*

[24] "Keras: Python Deep Learning Library," [Online]. Available: https://keras.io/.

[25] "TensorFlow," [Online]. Available: https://www.tensorflow.org/.

[26] "DLR," DLR, [Online]. Available: https://www.dlr.de/dlr//en/desktopdefault.aspx/tabid-10002/. [Accessed 29 06 2018].

[27] "NETEDIT," SUMO, [Online]. Available: http://sumo.dlr.de/wiki/NETEDIT.

[28] "Networks/SUMO Road Networks," SUMO, [Online]. Available: http://sumo.dlr.de/wiki/Networks/SUMO_Road_Networks#Normal_Edges. [Accessed 03 07 2018].

[29] "Networks/PlainXML," SUMO, [Online]. Available: http://sumo.dlr.de/wiki/Networks/PlainXML. [Accessed 03 07 2018].

[30] "Definition of Vehicles, Vehicle Types, and Routes," SUMO, [Online]. Available: http://sumo.dlr.de/wiki/Definition_of_Vehicles,_Vehicle_Types,_and_Routes#Car-Following_Models. [Accessed 03 07 2018].

[31] "Fundamentals of Transportation/Traffic Flow," wikipedia, [Online]. Available: https://en.wikibooks.org/wiki/Fundamentals_of_Transportation/Traffic_Flow#Time_headway. [Accessed 03 07 2018].

[32] "Lane-Changing Models," SUMO, [Online]. Available: http://sumo.dlr.de/wiki/Definition_of_Vehicles,_Vehicle_Types,_and_Routes#Lane-Changing_Models. [Accessed 03 07 2018].

[33] S. Krauß, "Microscopic Modeling of Traffic Flow:," Koln , 1998.

[34] "Repeated vehicles(Flows)," Sumo, [Online]. Available: http://sumo.dlr.de/wiki/Definition_of_Vehicles,_Vehicle_Types,_and_Routes#Repeated_vehicles_.28Flows.29. [Accessed 03 07 2018].

[35] "Flows with random number of vehicles," SUMO, [Online]. Available: http://sumo.dlr.de/wiki/Simulation/Randomness#Flows_with_a_random_number_of_vehicles. [Accessed 03 07 2018].

[36] "Configuration," SUMO, [Online]. Available: http://sumo.dlr.de/wiki/SUMO#Configuration. [Accessed 03 07 2018].

[37] "Collisions," SUMO, [Online]. Available: http://sumo.dlr.de/wiki/Simulation/Safety#Collisions. [Accessed 03 07 2018].

[38] "Time," SUMO, [Online]. Available: http://sumo.dlr.de/wiki/SUMO#Configuration. [Accessed 03 07 2018].

[39] "TraCI," SUMO, [Online]. Available: http://sumo.dlr.de/wiki/TraCI.

[40] "TraCI/Vehicle Value Retrieval," SUMO DLR, [Online]. Available: http://sumo.dlr.de/wiki/TraCI/Vehicle_Value_Retrieval. [Accessed 02 07 2018].

[41] "Getting Started with Gym," gym, [Online]. Available: https://gym.openai.com/docs/.

[42] Jonasschneider, "Creating new environments for Gym," GitHub, [Online]. Available: https://github.com/openai/gym/tree/master/gym/envs.

[43] L. Denoyer and P. Gallinari, "Deep Sequential Neural Network," 2014.

[44] "Getting Started with the Keras Sequential model," Keras , [Online]. Available: https://keras.io/getting-started/sequential-model-guide/. [Accessed 04 07 2018].

[45] S. Sharma, "Activation functions: Neural Networks," Towards Data Science , 06 09 2017. [Online]. Available: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6. [Accessed 04 07 2018].

[46] "TraCI/Vehicle Value Retrieval," SUMO, [Online]. Available: http://sumo.dlr.de/wiki/TraCI/Vehicle_Value_Retrieval. [Accessed 05 07 2018].

[47] "TraCI/Object Variable Subscription," SUMO, [Online]. Available: http://sumo.dlr.de/wiki/TraCI/Object_Variable_Subscription. [Accessed 05 07 2018].

[48] "TraCI/VehicleType Value Retrieval," [Online]. Available: http://sumo.dlr.de/wiki/TraCI/VehicleType_Value_Retrieval. [Accessed 05 07 2018].

[49] "TraCI/Lane Value Retrieval," SUMO, [Online]. Available: http://sumo.dlr.de/wiki/TraCI/Lane_Value_Retrieval. [Accessed 05 07 2018].

[50] R. Jain, in *The art of Computer Systems Performance Analysis*, John Wiley & Sons, INC.

[51] P. Mayfield, "Understanding Binomial Confidence Intervals," SigmaZone, [Online]. Available: http://www.sigmazone.com/binomial_confidence_interval.htm. [Accessed 07 06 2018].

[52] "Model Class API," Keras, [Online]. Available: https://keras.io/models/model/. [Accessed 04 08 2018].

Appendix A: Configuration Files

The SUMO configuration files used for this research are given in this Appendix.

Road Network file

```
<net version="0.27" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/net_file.xsd">


   <location netOffset="0.00,0.00" convBoundary="0.00,0.00,5000.00,0.00" origBoundary="-10000000000.00,-
10000000000.00,10000000000.00,10000000000.00" projParameter="!"/>


   <edge id="Lane" from="gneJ0" to="gneJ1" priority="1" length="40000.00">
      <lane id="Lane_0" index="0" speed="22.22" length="40000.00" shape="0.00,-4.95 5000.00,-4.95"/>
      <lane id="Lane_1" index="1" speed="22.22" length="40000.00" shape="0.00,-1.65 5000.00,-1.65"/>
   </edge>


   <junction id="gneJ0" type="dead_end" x="0.00" y="0.00" incLanes="" intLanes="" shape="0.00,-0.05 0.00,-
6.55"/>
   <junction id="gneJ1" type="dead_end" x="5000.00" y="0.00" incLanes="Lane_0 Lane_1" intLanes=""
shape="5000.00,-6.55 5000.00,-0.05"/>


</net>
```

Algorithm 3 The Highway Net file defining the read layout

Vehicle Route file

```
<routes>

<vType id= "Auto" accel="30.000000" decel="30.000000" sigma="0.500000" maxSpeed = "55.550000"
length="3.000000" color="0,0,255" vClass="passenger" minGap="0" tau="0.1"/>

<vType id= "Car" accel="30.000000" decel="30.000000" sigma="0.500000" maxSpeed = "11.100"
length="3.000000" color="255,0,0" vClass="passenger" minGap="10" tau="0.1"/>

<vType id= "FastCar" lcKeepRight="100" accel="30.000000" decel="30.000000" sigma="0.500000" maxSpeed =
"55.550000" length="3.000000" color="255,0,255" vClass="passenger" minGap="10" tau="0.1"/>

<route id="Straight" edges= "Lane"/>

<flow id="SlowCar" color="255,0,0"  begin="0" end= "200" probability="0.1" type="Car">

   <route edges="Lane"/>

</flow>

 <flow id="FastCar" color="255,0,255"  begin="0" end= "200" probability="0.01" type="FastCar">

   <route edges="Lane"/>

</flow>

<vehicle id="Auto" color="0,0,255" depart="60"  route="Straight" type="Auto"/>

</routes>
```

Algorithm 4 Vehicle Route File used without collision avoidance

SUMO Configuration File

```
<configuration>
  <input>
    <net-file value="StraightRoad.net.xml"/>
    <route-files value="StraightRoad.rou.xml"/>
                    <collision.action value="remove"/>
  </input>
  <time>
    <begin value="0"/>
  </time>
  <report>
    <verbose value="true"/>
    <no-step-log value="true"/>
  </report>
  <gui_only>
    <quit-on-end value="true" />
    <start value="true" />
  </gui_only>
  <random_numberType>
    <random value="true" />
  </random_numberType>
  <processing>
    <lanechange.overtake-right value="false" />
  </processing>
</configuration>
```

Algorithm 5 SUMO Configuration file

Appendix B: External APIs used

Details about the APIs from TraCI and Keras used in this research is given in this chapter.

APIs from TraCI

Table 8 below shows the APIs from TraCI used in this research for controlling the simulation environment and for accessing the vehicle information from the simulation and to give out the actions to the autonomous vehicle. The entire list of all the APIs available in TraCI can be found in [39]

S.No	Name of the Function	Description
1	traci.close()	Close interface with SUMO.
2	traci.start(sumoCommand)	Start SUMO and load the SUMO configuration.
3	traci.load(sumoCommand)	Load SUMO configuration.
4	traci.simulationStep()	Run one time step in the simulation.
5	traci.vehicle.getIDList()	Returns the ids of all the vehicles currently active in the simulation.
6	traci.simulation.getCurrentTime()	Returns the current simulation time in ms.
7	traci.vehicle.subscribe(vehID, parameters to subscribe)	The parameters subscribed by this command for the vehicle id is updated after each simulation step and can be got with the next function. The parameters subscribed are :- VehSpeed (m/s), VehPosition (in x,y coordinate), VehLaneIndex, VehTravellDistance (in m).

8	traci.vehicle.getSubscriptionResults()	Returns the latest values of the subscribed parameters of all the subscribed vehicles.
9	traci.vehicle.getSpeedMode()	Used to find the car following model, returns a 5 bit number
10	traci.vehicle.setSpeedMode(vehID, modeValue)	To set the car following model for the VehID. For disabling the automatic collision avoidance system, the 3rd and 4th bit of modeValue should be 0.
11	traci.vehicle.setLaneChangeMode(vehID, modeValue)	This is used for disabling automatic lane change of the autonomous vehicle by the simulator as this action needs to be done by the driving algorithm. For this the modeValue needs to be 0.
12	traci.vehicle.changeLane(vehID, laneIndex, time)	Moves the vehicle with id vehID to the lane given by laneIndex for the time provided. The time is given to be very long so that the vehicle remains in that lane until another action changes it to a different lane.
13	traci.vehicletype.getMaxSpeed(vehID)	Gets the maximum speed the vehicle can travel at.
14	traci.vehicle.slowDown(vehID, targetSpeed, time)	Accelerates or decelerates the vehicle to the target speed(m/s) in the time(ms) provided.
15	traci.lane.getMaxSpeed(laneID)	Returns the speed limit of the lane.

Table 8 APIs from TraCI used in this Research

APIs from Keras

Table 9 below gives the functions from the Keras library used in this research for the driving agent. The list of all APIs available in Keras is given in [52].

S.No	Name of The Function	Description
1	model.Add(LayerProperties)	This is used to add a layer to the neural network. The LayerProperties consists of the number of nodes in the layer, its activation method and the initial weights for the nodes.
2	model.compile(lossFunction, optimizer)	Configures the model for training.
3	model.predict(stateSet)	Generate the output Q values for each of the state from the current State.
4	model.fit(stateSet, TargetValues, epochs)	Trains the neural network for the given number of epochs.
5	model.load_weights(filePath)	Loads the weights of the nodes in the neural network from an existing file so that the agent need not be retrained during each simulation.
6	model.save_weights(filePath)	Save the weights of the nodes in the neural network to filePath.

Table 9 APIs from Keras used in this Research