Dissertation on

"Smart Traffic Light Controller using Deep Reinforcement Learning"

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology
in
Computer Science & Engineering

UE18CS390B – Capstone Project Phase - 2

Submitted by:

Krishna P Hegde	**PES1201801896**
Abhishek A	**PES1201801935**
Prathvik Nayak	**PES1201802006**
A Lakshmi Prasad	**PES1201802132**

Under the guidance of

Dr. Nagegowda K S
Associate Professor
PES University

June - December 2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

FACULTY OF ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled

'Smart Traffic Light Controller using Deep Reinforcement Learning'

is a bonafide work carried out by

Krishna P Hegde	**PES1201801896**
Abhishek A	**PES1201801935**
Prathvik Nayak	**PES1201802006**
A Lakshmi Prasad	**PES1201802132**

in partial fulfilment for the completion of seventh semester Capstone Project Phase - 2 (UE18CS390B) in the Program of Study - Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period June - December 2021. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 7th semester academic requirements in respect of project work.

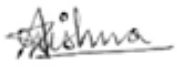
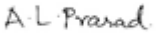
Signature Dr. Nagegowda K S Associate Professor	Signature Dr. Shylaja S S Chairperson	Signature Dr. B K Keshavan Dean of Faculty

External Viva

Name of the Examiners	**Signature with Date**
1. _____	_____
2. _____	_____

DECLARATION

We hereby declare that the Capstone Project Phase - 2 entitled **"Smart Traffic Light Controller using Deep Reinforcement Learning"** has been carried out by us under the guidance of Dr. Nagegowda K S, Associate Professor and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester June – December 2021. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

PES1201801896	**Krishna P Hegde**	
PES1201801935	**Abhishek A**	
PES1201802006	**Prathvik Nayak**	
PES1201802132	**A Lakshmi Prasad**	

ACKNOWLEDGEMENT

We would like to express my gratitude to Dr. Nagegowda K S, Department of Computer Science and Engineering, PES University, for his continuous guidance, assistance, and encouragement throughout the development of this UE18CS390B - Capstone Project Phase – 2.

We are grateful to the project coordinators, Prof. Sunitha R and Prof. Silviya Nancy J, for organizing, managing, and helping with the entire process.

We take this opportunity to thank Dr. Shylaja S S, Chairperson, Department of Computer Science and Engineering, PES University, for all the knowledge and support we have received from the department. We would like to thank Dr. B.K. Keshavan, Dean of Faculty, PES University for his help.

We are deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor – PES University, Dr. Suryaprasad J, Vice-Chancellor, PES University for providing to us various opportunities and enlightenment every step of the way. Finally, this project could not have been completed without the continual support and encouragement we have received from our family and friends.

ABSTRACT

Transportation has become a major priority for people nowadays. While technological advancements have made transportation easier, monitoring and controlling traffic as well as simulating it has become a significant challenge despite the significant research that has been carried out to tackle this problem. An emerging trend to solve this problem involves using deep reinforcement learning techniques which has shown significant progress and promising results in recent studies. There is an increasing demand for developing an intelligent traffic controlling agent which can dynamically adjust to real time traffic rather than just operating by hand-craft rules such as timers. We employ modern deep reinforcement learning methods such as Q-learning and deep neural networks with an agent-based traffic simulator SUMO ("Simulation of Urban MObility") which provides a synthetic yet realistic environment for simulating real-world like traffic and explore the outcomes of the actions that were performed on this environment. Additionally, we also aim to analyze the performance of the agent by comparing it to traditional traffic signals.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER-1

INTRODUCTION

We know pretty well for a fact that modern society depends heavily on transportation systems. Everyone desires to move from their origin to the destination in a smooth manner as possible, but due to the increase in population and the number of vehicles, traffic congestion, travel delay, accidents and unnecessary emissions have increased substantially and due to this there is a huge demand for improving the road infrastructure to ensure smooth flow of traffic.

"According to a report released by a major global location technology specialist, drivers in Bengaluru city have spent an average of 71% extra time on roads, while drivers in Mumbai spend 65% extra time in vehicles thanks to the city's traffic congestions."

The fact that the world is changing and developing very fast makes it a difficult task to control and monitor the traffic, especially in large metropolitan areas that are growing around the world such as Tokyo, Bengaluru, Los Angeles, Mumbai, etc.

Majority of the traffic congestions occur due to lack of capacity on the roads i.e. the number of vehicles that are travelling on the roads are exceeding their capacity. Apart from this, environmental issues, human errors, etc are other causes for traffic congestion.

Solving this problem of traffic control and monitoring and, in general, reducing traffic congestion still remains a major challenge in spite of substantial research that has been carried out to tackle this problem.

There are two main solutions to address this problem. First is to increase the road capacity so that more vehicles can travel at a given time avoiding unnecessary congestion. This method however, can be quite expensive. The second method is to improve the existing road infrastructure and traffic systems to ensure smooth flow of traffic without any unwanted congestion. Artificial Intelligence plays a key role in this. We attempt to try and develop a smart traffic signal controller that is able to analyse and adjust dynamically to real-time traffic rather than just following traditional hand-craft rules such as timers.

Reinforcement learning could be an optimal technique to solve this problem of traffic congestion and traffic control. Reinforcement learning is one of the ML techniques which comprises an agent, state representation, actions set, Q-function and a reward function.

The agent perceives the environment through its sensors and depending on that state of the environment, it performs a certain action. If the action taken was a good one, it gets a positive reward else it gets a negative reward. The agent has a goal which it achieves by maximizing the rewards.

The learning mechanism used here is deep Q learning, that is a mix of DNN and Q-learning. A form of model-free reinforcement learning technique which is used for determining the value of an action that was taken is Q-learning.

Q in Q-learning represents ‘quality’. This method involves assigning a Q-value or a ‘quality value’ to an action that was taken to alter the environment’s state.

CHAPTER 2

PROBLEM STATEMENT

Nowadays people spend a lot of time on the road due to Traffic congestion. Traffic congestion has become increasingly costly. According to reports, Bengaluru loses 5.92 Billion USD every year as a social cost of traffic congestion. So there is an increasing demand for controlling this traffic congestion in order for the smooth flow of traffic. One of the solutions for this is to increase the efficiency of the traffic signal controllers.

In order to increase the efficiency, the traffic signal controllers should intelligently analyze the traffic in each lane of an intersection and produce an optimal output in the form of choosing the correct traffic signal light phase, which would minimize the traffic congestion. “Reinforcement learning is a suitable technique for attempting to solve this traffic signal control problem, as it elegantly represents the elements of the problem - agent (traffic signal controller), environment (state of traffic) and actions (traffic signals).” Thus the origin of the product came into picture when this problem could be solved by Reinforcement learning.

Conventional traffic signals use a regular timer based approach which does not handle dynamic traffic conditions. Due to this, the vehicles have to wait for a long time even if the traffic density is less. In order to improve this, we present a Reinforcement Learning approach to control traffic signals so as to reduce traffic congestion in crossroads and intersections.

The scope of this project is to initially simulate traffic in a 4-way intersection and then extend it for multiple intersections using a traffic simulator(SUMO). We use an agent (traffic lights system) to perceive the environment and train this agent to self-learn using Reinforcement learning. “In order to design a system based on the Reinforcement learning approach, we define the state representation, the action set, the reward function and the agent learning techniques involved. The learning mechanism used here is Deep Q-Learning, which is a combination of two aspects widely adopted in the field of reinforcement learning which are deep neural networks and Q-Learning.” [1]

CHAPTER 3

LITERATURE REVIEW

3.1 "A Deep Reinforcement Learning Approach to Adaptive Traffic Lights Management" [1]

- Andrea Vidali, Giuseppe Vizzari, Stefania Bandini, Luca Crociani

3.1.1 Introduction

The paper presents a framework of Deep Reinforcement Learning for solving the problem of traffic congestion. It focuses mainly on two things, (i) the rules associated with traffic patterns (ii) analysis of scenarios based on agent-based simulations using a traffic simulator.

The simulations here are carried out on an agent-based simulator tool known as "Simulation of Urban MObility" (SUMO). SUMO provides an API for interfacing with external programs allowing the authors to define feasible sets of evident aspects of the surroundings, and also help in defining the rewards to the actions taken and controlling the traffic lights based on the decisions made by the learning agent.

3.1.2 Description of Reinforcement learning approach

In Reinforcement learning, we have an agent which perceives the environment's state at time t (S_t), and based on the environment's state, it performs an action a_t, resulting in the transition of the state of the environment to a new state(S_{t+1}). Once the transition is done, the agent gets a reward(r_{t+1}) which tells the quality of action (a_t) performed by the agent. A good action rewards the agent with a "positive reward" and a bad action rewards it with a "negative reward". The agent's goal is to maximize these rewards.

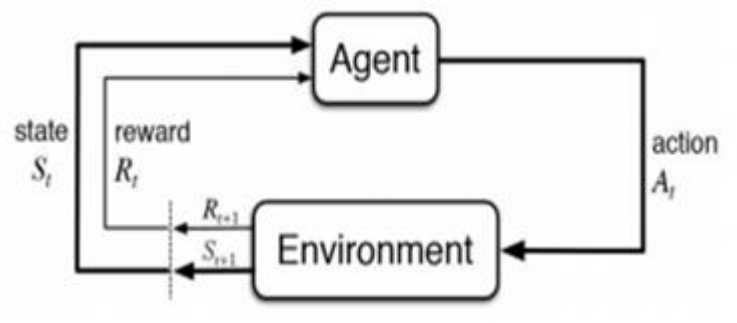


Fig 3.1 Reinforcement learning cycle

1) **State representation**: The approach proposed by authors in this paper is inspired by the DTSE (discrete traffic state encoding) where the agent perceives the state of the environment, and for every arm in the intersection, the incoming lanes are represented as cells which allows us to determine the presence or absence of vehicles in them.

2) **Action set**: This defines all the possible actions that the agent can take. Basically the action set is defined as a collection which incorporates all feasible actions that can be performed by the agent.
   Eg:- "A={NSA, NSLA, EWA, EWLA}, i.e. North south advance(NSA), North south left advance(NSLA), East west advance(EWA), East west left advance(EWLA)."[1]

3) **Reward Function**: It represents feedback from the environment for performing an action. A "positive reward" is the result of a good action whereas a "negative reward" is the result of a bad action. In this paper they have specified two different reward functions using slightly different traffic measures.
   - "literature reward function": here the "total waiting time" is used as a metric.
   - "alternate reward function": here the "accumulated total waiting time" is used as a metric.

4) **Deep Q Learning (DQN)**: This is the learning technique / mechanism adopted in this paper, which is a blend of Q-learning and DNN. It is a form of model free learning, which assigns a Q-value also known as quality value to an action that was performed on the state of the environment.

3.1.3 Training Process

The authors have proposed an experienced relay technique to enhance performance and efficiency. This technique consists of submitting all the knowledge needed by the agent for the training process as a group or collection of random samples also called as a batch rather than instantly submitting the knowledge gathered by the agent during the phase of simulation.
All the samples collected during the training phase are stored in the memory and a group of samples are retrieved from the memory as a batch.
Training includes iterative learning of a Q-value function with the help of knowledge available within the batch of samples retrieved from the memory.

For one sample the subsequent steps are performed

- Prediction of the Q-values $Q(s_t)$
- Prediction of the Q-values Q' (s_{t+1})
- Updating of Q (s_t, a_t) - Update the Q table with the quality of action(a_t) performed by the agent during simulation. The Q learning function is used to overwrite the values in this table.
- Training the neural network - Inputs to the neural network are the state set, and the output from the neural network is the updated Q - values for each action performed by the agent in that particular state. This helps us in selecting that action which gives the maximum reward.

The neural network also helps in sufficiently approximating the Q-learning function, once the function is approximated, the action with best Q value is selected for attaining the best traffic efficiency for a particular state.

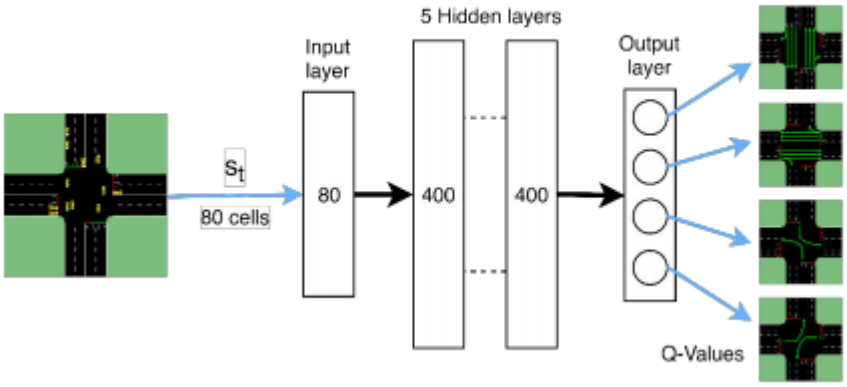


Fig 3.2 Idea of deep neural network

The action-selection policy is a big problem in reinforcement learning. We can either perform an exploratory action where the agent tries something new, or perform an exploitative action where the agent exploits what it already knows to achieve favorable rewards.

In this paper, the authors have chosen a greedy exploration policy.
For a current episode h, they define a probability ϵ to decide on taking an explorative action, and accordingly a probability of 1 - ϵ for deciding on an exploitative action.

$$\epsilon_h = 1 - (h/E)$$

Here E represents the total number of episodes and h represents the current episode of training. Initially when $\varrho = 1$, the agent only explores. But as the training process continues, the agent progressively exploits whatever it has learned till now, until it completely exploits.

3.1.4 Results

The agent's performance was mainly evaluated in two cases: (i) first, "the reward trend" was analyzed during the phase of training. (ii)The agent's performance was compared with traditional traffic signals, with the help of commonly used traffic metrics, like average waiting time per vehicle and cumulative waiting time.

- It was seen that the alternative reward agent performs much better in contrast to the literature reward agent: this is mainly due to the reward function it has adopted (accumulated waiting time) that reduces the waiting time which is exceeding the traffic light cycle, to a great extent.
- Taking into account only the waiting time ranging from the vehicle's last stop's, "results are not sufficiently emphasizing the usefulness of keeping longer light cycles, introducing too many yellow lights situations and changes, that are effective in low or medium traffic situations." [1]
- "The fact that the agent is simpler in low to medium traffic situations, results in the fact that a simple and almost immediate opportunity would be to separately develop agents dedicated to different traffic situations, having a kind of controllers/sensors that monitors the traffic flow and selects the foremost appropriate agent configuration." [1]

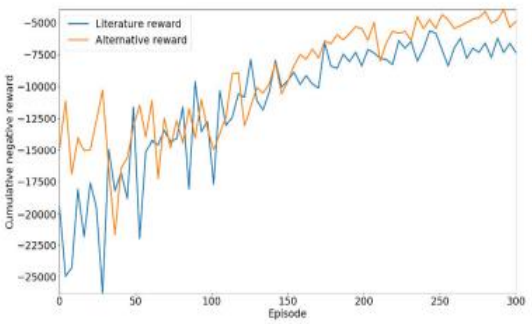


Fig 3.3 Cumulative negative reward per episode of agents during training (low traffic scenarios).

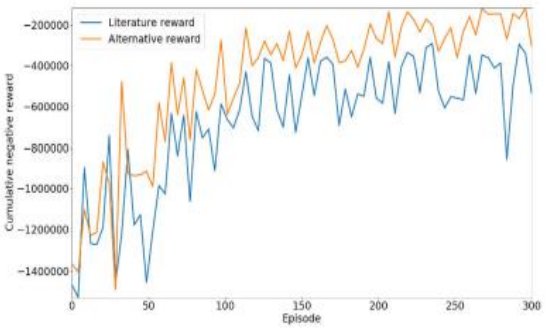


Fig 3.4 Cumulative negative reward per episode of agents during training (high traffic scenarios)

	Literature reward agent	Alternative reward agent
	Low-traffic scenario	
cwt	-30	-47
awt/v	-29	-45
	High-traffic scenario	
cwt	+145	+26
awt/v	+136	+25
	NS-traffic scenario	
cwt	-50	-62
awt/v	-47	-56
	EW-traffic scenario	
cwt	-65	-65
awt/v	-59	-58

Table 3.1 Agent's Performance overview

3.1.5 Conclusions and further developments

- The authors have utilized a simulator that provided a synthetic yet realistic environment for training the agent and evaluating its performance.
- Two reward metrics for agent's actions were assessed, showing that the correct use of proper ML techniques is as important as the correct description of the application context used, for attaining proper results.

- Future works include further improving of the obtained results, and also, improving within an extended term, at investigating what would be the implications of introducing multiple RL agents within a road network and what would be the likelihood to coordinate their efforts for achieving global improvements over local ones.

3.2 “Using a Deep Reinforcement Learning Agent for Traffic Signal Control” [2]

- Wade Genders, Saiedeh Razavi

3.2.1 Introduction

The paper has proposed an agent for traffic signal control using a deep artificial neural network. And to train this deep artificial neural network, they used reinforcement learning. An optimal control policy has been developed that makes a great effort to solve the problem of traffic signal control.

The traffic control problem has been defined as, what is the best phase of the traffic signal and its sequence that must be utilized given the current traffic condition at an intersection also called its state. The authors have developed an agent for traffic signal control using a deep Q-network, and they used a CNN that was deep to model the function for action-value and reinforcement learning technique was used to train it in an open source traffic simulator software called SUMO, using an intersection which is isolated.

They have proposed a new definition for the state space, which is the DTSE that stands for discrete traffic state encoding, and when compared to the state space definitions from previous research it contains more relevant information as it is an improved representation of traffic.

According to the authors, the convolutional neural network will be able to perceive information about the traffic that is more relevant with the help of the DTSE than previous research. Which can further be used to extract features that are useful and to develop state representations of high level. An optimal control can be achieved by the agent by picking the actions having the maximum value of the cumulative reward that was expected.

3.2.2 Reinforcement Learning components used in the paper

The problem of controlling the traffic signal can be solved by making use of reinforcement learning, and hence it is required that the problem be formulated in the reinforcement learning language. And this is done by defining the reward, the space of actions and the space of the state.

State Space:

The state space used in this research is DTSE. The DTSE divides a segment of lane of length, say l, which begins at the stop line for each lane in the intersection into cells of length c. The DTSE consists of three vectors. In which the first vector represents the cell if there is a vehicle present in it or not, the next vector represents the vehicle speed and the third vector represents the phase of the traffic signal.

(b)

0	0	0	0	0	0	0	0	1	0	0	1
1	0	0	0	1	0	0	0	1	0	1	1
0	1	0	0	0	0	1	0	0	1	0	1

(c)

0	0	0	0	0	0	0	0	0.2	0	0	0
0.8	0	0	0	0.5	0	0	0	0	0	0	0
0	0.6	0	0	0	0	0.4	0	0	0	0	0

Fig 3.5 (a)fig. of simulated traffic (b) representation in Boolean values
(c) real-valued DTSE vectors.

Action Space:

All the actions that are available are contained in a set from which the agent must make use of one action after it has observed the environment and its state. The possible actions of the agent are taken as the traffic signal phase configurations in this research, which is the combination of traffic lights that control the entire intersection for each of the individual lanes. The possible actions are considered as the following: Green in the West and East, Green in the South and North, Green in the West and East Advance Left and Green in the South and North Advance Left.

Reward:

After the state of the environment has been observed by the agent and it has chosen an action based on it, it receives a reward. A reward is an outcome of performing in the specific state a selected action. The change that is caused by the collective delay in the vehicles in between the actions has been defined as the reward in this research. This enables the reward to become negative or positive which means that the agent will be rewarded or punished.

Agent:

An entity that learns from the environment by interacting with it is called an agent. The agent has been modelled as a deep convolutional Q-network that controls the traffic signals.

This paper has used Q-Learning which is a reinforcement learning algorithm, which has been used for the development of an action selection policy that is optimal.

		Selected Action			
		NSG	**EWG**	**NSLG**	**EWLG**
Current Traffic Signal Phase	**NSG**	-	{NSY, R}	{NSY}	{NSY, R}
	EWG	{EWY, R}	-	{EWY, R}	{EWY}
	NSLG	-	{NSY, R}	-	{NSY, R}
	EWLG	{EWY}	-	{EWY. R}	-

Table 3.2 Traffic Signal Phase Action Transitions

3.2.3 Experimental Setup

A traffic microsimulator called SUMO was used to conduct all the experiments. For the implementation of the DQTSCA they have used the API for SUMO python along with a custom code. They have implemented the artificial neural network by making use of Python libraries like Theano and Keras. They have made use of libraries like SciPy and NumPy for more additional optimized functionality.

For the parameters of the DTSE, they defined c to be 5 m and l to be 75 m. They trained for 1600 epochs of training, where each of the epochs is around 1.25 hours of the traffic that is simulated. They executed the simulations on their desktop computer having a CPU with specifications as 3.45 GHz i7-2600. It was run on Ubuntu 14.04 with 8GB RAM. The actions of the agent are of length two

seconds and the transition phases are of five seconds.

The geometry of the intersections is four lanes that the intersections approach from the direction of the compass which are East, West, North and South, and are connected to four of the lanes that are outgoing from the intersection. For each approach the movements of the traffic are the following: the two lanes in the middle are for straight lanes, for taking left turns the lane in the inner part is used and the outer lane is for straight and turning right only. From the origin of the vehicle to the stop line of the intersection, the length of all lanes are 750 meters.

The quality of any traffic simulation is greatly influenced by the methods using which the vehicles are released and generated. In this research, for generating traffic that flows right and left the use of Inverse Weibull distribution has been done, and for generating the traffic that flows through the Burr distribution has been used.

3.2.4 Training

The agent is trained using experience replay which is a biologically inspired process. The agent stores all the experience for periodic and also for randomized batch training in a memory of experience instead of having to train after each of every individual action, state, reward and sequence of state. The training pseudocode is presented in the paper. The multithreading capabilities of modern computers are taken as an advantage in this research, as multiple traffic simulations can be run in parallel.

To make a comparison against their proposed DQTSCA, they also developed a neural network that is shallow TSCA. They used one hidden layer with 64 neurons for their STSCA that stands for traffic signal control agent that is shallow using the sigmoid activation function and for its output layer they used four neurons with linear activation functions.
Two vectors, the first of which contains the number of vehicles that are described as elements that are queued at the approach of each of the intersection which can be East, West, North and South, and the second vector represents the phase vector of the present traffic signal; these constitute the state space of the STSCA.

They used the same action space and the reward as the DQTSCA. They have done the training for the STSCA using the same gradient descent algorithm as the DQTSCA, number of epochs and action

selection policy. However, the use of experience replay is not done in the traditional agent, after every sequence of the state, action and the reward it trains

3.2.5 Results

They assessed the DQTSCA that was proposed for its performance in terms of traffic measures that are common which are: throughput, travel time, queue length and cumulative delay.

The STSCA was compared with the DQTSCA that was proposed. It was observed that the DQTSCA achieves a reduction of 82% than that of the STSCA in the cumulative delay when averaged. This provides confirmation that the control policy learned by the DQTSCA is much better than that of the STSCA because of the difference in this key metric.

When the other traffic metrics were compared, it was found that in the throughput there was almost no difference, but the DQTSCA was found to reduce the average time of travel by at least 20% and the average length of the queue by almost 66% when compared to that of the STSCA. In three of the four metrics, the STSCA has been outperformed by the DQTSCA, mainly because of the utilization of the DTSE and having an architecture that is deep.

Traffic Metric (μ, σ, $n = 100$)	**STSCA**	**DQTSCA**
Throughput (Vehicles)	(2 452, 257)	(2 456, 248)
Queue (Vehicles)	(33, 23)	(13, 9)
Travel Time (s)	(197, 107)	(157, 49)
Cumulative Delay (s)	(4 085, 5 289)	(719, 1 048)

Table 3.3 STSCA and DQTSCA Traffic Metrics

3.2.6 Conclusion

To conclude the authors have proposed, developed and tested an agent for controlling traffic signals using the DQTSCA which is a deep Q-network on an open source traffic simulator software. The results obtained from the research conducted has shown that when compared to methods that are traditional, improved performance can be achieved to fix the issue of traffic signal control by applying deep learning.

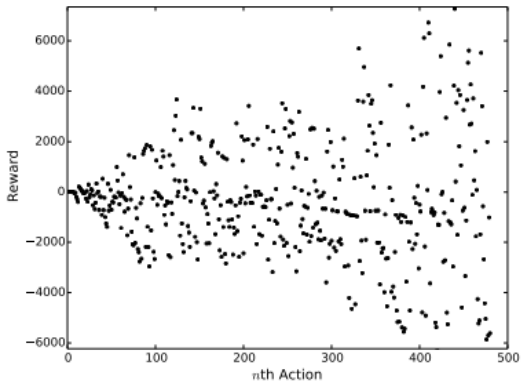


Fig 3.6 Reward in an epoch of DQTSCA while considering exploratory action only, in the early phases of training.

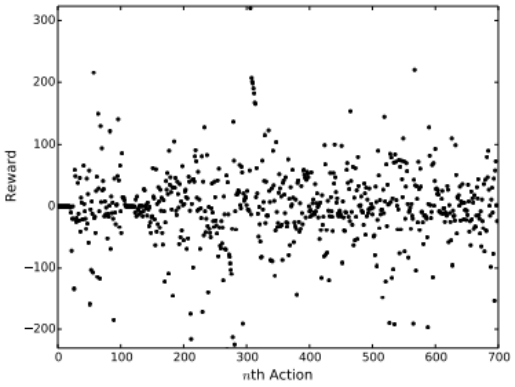


Fig 3.7 Reward in an epoch of DQTSCA while considering exploitative action only, after training

The agent's control can be extended to all of the traffic signals, which includes the red as well as yellow phases of the traffic light phase, in the future. The agent does not have the ability to take control of the red as well as yellow phases currently; they exist only in the sequences of transition in between the actions of the agent.

3.3 “IntelliLight: A Reinforcement Learning Approach for Intelligent Traffic Light Control” [3]

- Hua Wei, Zhenhui Li, Huaxiu Yao, Guanjie Zheng

3.3.1 Introduction

The authors of the paper propose a Deep Reinforcement Learning approach along with three more optimized approaches which include the Memory Palace and Phase Gate.

Traditional reinforcement learning proposes two main challenges (1) representation of the environment and (2) modelling how the environment and decision are correlated. Recent studies have used Deep Q-learning (DQN) which is a type of deep reinforcement learning technique to overcome these challenges.

3.3.2 Design of the Agent

1. State: State is defined for intersections. Every lane ‘i’ has queue length, number of vehicles, waiting time of vehicles updated as the state component. Also, the state has a picture portrayal of vehicles position, the current stage Pc and the following stage Pn.
2. Action: Action can be denoted with a bit, if the bit is 1, then the light changes to the following phase Pn, and if the bit is 0, then the present phase Pc is kept.
3. Reward: Each of the following is given a weight and the summing the product we get the reward

   (1) Summation of queue length of the approaching lanes. It is determined by adding all the number of vehicles that are delayed on all the approaching lanes.

   (2) Summation of delay D of every advancing lane. The delay Di for the ith lane is given in Eq.1, the mean speed of every vehicle in the ith lane is considered as the lane speed.

   $$Di = (\text{speed limit} - \text{speed of the lane}) / \text{speed limit} \quad \text{---------- (1)}$$

   (3) Summation of improved waiting time of all the approaching lanes. Equation 2 represents the improved waiting time W for vehicle i at time t. The improved waiting time of a vehicle becomes null after the vehicle advances every time.

   $$Wi(t) = Wi(t-1) + 1 \quad \text{speed of the vehicle} < 0.1$$
   $$Wi(t) = 0 \quad \text{speed of the vehicle} \geq 0.1 \quad \text{---------- (2)}$$

(4) Indicator of light switches (C): This can also be denoted by a bit, if the bit value is 0 then the present phase is kept, and if the bit value is 1 when the present phase is changed.
(5) After the last action A, during the time interval Δt, all the vehicles N that travelled through the crossing.
(6) Following the previous action A, during the period Δt, the total journey time of vehicles T that travelled through the crossing.

$$\text{Reward} = (w1 * \sum Li) + (w2 * \sum Di) + (w3 * \sum Wi) + (w4 * C) + (w5 * N) + (w6 * T) \quad \text{---- (3)}$$

Thus, when given the present state S of the traffic condition, the agent's task is to seek out the action A which will cause the utmost reward R within the end of the day, following the Bellman Equation.

3.3.3 Experiment

They have conducted tests using both the manufactured data and the data from real world traffic.

1. Experiment Setting

SUMO, a simulation platform is used during the conduction of the experiments. SUMO has a feature where the traffic moving consistently can be controlled with the given policy of traffic signals which is gained by the traffic signal agent. On synthetic data, four-way crossing is the environment for the experiments. The crossing is connected with four road sections, where each road has three incoming and outgoing lanes.

2. Evaluation Metric

Evaluation of the performance is done using the following measures:

1. Reward: Since the range of rewards is from $-\infty$ to ∞, from equation 3, there can be a peak for the reward at the point when every one of the vehicles moves uninhibitedly. Thus the average reward over time can be an evaluation metric.
2. Queue length: Length of the queue at a time t is the summation of all the vehicles waiting over all advancing lanes. Lesser the length of the queue, less will be the waiting vehicles on every lane. Hence, average of the queue length can be an evaluation metric.
3. Delay: Lower the delay, higher the speed of all lanes. Average of the delay over time, where the delay at time t is the sum of D from Eq.1 of all advancing lanes can be an evaluation metric.

4. Duration: This is one of the foremost key measures that people worry about while driving. Smaller the duration, time wasted by the vehicles passing through the crossing is less. Thus, average time period vehicles wasted on approaching tracks can be an evaluation metric. A far better reward shows a finer execution of the tactic, and a lesser length of the queue, duration and delay shows that the traffic is a smaller amount congested.

The experiment is performed on two types of datasets: (1) Synthetic data and (2) Real world data

Different execution figures are created by contrasting techniques. In true information, perceptions with peak hour versus non-peak hour; work day versus end of the week.

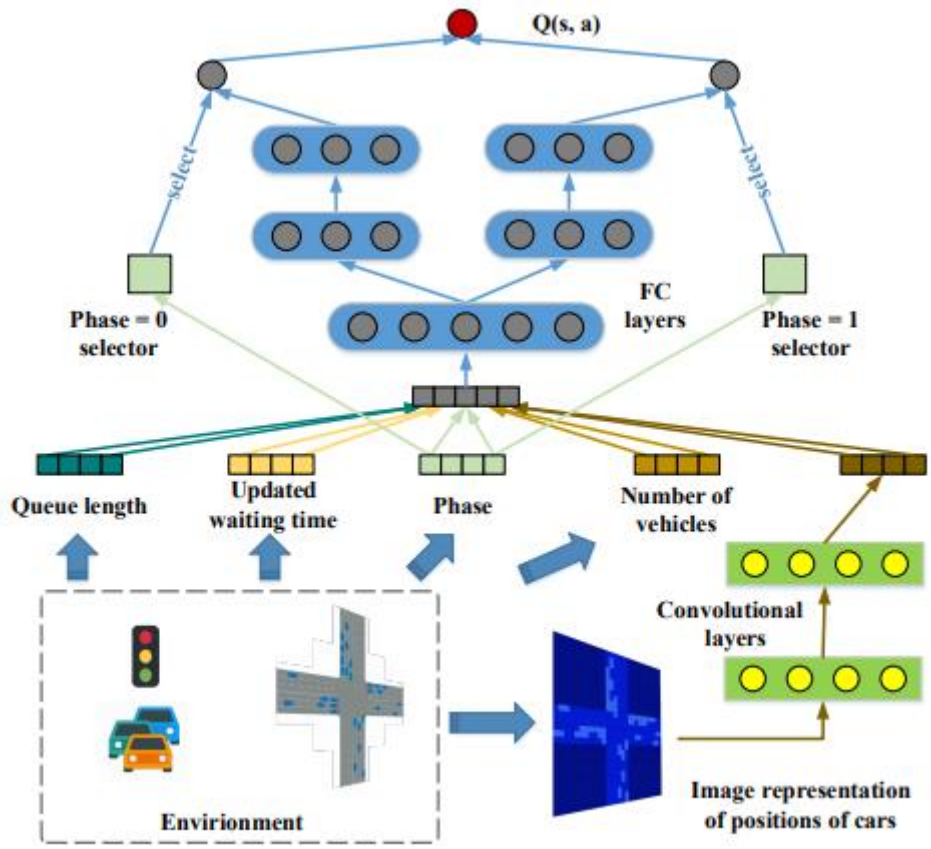


Fig 3.8 Q-network

3.3.4 Conclusion

- In the paper, they have addressed the traffic light control issue by utilizing a very much planned RL approach. They have conducted extensive tests using both the manufactured and the real world data and showed the best performance of suggested strategies over modern methods.
- They also acknowledge the restrictions of current approaches and have pointed out several key future alterations to the technique to make it further useful in the real world.
- First, the two-stage traffic signal can be stretched out to multi-stage traffic signals, which will include more muddled but more sensible state transitions.
- Second, the road network in the real world is much more complex than the simple one intersection that the paper has addressed.
- Lastly, the feedback is simulated since the proposition is analyzed on a simulation framework.
- At last, there is a need for field study to become familiar with this present reality criticism and to approve the recommended reinforcement learning approach.

3.4 Reinforcement learning-based multi-agent system for network traffic signal control [4]

- Arel C. Liu1 T. Urbanik, A.G. Kohls

3.4.1 Introduction

A basic challenge in traffic designing tests is the planning and the handling of multi-crossing point organizations. Customary traffic signal systems fail to handle traffic issues, to a great extent because of the absence of a good reward policy

In a multi-crossing point traffic organization, clogging in a solitary path doesn't just affect upstream traffic, yet in addition different convergences. Hence, a proficient strategy that can boost the reward and which could handle traffic issues is exceptionally wanted.

Here, we introduce an answer for traffic issues by utilizing RL–a machine learning system that endeavors to rough an ideal dynamic strategy.

We use a multi-agent setup, whereby RL is utilized as methods for handling the various convergences in the traffic system. In rundown, this calculation is created in view of the traffic plan for an artificial intelligent acquiring way, as opposed to the ordinary traffic system.

3.4.2 Adaptive signal control systems

Here, notable frameworks incorporate SCOOT [9] and SCATS. SCOOT is unified framework that consistently gauges traffic volumes and focuses to handle the signal timings with the essential goal of limiting the traffic issues in a particular region.

A heuristic improvement assesses potential planning plans changing the traffic signal timings. In bigger organizations, the framework characterizes more modest regions for demonstrating and advancement purposes.

SCATS doesn't need demonstration. It is a computerized, continuous, traffic responsive sign control procedure utilizing neighborhood regulators and provincial PCs for strategic and key control. Steady observing of traffic permits the framework to pick the proper sign timings, boosting throughput

Every agent faculties and controls the factors of its subsystems, while speaking with specialists nearby to acquire neighborhood factors and arrange their activities.

3.4.3 Definition of the RL elements

A RL issue is characterized once states, actions, rewards are plainly characterized. Keeping that in mind, we further depict this fundamental builds with regards to our problem domain

System states: At every simulation time step, local traffic statistics influences the local state of an intersection. We try to use an eight-dimensional feature vector for indicating each state with each element indicating the relative flow of traffic at any one lane. Relative traffic flow can be measured by dividing the total delay of vehicles with the average delay.

However, a central junction agent can have access to all its neighboring junction states, thus such extra data permits it to anticipate upstream traffic streams, subsequently increasing the performance.

Reward function: For the two kinds of intersection agent, a reward is given to a convergence agent after executing a given activity. The prize has a value ranging from -1 to 1, where a positive reward will be issued to agent if the delay is low compared to that of the previous interval

In this case, the local reward is directly having an on effect on vehicle volume and delay at a given intersection, which can be defined as follows

$$r = Dlast - D\ Current / \max[Dlast, Dcurrent]$$

Here D current and D last are the current and previous intersection total delay, respectively.

By taking into account, the way that the conduct of the central junction agent has an impact on

other convergence in the organization, we try to consider delay info of other junctions while calculating the reward of the central junction

Vehicular sensors and traffic controllers can be used to measure vehicle delay. To track vehicular arrivals at a given junction we can use Advance detector actuations.

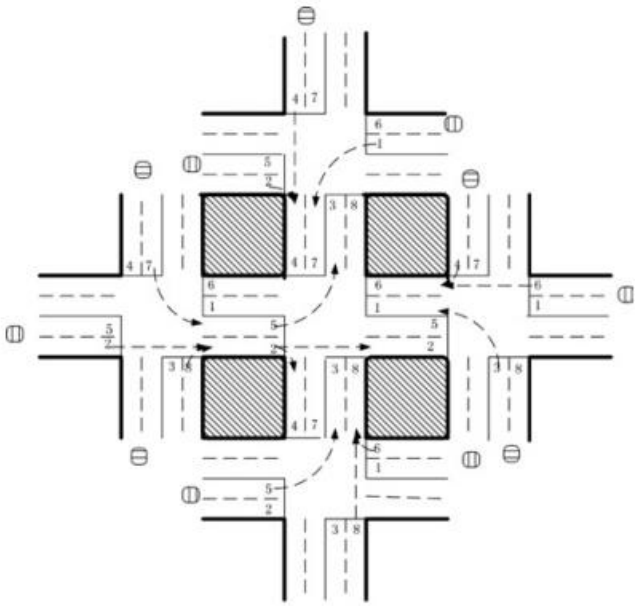


Fig 3.9 Five-intersection traffic network.

Simulation results: They executed all the simulations in the Matlab environment. One-time step is referred to as one-time unit in this considered environment. one discrete second is referred to as one-time step. Once every 20 units an action is taken, only one vehicle is permitted to cross a junction at a given interval.

3.4.4 Results and discussion

- Good and effective results were observed compared to that of the traditional traffic system
- Upon issuing an action, a positive reward will be issued to central intersection agent if the delay is low compared to that of the previous interval
- If the delay is high when compared to that of previous interval, a negative reward will be given to the central intersection agent

3.4.5 Conclusions

- This study presented a multi-agent framework and RL-based system in order to organize signal lights at junction points
- The basic goal is to increase expected reward points which directly influences in reducing traffic issues.
- Thus calculation could be promptly handle more intersections without making much changes to center design
- The approach mentioned was created while seeing the traffic issue artificial intelligence task, indicating a shift from traditional traffic scheduling issue formulations
- Further, creators mean to broaden the observations to incorporate extra execution measurements, like likelihood of stopping.

CHAPTER 4

PROJECT REQUIREMENTS SPECIFICATION

Product Features

- The agent can detect traffic from the intersection.
- Based on the state of traffic the agent takes an appropriate action to minimize traffic congestion.
- The agent uses a method of Deep-Q learning for training.
- The reward function rewards the agent for every good action it takes and penalizes it for every bad action it takes.
- The agent minimizes the traffic congestion and the waiting time for vehicles.
- Performance analysis of the agent when compared to the conventional traffic signals.
- Performance analysis of the agent for different reward functions.

Operating Environment

- Operating System: Windows 7+, MacOS version 10.7+, Ubuntu 10+, with RAM of 8GB or above.
- Simulation: Simulation of Urban MObility (SUMO).

General Constraints, Assumptions and Dependencies

- This simulation should be done only in SUMO software and not any other simulator software.
- Since this is a simulation, it may not be very precise compared to real world traffic.
- The TraCI interface is used to interact with SUMO using python.

Risks

- Training the agent could be a very tedious job as we will need to run a large number of simulations which is time consuming.
- Extending to multiple intersections could be challenging and complex as we will need to have proper communications between agents at different intersections for smooth flow of traffic.
- Since this is a simulation, it may not be very precise.

Functional Requirements

- Generation of road network and flow network in SUMO simulator and creating simulations.
- The software should simulate heavy traffic for training the agent.
- Training the agent (traffic controller) with the simulations.
- Testing the agent with new simulations.
- The agent must be able to minimize traffic congestion as much as possible to allow smooth flow of traffic.
- Performance analysis of the agent for different reward functions.
- Performance analysis of the smart traffic controller when compared to conventional traffic signals.

External Interface Requirements

User Interfaces

- SUMO traffic simulator for simulations.
- SUMO provides us with an editor to edit the road infrastructure, an API and an interface to view the simulations.
- It allows the user to design, configure and implement the road network infrastructure and exchange data while running simulations.
- A road network to see the structure of the road and a flow network to see the flow of traffic.

Hardware Requirements

Hardware requirements for this product can be considered as just the computer system. Further developments in this product with the real world implementation can include different sensors to scan the traffic in each lane and the traffic lights coordinated with it.

Software Requirements

- Github - for maintaining version control.
- OS - Windows 7+, MacOS version 10.7+, Ubuntu 10+, with RAM of 8GB or above.
- SUMO software - 1.8.0 or above.
- netedit software - 1.8.0 or above for creating road and flow networks.
- Development tool - Google colab / jupyter notebooks, Tensorflow, Keras.
- Language - python3 and above.
- Design tool - StarUML

Communication Interfaces

- TraCI(Traffic Control Interface) - TraCI allows us to control a running simulation.
- TraCI follows a client server architecture based on TCP, where the controller (external script) acts as a client and SUMO software acts as a server.
- The "controller" is an external python script which gets the simulation state information from the server (SUMO) and then responds by sending instructions.

Non-Functional Requirements

Performance Requirement

- Reliability - The agent should be able to minimize traffic congestions for various simulations.
- Robustness - The agent should be able to perform this task for a long period of time.
- Compatibility - The program should be compatible with different versions of SUMO software.
- Maintainability - Presence of appropriate level of documentation as well as well commented and easy-to-read code for easy maintenance and bug fixing (if any) in the future.

Safety Requirements

- The RL model should be capable of analyzing the intensity in each traffic lane to ensure maximum efficiency.
- The sensors when implemented in the real world should be able to correctly produce the inputs so that the model can produce better outputs.

Security Requirements

With the implementation in the real world, the product can generate many security vulnerabilities. The product should be able to send a report immediately if the traffic light is not changing lights for more than a certain set limit. The product should also be able to receive feedback from the commuters when there is any discrepancy.

Other Requirements

If we need to implement our project in real life in real time we need many other requirements such as to keep track of the number of vehicles at each interaction at a given time, and to compute it and produce results in real time.

CHAPTER 5

SYSTEM DESIGN

5.1 Deep Reinforcement learning framework

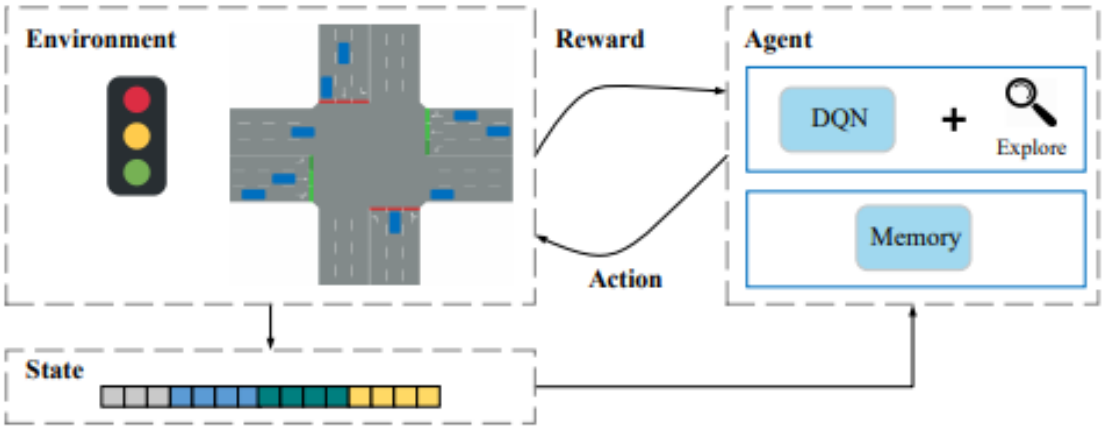


Fig 5.1 Deep reinforcement learning framework

The above figure shows a basic framework for deep reinforcement learning. Environment comprises traffic light phases as well as the traffic conditions. State depicts the feature of the environment's representation. The state of the environment is given as an input to the agent, the agent uses this information to learn the framework to decide as to "keep the current phase of traffic lights" or "change the current phase". The agent's decision is forwarded to alter the environment's state and the environment sends a feedback to the agent. Based on the feedback the agent updates the model it has learnt and further settles on the new choice for the following timestamp dependent on the new environment's state and the improved model.

In this framework, traffic conditions are depicted as an image, and the image is given as an input directly to the CNN model to improve the features of the environment. Ongoing deep RL approaches have shown promising results for controlling traffic congestion and enabling smooth flow of traffic.

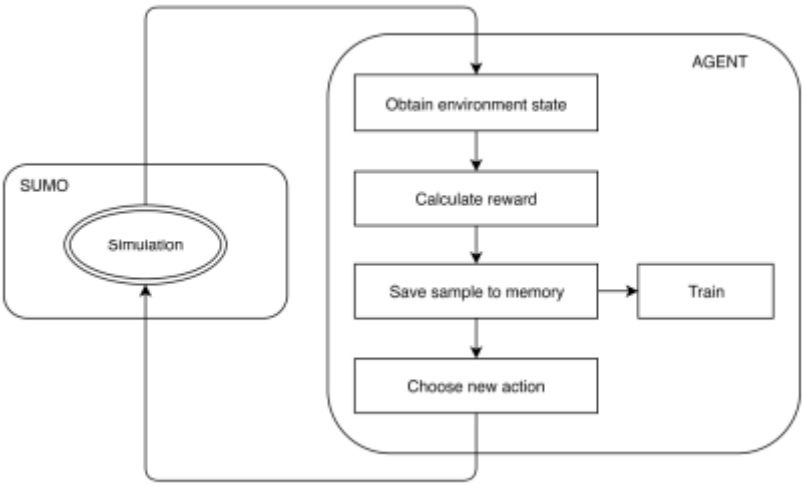


Fig 5.2 Agent workflow

The framework above shows a simple state diagram workflow of an agent, where it perceives the environment state from the simulation, and saves it in memory for training using DQN. Based on the environment's state it chooses the steps to alter the environment's state.
We can also see that once the environment has changed, it calculates the reward based on the previous action taken.

5.2 Design Constraints, Assumptions, and Dependencies

- This simulation should be done on SUMO software.
- Since this is a simulation, it may not be very precise compared to real world traffic.
- The TraCI interface is used to interact with SUMO using python.
- Training the agent could be a very tedious job as we will need to run a large number of simulations which is time consuming.
- Extending to multiple intersections could be challenging and complex as we will need to have proper communications between agents at different intersections for smooth flow of traffic.
- Further implementation in the real world could be a complex process due to increase in resources such as sensors to scan the traffic.
- The sensors when implemented in the real world should be able to correctly produce the inputs so that the model can produce better outputs.

5.3 Design Description

- **Traffic generation**
  - This module is responsible for generating traffic and creating a route file for the SUMO simulator. Each call to this module will generate random traffic simulation using Weibull distribution.
- **Model creation**
  - This module is responsible for the creation of deep neural network models for training and predicting the outputs. We will be using 2 different models, Artificial neural networks and Convolutional neural networks.
- **Simulation and training**
  - This class handles the simulation part. Here we have a function called 'run' that simulates one episode. There are also other functions being used during 'run' to interact with SUMO, eg: getting the state information from environment using get_state, setting the next green light phase using _set_green_phase and preprocessing the data in order to train the neural network using _replay. The two files training_simulation.py and testing_simulation.py are similar but just have a slightly different Simulation classes. Depending on whether we are training or testing, we load the necessary file.
- **Memory**
  - This module handles the memorization for the experience replay mechanism. It has multiple functions to add samples to memory and retrieve a batch of samples for training.
- **Utils**
  - This file contains directory related functions such as loading models for testing and creating different versions of trained models, importing configuration settings, etc.
- **Testing**
  - This module is responsible for testing the model with a single test episode running a completely different and random simulation with emergency vehicles configured in such a way that they can ignore the red lights. We will be testing both the models (ANN and CNN) for multiple intersections as well.

- **Visualization and analysis study**
    - Once the testing is done we will be performing an analysis study to see the performance of the agent in both the models compared to normal traffic, to study the reward trend, and see the agent's behavior in different levels of traffic.

5.3.1 Master Class Diagram

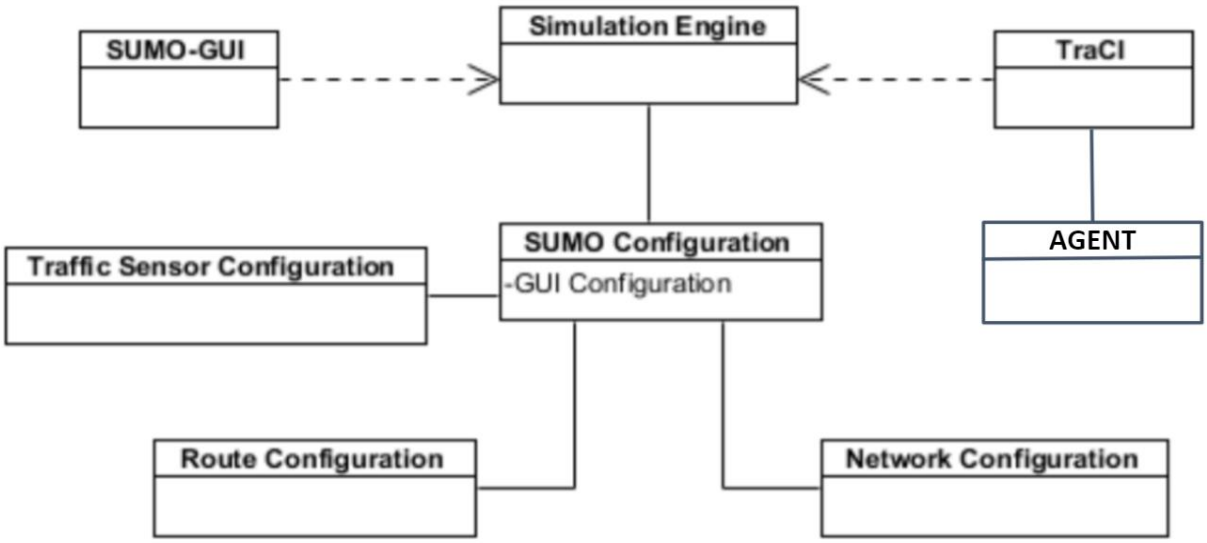


Fig 5.3 Master Class Diagram

5.3.2 Traffic generation

5.3.2.1 Description

- The road network structure(environment.net.xml) and the initial traffic light sequence with durations for both single and double intersection environments was created using netedit software.
- The traffic on these networks was generated using a python script(generator.py).
- The generator.py file takes the parameter 'no of cars per episode' and generates traffic with all kinds of vehicles. You can even pass no of high priority and low priority vehicles.
- The generation of vehicles is distributed according to Weibull distribution.

5.3.2.2 Class Diagram

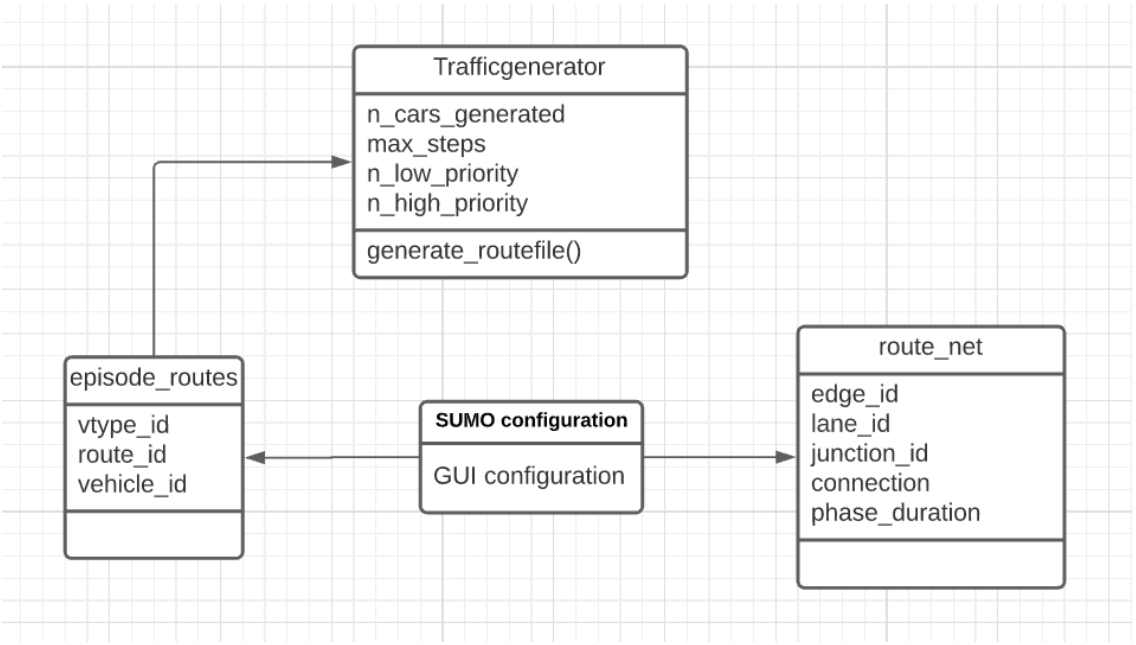


Fig 5.4 Traffic generation Class Diagram

5.3.2.3 Traffic Generator

- This class has attributes such as the number of cars that needed to be generated, max_steps, and no of high and low priority vehicles. Using these attributes it generates a route file using the generate_routefile() method.
    - ❖ generate_routefile()-This method creates a route file and generates different route id's for the simulation using Weibull distribution.

5.3.2.4 SUMO Configuration

- This class represents the main configuration file which makes use of both the route file and the net file and the SUMO gui to run the simulation.

5.3.2.5 episode_routes

- This class represents the file having all the attributes such as vehicle id, vehicle type, route id, etc. All these attributes are used to generate different routes.

5.3.2.6 route_net

- This class represents the file having all attributes such as edge id, lane id, junction id, etc. All these attributes are used to generate the network configuration and the different traffic light phases and its durations.

- **Data members of all classes**

Data Type	Data Name	Description
int	n_cars_generated	no of cars to be generated in the simulation.
int	max_steps	Each episode duration, 1 step equals 1 second.
int	n_low_priority	no. of low priority vehicles.
int	n_high_priority	no. of high priority vehicles.
string	vtype_id	type of vehicle.
alphanumeric	vehicle_id	vehicle id for a particular type.
string	route_id	route id for each route.
alphanumeric	edge_id	edge id in the network.
alphanumeric	lane_id	lane id in the network.
string	junction_id	junction id in the network.
int	phase_duration	duration of each phase in a traffic light.

Table 5.1 Data members of all classes in Traffic generation

5.3.3 Model creation

5.3.3.1 Description

We will be creating two models.

- The first model consists of a fully connected ANN with 80 neurons in the input later, 5 hidden layers with 400 neurons each with relu activation function and an output layer of 4 neurons with a linear activation function.
- For the second model we will be using a CNN, where the input will be two matrices (position and velocity) obtained using DTSE. We will be performing a few

convolutions on these inputs to obtain essential features and finally obtain a linear output of size 4, each representing Q-value for each action.

5.3.4 Simulation and training

5.3.4.1 Description

- Here we'll be making use of TraCI api to connect to the simulations.
- From the simulations we will be retrieving multiple state spaces using the DTSE method and store them in memory.
- We will discretize the incoming lanes into cells to identify the presence or absence of vehicles in them.
- We will be training the model using an experience replay method to improve its efficiency and reduce correlation between consecutive simulation steps.
- The model will also be trained to select an action using the epsilon greedy.

5.3.4.2 Class diagram

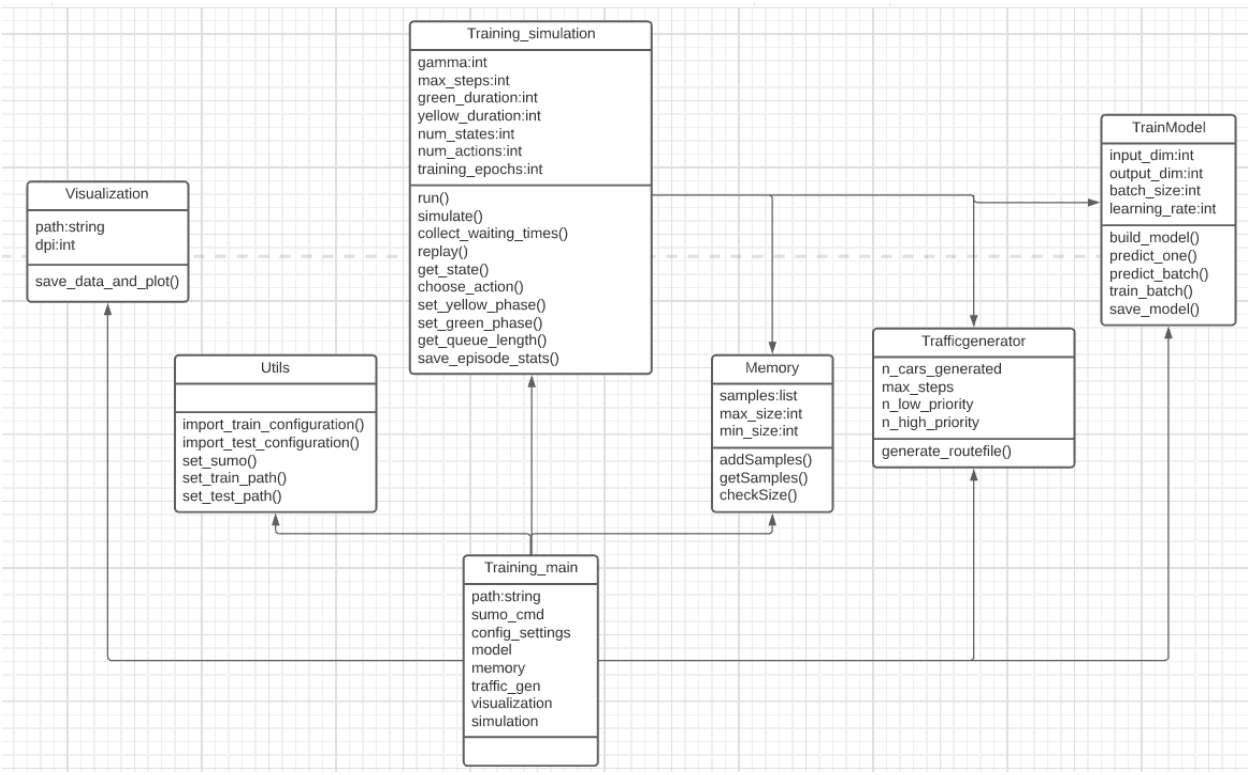


Fig 5.5 Simulation and training Class Diagram

5.3.4.3 Training_main

- This class represents the file training_main which is used to run the training simulation. It is associated with all other files/classes and has various attributes and objects.

5.3.4.4 Training_simulation

- This class represents the file training_simulation. It is responsible for running the simulations for training using various methods and attributes.
  - ❖ run() – Starts a training session by running a simulation episode.
  - ❖ simulate() - Execute steps in sumo while gathering statistics.
  - ❖ collect_waiting_times() - Get the waiting times of all the cars in the all the incoming roads.
  - ❖ choose_action() – Using epsilon-greedy policy, it decides if an explorative action is to be taken or an exploitative action is to be taken.
  - ❖ set_yellow_phase() - Set the correct yellow light phase in sumo.
  - ❖ set_green_phase() - Set the correct green light phase in sumo.
  - ❖ get_queue_length() - Get the count of all the cars having a speed of 0 in all the incoming lanes.
  - ❖ get_state() – Get the state information from the environment using DTSE in cell occupancy form.
  - ❖ replay() - Get a group of samples called batch from the memory and for each batch, update the learning equation and then train.
  - ❖ save_episode_stats() – Saving all the data and stats of each episode for analysis purposes by plotting graphs.

5.3.4.5 Memory

- The Memory class handles the memorization for the experience replay mechanism.
  - ❖ add_sample() - Add a sample into the memory.
  - ❖ get_samples() - Get n samples randomly from the memory.
  - ❖ checkSize() - Check how full the memory is.

5.3.4.6 Traffic Generator

- This class has attributes such as the number of cars that needed to be generated, max_steps, and no of high and low priority vehicles. Using these attributes it generates a route file using the generate_routefile() method.
    - ❖ generate_routefile()-This method creates a route file and generates different route id's for the simulation using Weibull distribution.

5.3.4.7 TrainModel

- This class represents a file that defines the neural network and different methods to train the neural network and predict the outputs.
    - ❖ build_model() - Build and compile a fully connected deep neural network.
    - ❖ predict_one() - Predict the action values from a single state.
    - ❖ predict_batch() - Predict the action values from a batch of states.
    - ❖ train_batch() - Train the nn using the updated q-values.
    - ❖ save_model() - Save the current model in the folder as h5 file and a model architecture summary as png.

5.3.4.8 Utils

- This class represents a file that contains some important functions related to directories like handling the creation of new model versions automatically and loading the existing models during testing.
    - ❖ import_train_configuration() - Read the config file regarding the training and import its content.
    - ❖ import_test_configuration() - Read the config file regarding the testing and import its content.
    - ❖ set_sumo() - Configure various parameters of SUMO.
    - ❖ set_train_path() - Create a new model path with an incremental integer, also considering previously created model paths.
    - ❖ set_test_path() - Returns a model path that identifies the model number provided as argument and a newly created 'test' path.

5.3.4.9 Visualization

- This class is used for plotting data and analyzing the results.
  - ❖ save_data_and_plot() - Produce a plot of performance of the agent over the session and save the relative data to txt.

- **Data members of all classes**

Data Type	Data Name	Description
string	path	Folder containing the different model versions and results
list	sumo_cmd	the command to set SUMO gui and configure various parameters of SUMO
dict	config_settings	dictionary of training configurations
object	model	object of TrainModel class
object	memory	object of Memory class
object	traffic_gen	object of TrafficGenerator class
object	visualization	object of Visualization class
object	simulation	object of Simulation class
int	input_dim	Size of environment state in the perspective of an agent
int	output_dim	the number of possible actions
int	batch_size	the number of samples to be retrieved from memory for each training iteration.
int	learning_rate	the learning rate defined for the neural network
list	samples	list of training samples stored in memory
int	max_size	the maximum number of samples the can be held in the memory

int	min_size	the min number of samples needed into the memory
int	n_cars_generated	no of cars to be generated in the simulation.
int	max_steps	Each episode duration, 1 step equals 1 second.
int	n_low_priority	no. of low priority vehicles.
int	n_high_priority	no. of high priority vehicles.
int	dpi	dots per inch
int	gamma	the parameter gamma used in the Bellman's equation
int	green_duration	the duration of each green phase in seconds
int	yellow_duration	the duration of each yellow phase in seconds
int	num_states	the size of environment state from the perspective of an agent
int	num_actions	the possible number of actions that can be taken
int	training_epochs	at the end of each episode, the number of training iterations that are to be executed

Table 5.2 Data members of all classes in Simulation and training

5.3.5 Testing

5.3.5.1 Description

- We will be testing the model with a single test episode running a completely different and random simulation with emergency vehicles configured in such a way that they can ignore the red lights.
- We will be testing both the models (ANN and CNN) for multiple intersections as well.

5.3.5.2 Class diagram

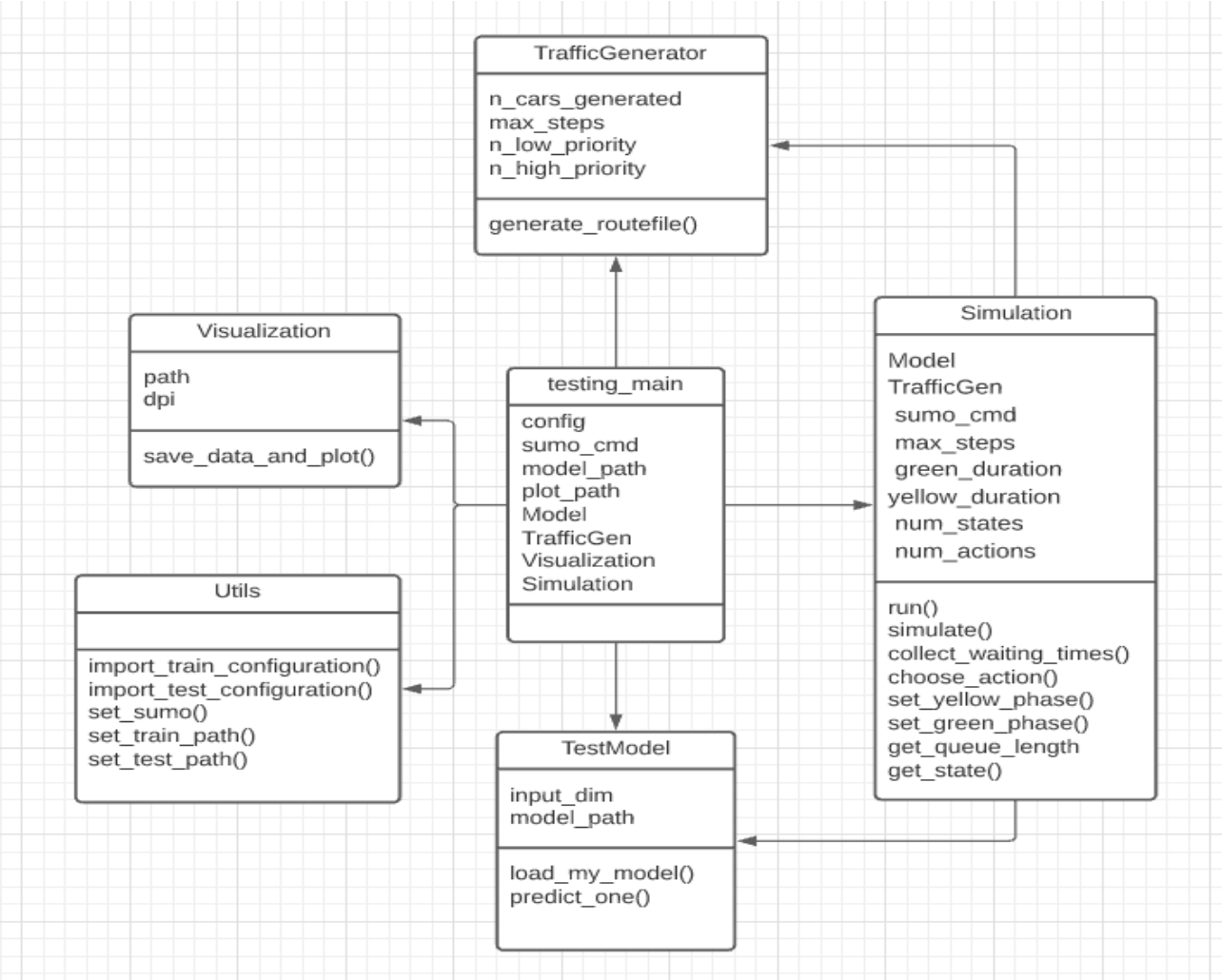


Fig 5.6 Testing Class Diagram

5.3.5.3 Testing_main

- This class represents the file testing_main which is used to run the testing simulation.

5.3.5.4 TestModel

- This class represents a file that defines the neural network and different methods to train the neural network and predict the outputs.
  - ❖ load_my_model() - Load the model stored in the folder specified by the model number, if it exists

- predict_one() - Predict the action values from a single state

5.3.5.5 Visualization

- This class is used for plotting data and analyzing the results.
  - save_data_and_plot() - Produce a plot of performance of the agent over the session and save the relative data to txt

5.3.5.6. Simulation

- This class represents the file testing_simulation.
  - run() - Starts a training session by running a simulation episode.
  - simulate() - Execute steps in sumo while gathering statistics.
  - collect_waiting_times() - Get the waiting times of all the cars in the all the incoming roads.
  - choose_action() - Using epsilon-greedy policy, it decides if an explorative action is to be taken or an exploitative action is to be taken.
  - set_yellow_phase() - Set the correct yellow light phase in sumo.
  - set_green_phase() - Set the correct green light phase in sumo.
  - get_queue_length() - Get the count of all the cars having a speed of 0 in all the incoming lanes.
  - get_state() - Get the state information from the environment using DTSE in cell occupancy form.

5.3.5.7 Utils

- This class represents a file that contains some important functions related to directories like handling the creation of new model versions automatically and loading the existing models during testing.
  - import_train_configuration() - Read the config file regarding the training and import its content
  - import_test_configuration() - Read the config file regarding the testing and import its content
  - set_sumo() - Configure various parameters of SUMO

- set_train_path() - Create a new model path with an incremental integer, also considering previously created model paths
- set_test_path() - Returns a model path that identifies the model number provided as argument and a newly created 'test' path.

- **Data members of all classes**

Data Type	Data Name	Description
dict	config	dictionary of testing configurations
list	sumo_cmd	the command to set SUMO gui and configure various parameters of SUMO.
string	model_path	path to the folder containing the model version.
string	plot_path	path to the folder containing the model version and test results
object	model	object of TestModel class
object	traffic_gen	object of TrafficGenerator class
object	visualization	object of visualization class
object	simulation	object of simulation class
int	input_dim	Size of environment state in the perspective of an agent
int	n_cars_generated	no of cars to be generated in the simulation.
int	n_low_priority	no. of low priority vehicles.
int	n_high_priority	no. of high priority vehicles.
string	path	path to the folder containing the model versions and results.
int	dpi	dots per inch.

int	max_steps	Each episode duration, 1 step equals 1 second.
int	green_duration	duration of green light phase.
int	yellow_duration	duration of yellow light phase.
int	num_states	the size of the state of the env from the agent perspective.
int	num_actions	no of possible actions.

Table 5.3 Data members of all classes in Testing

5.3.6 Memory

5.3.6.1 Description

- We will be implementing the experience replay mechanism.
- The memory.py file will handle the memorization for the experience replay mechanism.
- We have a function to add a sample to memory, while there is another function which retrieves a batch of samples from the memory.

5.3.7 Utils

5.3.7.1 Description

- This file contains directory related functions such as loading models for testing and creating different versions of trained models, importing configuration settings, etc.

5.3.8 Visualization and analysis study

5.3.8.1 Description

- Once the testing is done we will be performing an analysis study to see the performance of the agent in both the models compared to normal traffic, to study the reward trend, and see the agent's behavior in different levels of traffic.

5.3.9 Packaging and Deployment Diagrams

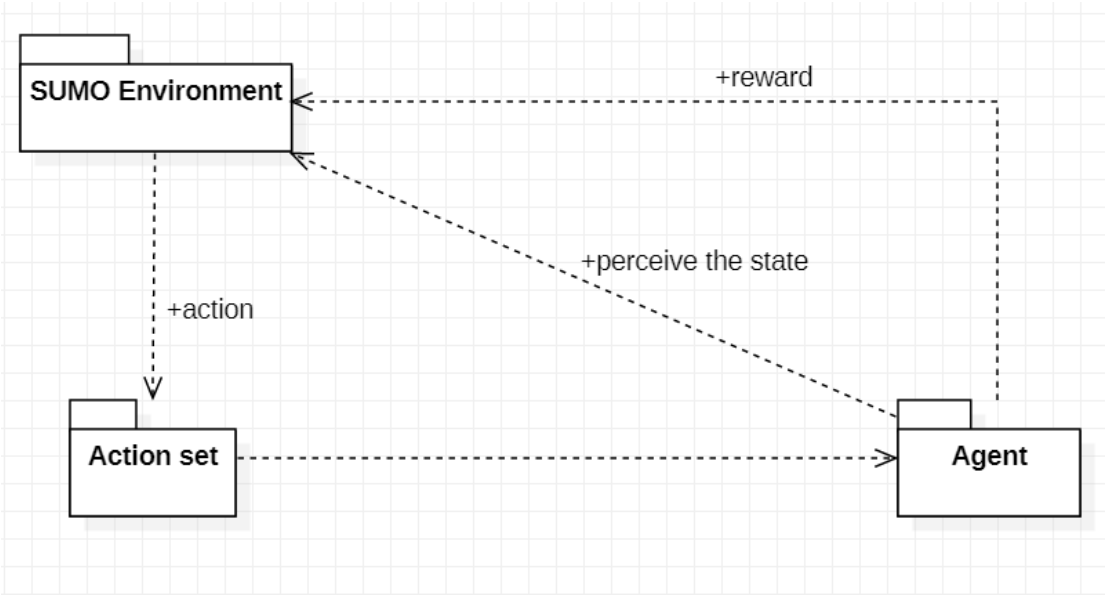


Fig 5.7 Packaging Diagram

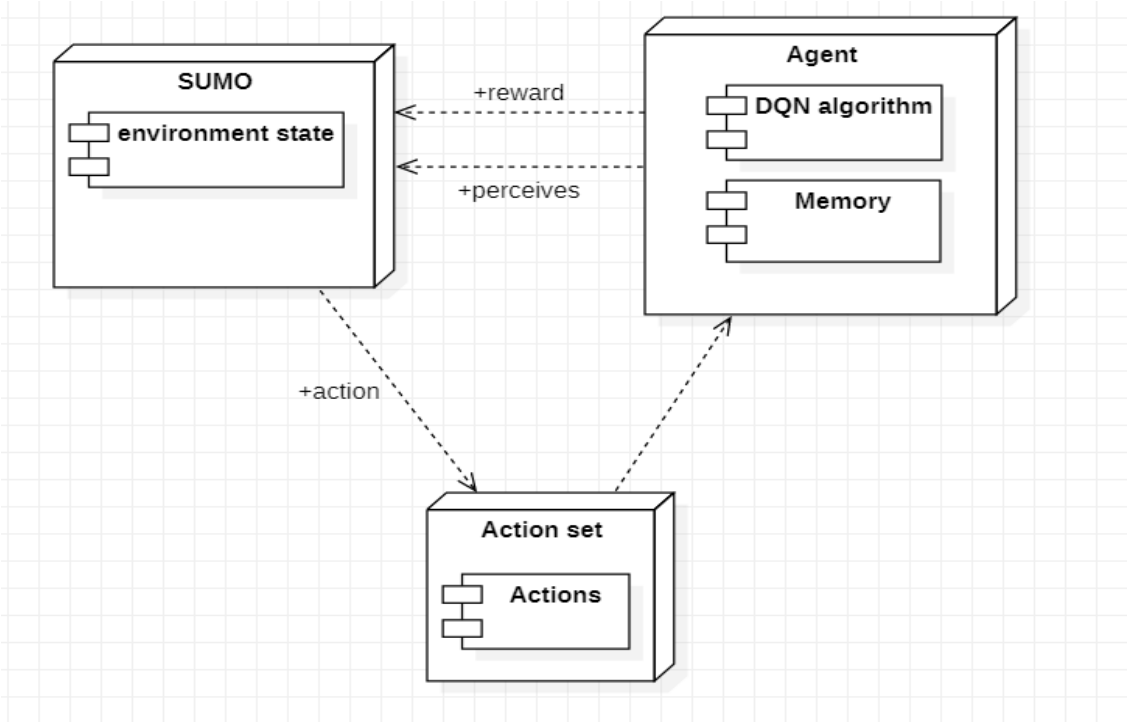


Fig 5.8 Deployment Diagram

CHAPTER 6

PROPOSED METHODOLOGY

- Reinforcement learning is the concept that we will be using to solve this problem of traffic congestion and traffic control
- Here we have an agent perceiving the environment and depending on that state of the environment, it performs a certain action. If the action taken was a good one, it gets a positive reward else it gets a negative reward. The agent has a goal which it achieves by maximizing the rewards
- The learning mechanism used here is deep Q learning, that is a mix of DNN and Q-learning. We propose to use an experience replay mechanism for enhancing the performance and efficiency of the agent. We also use an epsilon greedy policy for training the agent.
- We propose to use an agile methodology/approach to solve this problem
- Agile is a simple and effective process with reduced risks and better control over the project.
- Since Agile is an iterative process, it allows teams to keep on learning and growing with time and continue improving.
- Agile allows us to make changes if necessary without causing major issues.
- It also allows us to fix errors in the middle of the project making it more flexible to change.
- Delivers early partial working solutions.
- Promotes teamwork and cross training.

CHAPTER 7

IMPLEMENTATION

Description of the Reinforcement Learning approach used in our project is as follows:

State representation: We followed the Discrete State Space Encoding (DTSE) to represent our state. In this encoding each incoming lane is divided into cells that can identify the presence or absence of a vehicle inside them.

Action set: The possible actions that can be taken by the agent are contained in the action set. The number of actions in the action set is the same as the number of incoming lanes in an intersection. For a 4-way intersection there are 4 possible actions and each action is a phase of the traffic signal that is North Advance, East Advance, South Advance and West Advance. Similarity 3-way will have 3 actions in its action set and 5-way will have 5 actions in its action set.

Reward: Reward is something which the agent receives as a consequence of its chosen action. A good action results in a positive reward while a bad action results in a negative reward. We have used change in cumulative waiting time between actions as the metric to calculate the reward.

Learning mechanism: The learning mechanism used is Deep Q-Learning. It is a form of model free learning, which assigns a Q-value also known as quality value to an action that was performed on the state of the environment.

In Q learning performs well in small state-spaces but its performance falls off when working with complex sophisticated environments. In small state spaces we just need to update a few states iteratively in the q-table, but when we are working in complex and large environments, the state-space is very large and it's computationally inefficient to update each state-action pair in the state space iteratively. Now instead of using value iteration to directly compute the Q values and find the optimal Q function, we use a function approximator to estimate the optimal Q function, hence neural networks come into picture. The output Q values from the neural network (one for each action taken) is compared to the q values obtained from the Bellman’s equation to compute loss. This loss is minimized using back propagation and tweaking the weights using stochastic gradient descent using Adam optimizer.

Training Process:

We have used the experienced relay technique to enhance performance and efficiency. This technique consists of submitting all the knowledge needed by the agent for the training process as a group or collection of random samples also called as a batch rather than instantly submitting the knowledge gathered by the agent during the phase of simulation. All the samples collected during the training phase are stored in the memory and a group of samples are retrieved from the memory as a batch. Training includes iterative learning of a Q-value function with the help of knowledge available within the batch of samples retrieved from the memory.

The equation for the Q-learning function is

$$Q(s_t, a_t) = \text{reward} + \gamma \cdot \max_A Q'(s_{t+1}, a_{t+1})$$

The action-selection policy we used is the epsilon-greedy policy. This is the policy that decides whether an agent performs an exploratory action where the agent tries something new, or performs an exploitative action where the agent exploits what it already knows to achieve favorable rewards.

We have implemented different kinds of road network which are the following:'

3-way:

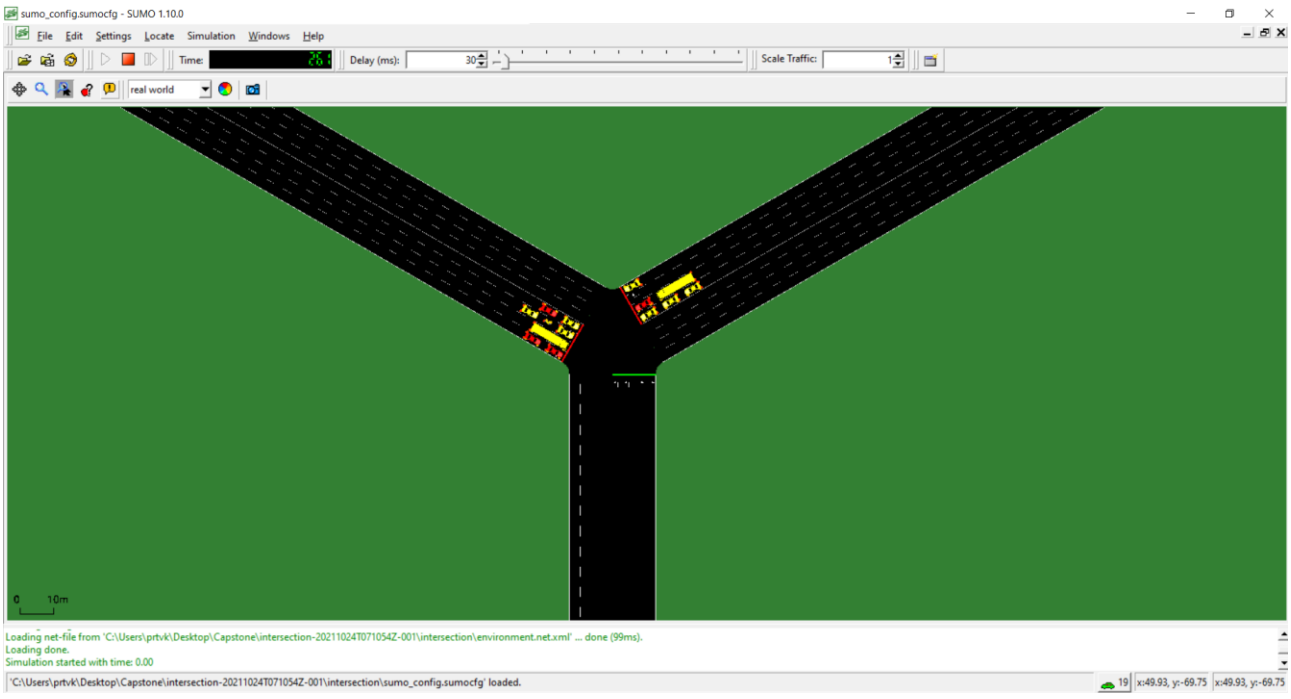


Fig 7.1 3-way intersection

4-way:

Fig 7.2 4-way intersection

5-way:

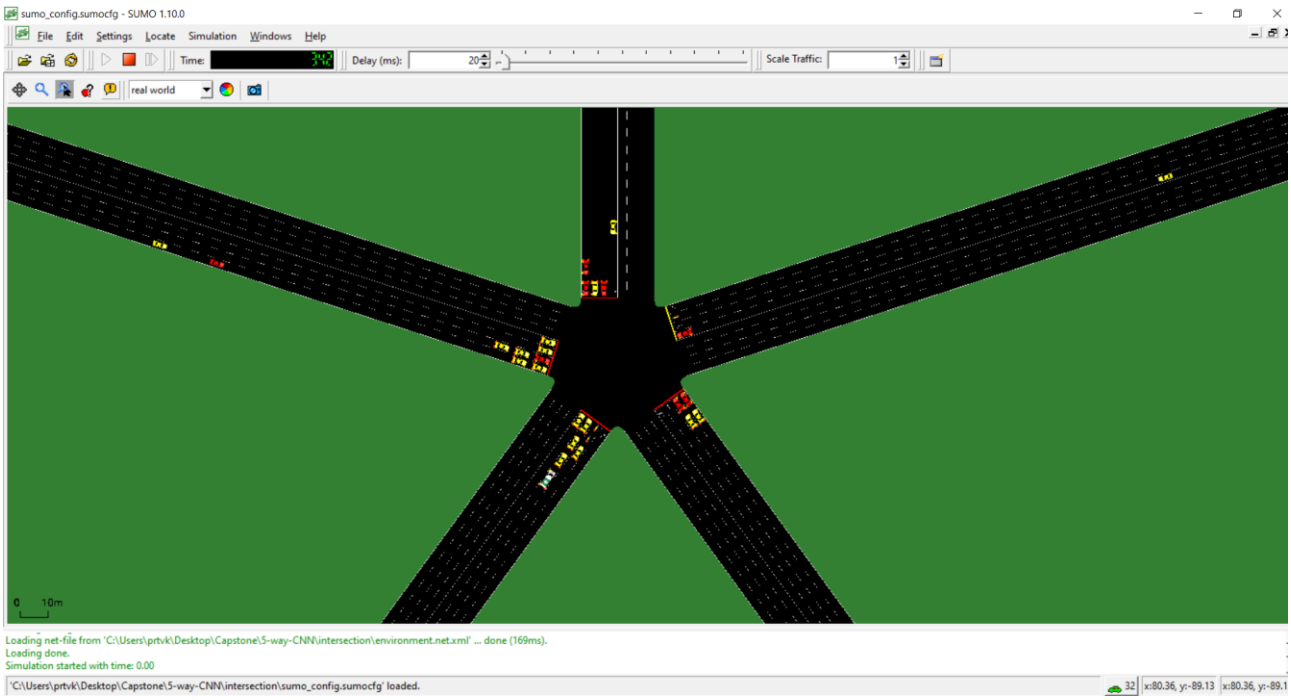


Fig 7.3 5-way intersection

Complex:

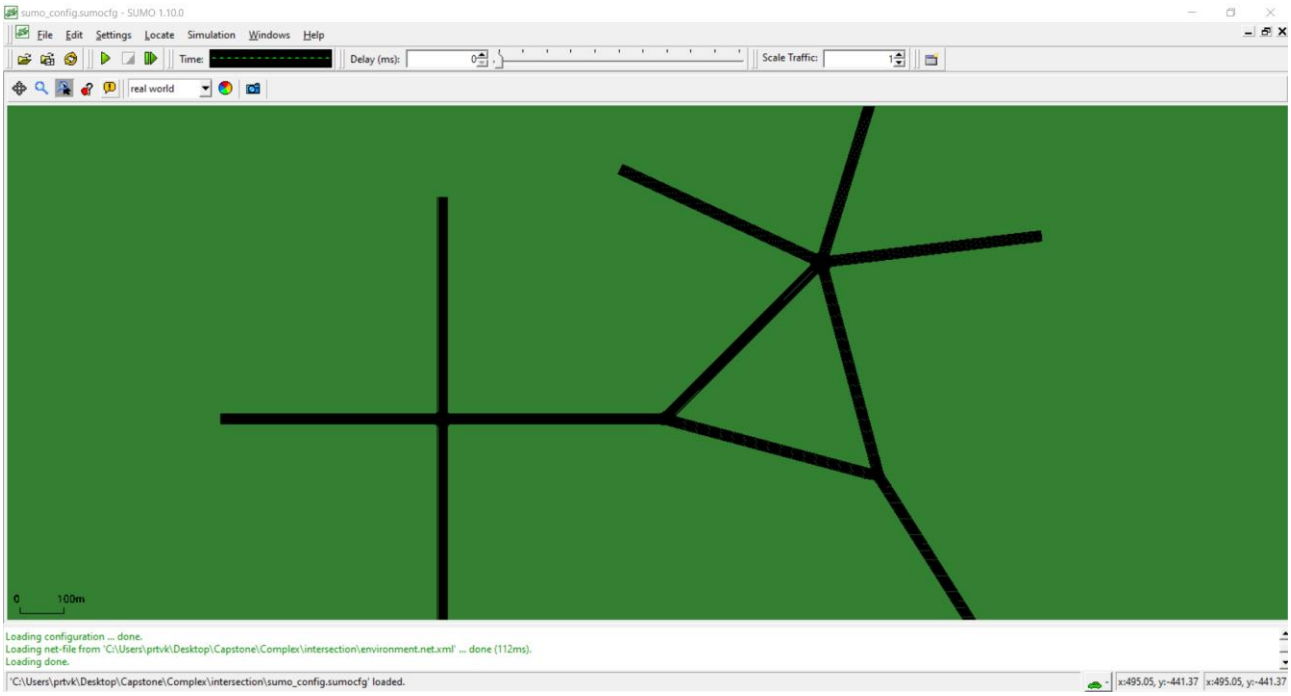


Fig 7.4 Complex network

Further during the testing phase, we tested the trained model with random traffic simulations to see its performance. Here we also configured our emergency vehicles like ambulances, fire trucks, police vehicles, etc. with blue light devices in SUMO to ignore traffic signals so that they don't have to wait unnecessarily. By doing this any vehicle can drive with special rights and change lanes without regard for strategic considerations. All the other vehicles ahead of the emergency vehicles are forced to form a rescue lane to let the emergency vehicle pass through.

Once the testing is done, we perform visualization and analysis study to assess the performance of the agent. This includes visual inspection where we can see the performance of the agent compared to traditional traffic signals. We analyze the reward trend, waiting times and queue lengths for the agents trained using both ANN and CNN. We further analyze the comparison between the models trained using ANNs and CNNs and finally investigate the effects of partially training the models at some intersections in a complex road network.

CHAPTER 8

RESULTS AND DISCUSSIONS

On visual inspection it was clearly seen that the trained model performed much better than the normal simulation, as the trained model had learnt an optimal policy function to maximize the rewards and choose an optimal action.

Our agents were able to reduce the cumulative delay of vehicles and maximize the rewards during the training process as we can see in the below graphs.

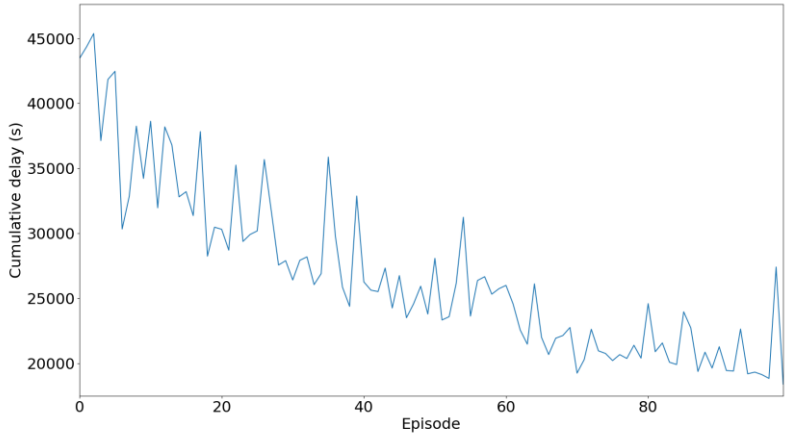


Fig 8.1 Cumulative delay vs. Episode graph

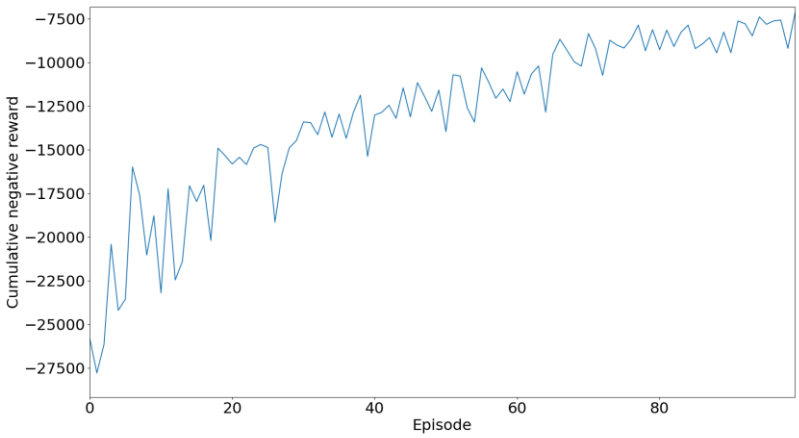


Fig 8.2 Cumulative negative reward vs. Episode graph

On comparison with traditional traffic signals it was seen that the trained agent was able to reduce queuing in traffic signals significantly as seen in the below graph

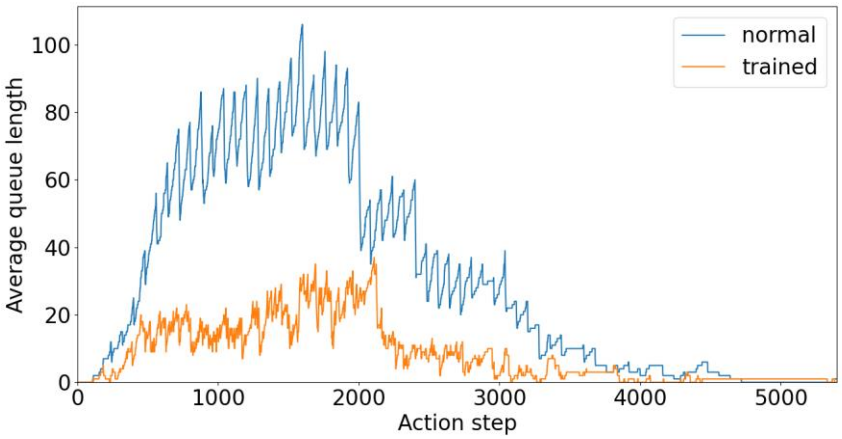


Fig 8.3 Average queue length vs. Action step graph (normal vs. trained)

The reason for this is the duration of the traffic light phases. Here we have chosen a normal scenario wherein the duration of green and yellow phases is 30 seconds. During training, apart from the agent choosing an appropriate action, we are making sure that the green light duration for a particular lane does not exceed 10 seconds and the yellow light duration does not exceed 4 seconds.
By doing this it allows the agent to take quicker actions depending on the surroundings. In case the action chosen at action step t is the same as the action taken at action step t-1, then the green light persists and there is no yellow phase, hence there is only 10 seconds (10 simulation steps between the two actions). Else if the consecutive actions are not the same then there is a 4 second yellow phase between the two actions making it a total of 14 seconds (14 simulation steps) between the two actions.

On analyzing the performance of the agents, it was seen that the agents trained using the convolutional neural networks performed much better than the agents trained using the Artificial neural network as seen in the plots below.

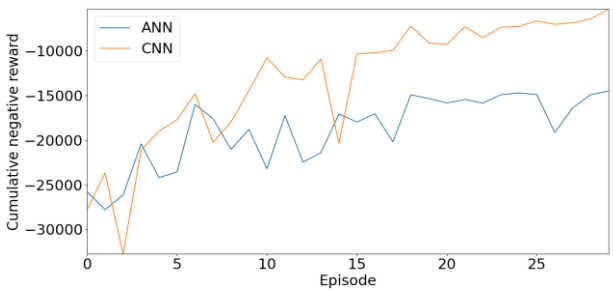


Fig 8.4 Cumulative negative reward vs. Episode graph (ANN vs. CNN)

On analysis of the reward trend we can see that the CNN is able to maximize the rewards much quicker compared to ANN with lesser training.

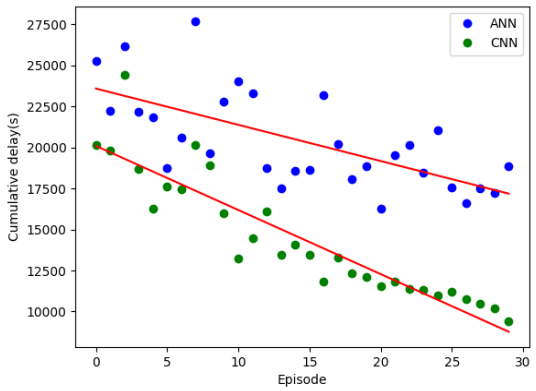


Fig 8.5 Cumulative delay vs. Episode regression (ANN vs. CNN)

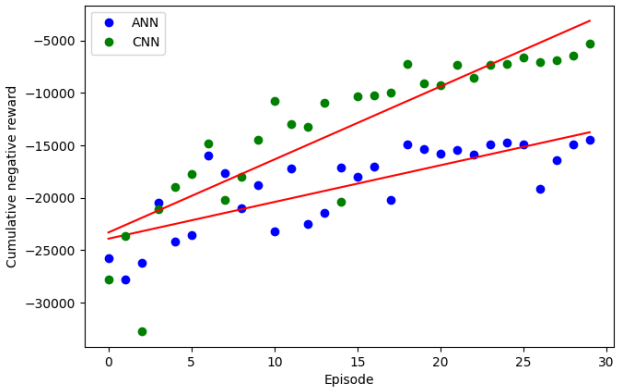


Fig 8.6 Cumulative negative reward vs. Episode regression (ANN vs. CNN)

As we can see from the above plot, CNN has a much more negative slope than ANN while reducing the cumulative delay during the training phase.

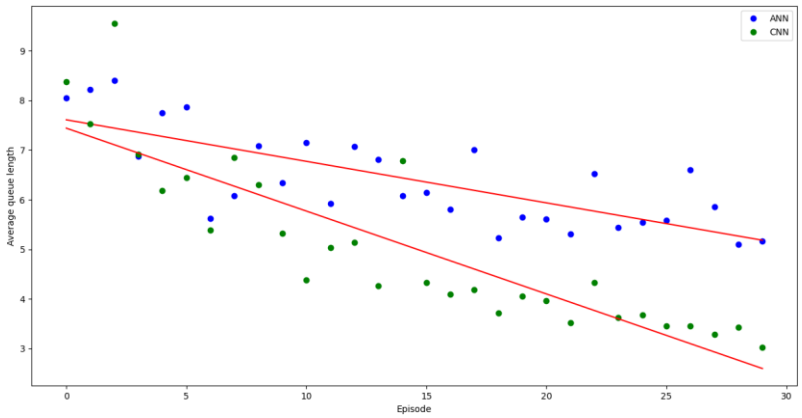


Fig 8.7 Average queue length vs. Episode regression (ANN vs. CNN)

We can also see that the CNN is much more effective in reducing the Average queue length during training compared to CNN.

The reason why CNN performs much better compared to ANN is because of its ability to extract important features using DTSE. This allows CNN to develop high-level state representations. Moreover, the CNN takes multiple data inputs, in our case the position and velocity matrix compared to the position matrix alone in the ANN. This allows CNNs to achieve better results with higher accuracy. Although CNN takes longer to train, it gives efficient results much quickly.

It was also seen that the models performed better in smaller intersections than the bigger ones.

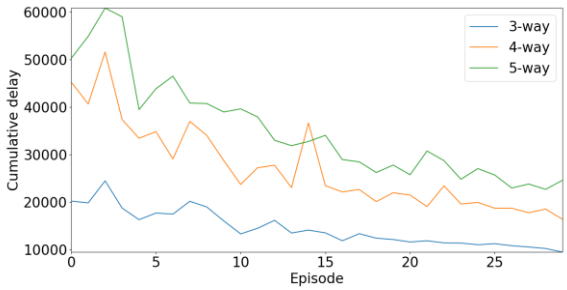


Fig 8.8 Cumulative delay vs. Episode graph (3-way vs. 4-way vs. 5-way)

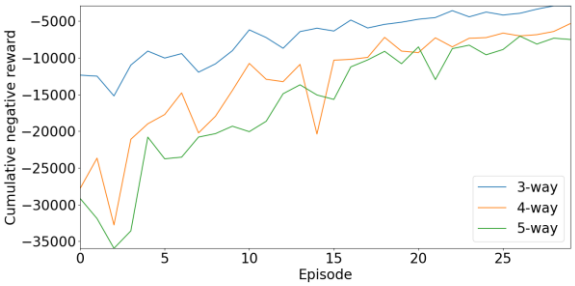


Fig 8.9 Cumulative negative reward vs. Episode graph (3-way vs. 4-way. vs 5-way)

This is mainly due to the fact that in smaller intersections, the state space representation is smaller compared to larger intersections and the model needs to choose from a smaller action set making it easier and quicker to train.

For complex intersection networks, when one model is undertrained, we found that it performs worse when compared to the fully trained models. The queue length keeps on increasing for the partially trained model because of the congestion in the undertrained network. This can be seen from the plot below.

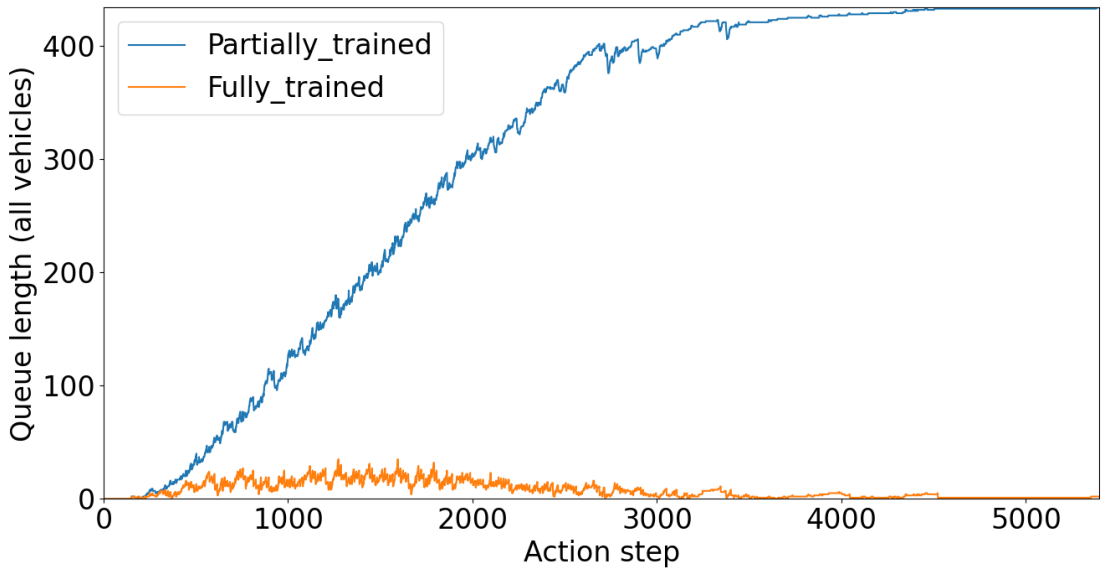


Fig 8.10 Queue length vs. Action Step graph (Partially trained vs Fully trained)

This shows that when training for complex networks, each model has to be trained fully to achieve better results. Even if one of them is undertrained, the whole network will perform worse, since each individual network is interconnected and queue length in one network impacts the whole network.

CHAPTER 9

CONCLUSION AND FUTURE WORK

The work carried out presented a RL approach to monitor and control traffic lights in a much more effective manner wherein they are able to adjust dynamically to real time traffic and take appropriate actions. This work was carried out on a traffic simulator which provides a realistic environment to carry out complex simulations. The agents were trained using the DQN mechanism and its performance was analyzed using various parameters. It was important to properly understand the application context and apply them to machine learning models to achieve proper results.

This work has shown that proper use of Machine learning approaches can provide effective and excellent results. Reinforcement learning, despite its limited application in the real world, is still one of the hot topics in the field of Artificial Intelligence and could potentially have a great impact in the future.

There are however many things that can be improved in this work. Future works could include improving the results achieved and expanding it to a much larger network. The training of the agents could be improved by using different techniques such as target network, clipping rewards and skipping frames. We can also try using different reward and Q functions to try and improve the model further. Proper analysis is important to understand the advantages and possible disadvantages that it can bring on introduction to the real world.

REFERENCES AND BIBLIOGRAPHY

[1] Andrea Vidali, Luca Crociani, Giuseppe Vizzari, Stefania Bandini, "A Deep Reinforcement Learning Approach to Adaptive Traffic Lights Management".

[2] Wade Gendersa, Saiedeh Razavib, "Using a Deep Reinforcement Learning Agent for Traffic Signal Control".

[3] Hua Wei, Guanjie Zheng, Huaxiu Yao, Zhenhui Li, "IntelliLight: A Reinforcement Learning Approach for Intelligent Traffic Light Control"

[4] Arel C. Liu1 T. Urbanik, A.G. Kohls, "Reinforcement learning-based multi-agent system for network traffic signal control"

[5] SUMO documentation: https://sumo.dlr.de/docs/
Copyright © 2001-2021 German Aerospace Center (DLR) and others.

[6] R. S. Sutton, A. G. Barto et al. Introduction to reinforcement learning. MIT press Cambridge, 1998, vol. 135.

[7] Monireh Abdoos, Nasser Mozayani, and Ana LC Bazzan. 2013. Holonic multi-agent system for traffic signals control. Engineering Applications of Artificial Intelligence 26, 5 (2013), 1575–1587.

[8] Baher Abdulhai, Rob Pringle, and Grigoris J Karakoulas. 2003. Reinforcement learning for true adaptive traffic signal control. Journal of Transportation Engineering 129, 3 (2003), 278–285.

[9] JAYAKRISHNAN R., MATTINGLY S., MCNALLY M.: 'Performance study of SCOOT traffic control system with non-ideal detectorization: field operational test in the city of Anaheim'. 80th Ann. Meeting of the Transportation Research Board, Washington, DC, 2001.

Appendix A: Definitions, Acronyms and Abbreviations

1. ANN – Artificial Neural Network
2. API - Application programming interface
3. CNN - Convolutional Neural Network
4. CPU – Central processing unit
5. DNN - Deep Neural Network
6. DTSE - Discrete traffic state encoding
7. DQN - Deep Q Network
8. DQTSCA – Deep Q network traffic signal control agent
9. ML - Machine Learning
10. RAM - Random Access Memory
11. RL - Reinforcement learning
12. STSCA – Shallow traffic signal control agent
13. SUMO - Simulation of Urban MObility
14. TCP - Transmission Control Protocol
15. TraCI - Traffic Control Interface

Final_Report_PW22NKS01

ORIGINALITY REPORT

11%	6%	5%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to PES University Student Paper	5%
2	faculty.ist.psu.edu Internet Source	1%
3	pdfs.semanticscholar.org Internet Source	1%
4	ceur-ws.org Internet Source	1%
5	Submitted to Flinders University Student Paper	<1%
6	www.programcreek.com Internet Source	<1%
7	Salomon Wollenstein-Betech, Christian Muise, Christos G. Cassandras, Ioannis Ch. Paschalidis, Yasaman Khazaeni. "Explainability of Intelligent Transportation Systems using Knowledge Compilation: a Traffic Light Controller Case", 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), 2020	<1%

Publication

8	Arel, I., C. Liu, T. Urbanik, and A.G. Kohls. "Reinforcement learning-based multi-agent system for network traffic signal control", IET Intelligent Transport Systems, 2010. Publication	<1 %
9	Mahboubeh Shamsi, Abdolreza Rasouli Kenari, Roghayeh Aghamohammadi. "Reinforcement learning for traffic light control with emphasis on emergency vehicles", The Journal of Supercomputing, 2021 Publication	<1 %
10	Hua Wei, Guanjie Zheng, Huaxiu Yao, Zhenhui Li. "IntelliLight", Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018 Publication	<1 %
11	Aquib Junaid Razack, Vysyakh Ajith, Rajiv Gupta. "A Deep Reinforcement Learning Approach to Traffic Signal Control", 2021 IEEE Conference on Technologies for Sustainability (SusTech), 2021 Publication	<1 %
12	www.ijert.org Internet Source	<1 %

13	"Deep Reinforcement Learning", Springer Science and Business Media LLC, 2020 Publication	<1 %
14	arxiv.org Internet Source	<1 %
15	etrr.springeropen.com Internet Source	<1 %
16	"Modeling, Design and Simulation of Systems", Springer Science and Business Media LLC, 2017 Publication	<1 %
17	Submitted to Middle East College of Information Technology Student Paper	<1 %
18	digitalcommons.njit.edu Internet Source	<1 %
19	docs.microsoft.com Internet Source	<1 %
20	www.newindianexpress.com Internet Source	<1 %
21	Longxin Lin, Haibin Xie, Daibing Zhang, Lincheng Shen. "Supervised Neural Q_learning based Motion Control for Bionic Underwater Robots", Journal of Bionic Engineering, 2010 Publication	<1 %

22	escholarship.org Internet Source	<1%
23	lib.buet.ac.bd:8080 Internet Source	<1%
24	tudr.thapar.edu:8080 Internet Source	<1%
25	Lei Li, Wenting Liu, Lindi Xiao, Hui Sun, Shi Wang. "Environmental Protection in Scenic Areas: Traffic Scheme for Clean Energy Vehicles Based on Multi-agent", Computational Economics, 2018 Publication	<1%
26	businessdocbox.com Internet Source	<1%
27	ir.lib.uwo.ca Internet Source	<1%
28	Omid Rezvan, Seyed Tabatabaei, Marziyeh Yoosefi. "A Signal Timing Method to Reduce the Emissions in Arterials (Case Study: Enghelab Avenue in Ahvaz)", Current World Environment, 2015 Publication	<1%
29	Yujian Ye, Dawei Qiu, Jing Li, Goran Strbac. "Multi-Period and Multi-Spatial Equilibrium Analysis in Imperfect Electricity Markets: A	<1%

Novel Multi-Agent Deep Reinforcement Learning Approach", IEEE Access, 2019
Publication

30	library.fpinnovations.ca Internet Source	<1 %
31	www.ai.rug.nl Internet Source	<1 %
32	Constança Oliveira e Sousa. "Dealing with Errors in a Cooperative Multi-agent Learning System", Lecture Notes in Computer Science, 2006 Publication	<1 %
33	Faisal Ahmed, Ho-Shin Cho. "A Time-Slotted Data Gathering Medium Access Control Protocol Using Q-Learning for Underwater Acoustic Sensor Networks", IEEE Access, 2021 Publication	<1 %
34	cuckoo.readthedocs.io Internet Source	<1 %
35	deepai.org Internet Source	<1 %
36	www.purdue.edu Internet Source	<1 %
37	Anand Paul, Hameed Pinjari, Won-Hwa Hong, Hyun Cheol Seo, Seungmin Rho. "Fog	<1 %

Computing-Based IoT for Health Monitoring System", Journal of Sensors, 2018

Publication

Exclude quotes	On	Exclude matches	< 5 words
Exclude bibliography	On		

Smart Traffic Light Controller using Deep Reinforcement Learning

Abhishek A
Dept. of Computer Science Engineering
PES University
Bengaluru, India
abhishek.adinarayan@gmail.com

Prathvik Nayak
Dept. of Computer Science Engineering
PES University
Bengaluru, India
prtvk13@gmail.com

Krishna P Hegde
Dept. of Computer Science Engineering
PES University
Bengaluru, India
krishnahegde02@gmail.com

A Lakshmi Prasad
Dept. of Computer Science Engineering
PES University
Bengaluru, India
avineniprasad199@gmail.com

Dr. Nagegowda K S
Dept. of Computer Science Engineering
PES University
Bengaluru, India
nagegowda@pes.edu

***Abstract*—Transportation has become a major priority for people nowadays. While technological advancements have made transportation easier, monitoring and controlling traffic as well as simulating it has become a significant challenge despite the significant research that has been carried out to tackle this problem. An emerging trend to solve this problem involves using deep reinforcement learning techniques which has shown significant progress and promising results in recent studies. There is an increasing demand for developing an intelligent traffic controlling agent which can dynamically adjust to real time traffic rather than just operating by hand-craft rules such as timers. We employ modern deep reinforcement learning methods such as Q-learning and deep neural networks with an agent-based traffic simulator SUMO ("Simulation of Urban MObility") which provides a synthetic yet realistic environment for simulating real-world like traffic and explore the outcomes of the actions that were performed on this environment. Additionally we also aim to analyze the performance of the agent by comparing it to traditional traffic signals.**

Keywords—Reinforcement learning, traffic controlling agent, Q-Learning, deep neural network, SUMO

I. Introduction

We know pretty well for a fact that modern society depends heavily on transportation systems. Everyone desires to move from their origin to the destination in a smooth manner as possible, but due to the increase in population and the number of vehicles, traffic congestion, travel delay, accidents and unnecessary emissions have increased substantially and due to this there is a huge demand for improving the road infrastructure to ensure smooth flow of traffic. The fact that the world is changing and developing very fast makes it a difficult task to control and monitor the traffic, especially in large metropolitan areas that are growing around the world such as Tokyo, Bengaluru, Los Angeles, Mumbai, etc.

Majority of the traffic congestions occur due to lack of capacity on the roads i.e the number of vehicles that are travelling on the roads are exceeding their capacity. Apart from this, environmental issues, human errors, etc are other causes for traffic congestion. Solving this problem of traffic control and monitoring and, in general, reducing traffic congestion still remains a major challenge in spite of substantial research that has been carried out to tackle this problem.

Artificial Intelligence plays a key role in tackling this problem. We attempt to try and develop a smart traffic signal controller that is able to analyse and adjust dynamically to real-time traffic rather than just following traditional hand-craft rules such as timers.

Reinforcement learning could be an optimal technique to solve this problem of traffic congestion and traffic control. Reinforcement learning is one of the ML techniques which comprises an agent, state representation, actions set, Q-function and a reward function. The agent perceives the environment through its sensors and depending on that state of the environment, it performs a certain action. If the action taken was a good one, it gets a positive reward else it gets a negative reward. The agent has a goal which it achieves by maximizing the rewards.

In this paper, we have referred to the ideas proposed by Andrea Vidali et al. [1] to use a reinforcement learning technique called deep Q learning which is a mix of deep neural networks and Q-Learning. The approach is to use discrete traffic state encoding (DTSE) for state space representation of traffic and use an epsilon greedy policy along with experience replay mechanism for training purposes. We have extended this approach using different models like Artificial neural networks (ANN) and Convolutional neural networks (CNN) for different types of networks namely 3-way, 4-way, 5-way intersections and complex combinations of these intersections.

We will be using the SUMO traffic simulator for simulating the traffic. "Simulation of Urban MObility" (SUMO) is an open source, highly portable, microscopic and continuous traffic simulation package designed to handle large networks. It allows for intermodal simulation including

pedestrians and comes with a large set of tools for scenario creation. We will be using the TraCI API for interacting with SUMO using python. TraCI follows a client server architecture based on TCP, where the controller (external script) acts as a client and SUMO software acts as a server. The "controller" is an external python script which gets the simulation state information from the server (SUMO) and then responds by sending instructions [5].

II. Literature Survey

The research that has been conducted on traffic signal control using reinforcement learning has been significant. The limitations of the earlier attempts include absence of sufficient compute power and simple simulations [9][10]. Starting from the early 2000's, the constant advancement in these areas has led to the creation of various simulation tools which are more realistic and sophisticated. Traffic simulators have become a popular tool among traffic researchers since they can replicate traffic behaviour from the real-world. The research held has varied in the type of reinforcement learning, definition of state space, definition of action space, definition of reward, the simulator used, the geometry of the traffic network and the model for generation of vehicle. The number of queued vehicles [10][11][12] and traffic flow [13] are the most popular attributes of traffic that were defined for the state space in earlier research attempts. All available signal phases [13] or restricted to green phases only [11][12], have been the defined action space. Change in delay [13] and change in queued vehicles [11][12] are among the frequently used reward definitions.

Andrea Vidali et al. worked on simulation of traffic, as well as traffic monitoring and control using reinforcement learning approach with the help of a microscopic agent-based simulator [1]. Here they have used the DTSE [2] method for encoding the state information and have used a four way intersection as their environment. They have generated traffic according to Weibull distribution and have simulated different levels of traffic as different traffic scenarios. They have made use of two different reward functions: i) literature reward function - which uses *total waiting time* as metric, and ii) alternate reward function - which uses *accumulated total waiting time* as metric. The learning technique used here was Deep Q learning using an artificial neural network and the training was done using an epsilon greedy policy with experience replay mechanism. The performance was assessed in two parts: analysing the reward trend and then comparing with regular traffic signals. It was found that the alternative reward function gave better results than the literature reward function and it was seen that the agent performed better in low to medium traffic situations which led to think that the agents need to be separately developed to different traffic situations so that it can select the most appropriate configuration.

The Pennsylvania State University scholars have conducted research on intelligent traffic light control using reinforcement learning [3]. The experiment was conducted on a simulation platform SUMO which provides flexible APIs for road network design, traffic volume simulation and traffic light control. The evaluation metrics were reward, queue length, delay and duration. They compared methods of fixed-time control, self organising Traffic light control, deep reinforcement learning for traffic light control. In addition to the baseline methods, they considered several other variations of the model by adding memory palace and phase gates to the base model. The datasets that were used for conducting the experiments were synthetic data and real world data. They conducted in-depth analysis on the performance of all the proposed models with different constraints on the dataset.

In the research carried out by Wade Genders, they developed an agent for traffic signal control using a Deep Q-network (DQTSCA) [2] and a deep convolutional neural network (CNN) was used to model the action-value function and reinforcement learning technique was used to train it in an open source traffic simulator software called SUMO, using an intersection which is isolated [2]. They have proposed a new definition for the state space called discrete traffic state encoding (DTSE) [2] which contains more relevant information as it is an improved representation of traffic. The CNN will be able to perceive information about the traffic that is more relevant with the help of DTSE. To make a comparison against their proposed DQTSCA, they also developed a shallow traffic signal control agent (STSCA) [2]. From the results obtained it was clear that the STSCA was outperformed by the DQTSCA, mainly because of the utilization of the DTSE and its deep architecture [2].

In the research carried out by Arel C. Liu1 T. Urbanik, A.G.Kohls [4], it has been observed that multi-crossing point traffic organization, clogging in a solitary path doesn't just affect upstream traffic, yet in addition different convergences. Hence, a proficient strategy that can improve the reward and could handle traffic issues was desperately needed. In this research [4], a multi-agent setup is being used, whereby Reinforcement Learning is used as a method for handling the various intersections in the traffic system. A RL issue is characterized once states, actions, rewards are plainly characterized. Keeping that in mind, these fundamental builds are depicted with regards to the problem. At every simulation time step, local traffic statistics influence the local state of an intersection. An eight-dimensional feature vector is used for indicating each state with each element indicating the relative flow of traffic at any one lane. Relative traffic flow can be measured by dividing the total delay of vehicles with the average delay. However, a central junction agent can have access to all its neighbouring junction states, thus such extra data permits it to anticipate upstream traffic streams, subsequently increasing the performance simulation results. One time step is referred to as one time unit in the considered environment. One discrete second is referred to as one time step. Once every 20 time units an action is taken, also only one vehicle is permitted to cross junctions at a given interval. The approach mentioned was created while seeing the traffic issue artificial intelligence task, indicating a shift from

traditional traffic scheduling issue formulations. The basic goal here was to increase expected reward points which directly influences in reducing traffic issues.

However, in some of these researches the solution is limited to a single intersection (generally 4 way) using a single model. Also these researches do not consider priorities of emergency and authority vehicles in the simulations. In our project we aim to expand these solutions to different intersections and also to multiple intersections in a complex network using different models such as ANN and CNN. We will be analysing the performance of these agents compared to traditional traffic signals and also study it's reward trends and its effects on undertraining in a complex network. We will also be studying the performance of these models in different intersections and also determine which type of model performed better than the other.

III. Proposed Solution

Description of the Reinforcement Learning approach used in our project is as follows:

State representation: We followed the Discrete Traffic State Encoding (DTSE) to represent our state. In this encoding each incoming lane is divided into cells that can identify the presence or absence of a vehicle inside them.

Position Matrix

0	0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	1	1

Velocity Matrix

0	0	0	0	0	0	0	0.2	0	0
0	0	0	0	0	0	0	0.4	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0.8	0	0	0	0	0	0.2	0

Fig. 1. Discrete Traffic State Encoding (DTSE)

Action set: The possible actions that can be taken by the agent are contained in the action set. The number of actions in the action set is the same as the number of incoming lanes in an intersection. For a 4-way intersection there are 4 possible actions and each action is a phase of the traffic signal that is North Advance, East Advance, South Advance and West Advance. Similarity 3-way will have 3 actions in its action set and 5-way will have 5 actions in its action set.

Reward: Reward is something which the agent receives as a consequence of its chosen action. A good action results in a positive reward while a bad action results in a negative reward. We have used change in cumulative waiting time between actions as the metric to calculate the reward.

Learning mechanism: The learning mechanism used is Deep Q-Learning. It is a form of medel free learning, which assigns a Q-value also known as quality value to an action that was performed on the state of the environment.

In literature, the Q-value is obtained from the Bellman's equation defined as in equation (1).

$$Q(s_t, a_t)=Q(s_t, a_t)+\alpha\,(\text{reward} + \gamma \cdot \max_A Q(s_{t+1}, a_t) - Q(s_t, a_t)) \quad (1)$$

Here $Q(s_t, a_t)$ is the Q-value for an action a_t taken on state s_t. We are updating the current Q-value with a quantity discounted by the learning rate (α). The immediate future's Q-value is represented by $Q(s_{t+1}, a_t)$ and $\max_A$ tells that among all the possible actions a_t in state s_{t+1}, the one with maximum value is selected. The term γ is the discount factor having a value between 0 and 1 to, this gives more importance to the immediate reward than the future reward.

In Q learning performs well in small state-spaces but its performance falls off when working with complex sophisticated environments. In small state spaces we just need to update a few states iteratively in the q-table, but when we are working in complex and large environments, the state-space is very large and it's computationally inefficient to update each state-action pair in the state space iteratively. Now instead of using value iteration to directly compute the Q values and find the optimal Q function, instead we use a function approximator to estimate the optimal Q function, hence neural networks come into picture. The output Q values from the neural network (one for each action) is compared to the q values obtained from the Bellman's equation to compute loss. This loss is minimised using back propagation and tweaking the weights using stochastic gradient descent using Adam optimizer.

Training Process: We have used the experienced relay technique to enhance performance and efficiency. This technique consists of submitting all the knowledge needed by the agent for the training process as a group or collection of random samples also called as a batch rather than instantly submitting the knowledge gathered by the agent during the phase of simulation. All the samples collected during the training phase are stored in the memory and a group of samples are retrieved from the memory as a batch. Training includes iterative learning of a Q-value function with the help of knowledge available within the batch of samples retrieved from the memory.

In this paper, we have used a slightly modified version of equation (1) which is presented in the form of equation (2).

$$Q(s_t, a_t) = \text{reward} + \gamma \cdot \max_A Q'(s_{t+1}, a_{t+1}) \quad (2)$$

The action-selection policy we used is the epsilon-greedy policy. This is the policy that decides whether an agent performs an exploratory action where the agent tries something new, or performs an exploitative action where the agent exploits what it already knows to achieve favourable rewards.

Further during the testing phase, we tested the trained model with random traffic simulations to see it's performance. Here we also configured our emergency vehicles like ambulances, fire trucks, police vehicles, etc with blue light devices in SUMO to ignore traffic signals so that they don't have to wait unnecessarily. By doing this any vehicle can drive with special rights and change lanes without regard for strategic considerations. All the other vehicles ahead of the emergency vehicles are forced to form a rescue lane to let the emergency vehicle pass through.

IV. Results And Discussion

On visual inspection it was clearly seen that the trained model performed much better than the normal simulation, as the trained model had learnt an optimal policy function to maximize the rewards and choose an optimal action.

Our agents were able to reduce the cumulative delay of vehicles and maximize the rewards during the training process as we can see in the below graphs (Fig. 2 & Fig. 3).

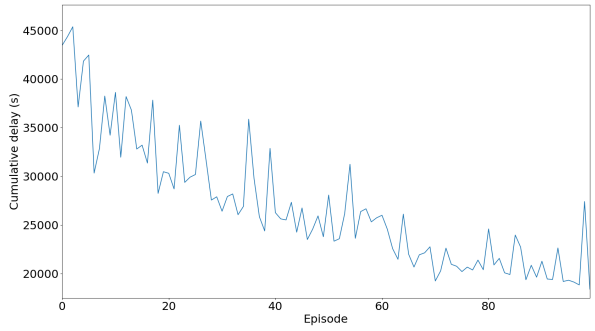


Fig. 2. Cumulative delay(s) vs. Episode graph

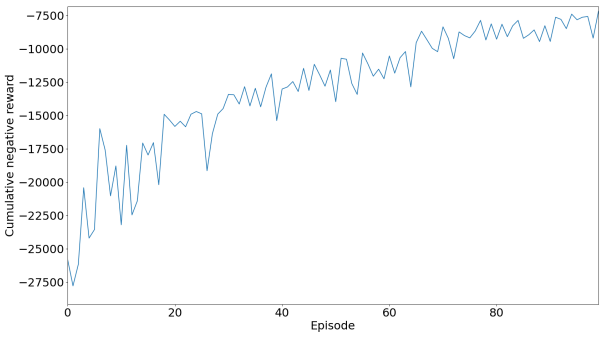


Fig. 3. Cumulative negative reward vs. Episode graph

In comparison with traditional traffic signals it was seen that the trained agent was able to reduce queuing in traffic signals significantly as seen in Fig. 4.

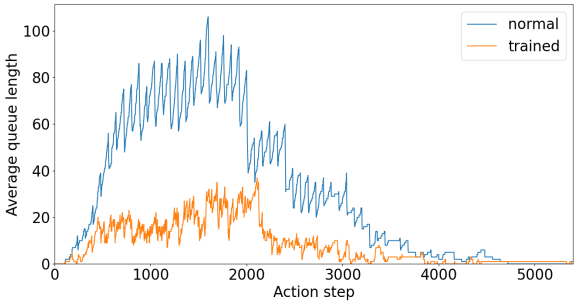


Fig. 4. Average queue length vs. Action step graph (normal vs. trained)

The reason for this reduction in queue lengths as seen in Fig. 4 is the duration of the traffic light phases. Here we have chosen a normal scenario wherein the duration of green and yellow phases is 30 seconds. During training, apart from the agent choosing an appropriate action, we are making sure that the green light duration for a particular lane does not exceed 10 seconds and the yellow light duration does not exceed 4 seconds. By doing this it allows the agent to take quicker actions depending on the surroundings. In case the action chosen at actionstep t is the same as the action taken at actionstep t-1, then the green light persists and there is no yellow phase, hence there is only 10 seconds (10 simulation steps between the two actions). Else if the consecutive actions are not the same then there is a 4 second yellow phase between the two actions making it a total of 14 seconds (14 simulation steps) between the two actions.

On analysing the performance of the agents, it was seen that the agents trained using the CNN performed much better than the agents trained using the artificial neural network as seen in Fig. 5, Fig. 6, Fig. 7 & Fig. 8.

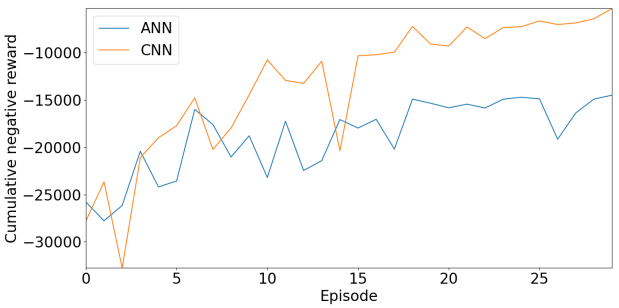


Fig. 5. Cumulative negative reward vs. Episode (ANN and CNN)

On analysis of the reward trend we can see that the CNN is able to maximise the rewards much quicker compared to ANN with lesser training (Fig. 5).

When cumulative delay is plotted for both ANN and CNN, we can clearly see that CNN has lesser cumulative delay when compared to ANN (Fig. 6).

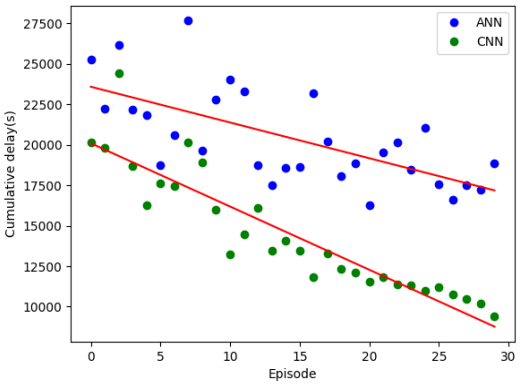


Fig. 6. Cumulative delay(s) vs. Episode (ANN and CNN)

As we can see from Fig. 7 and Fig. 8 the CNN agent is able to increase the negative reward and reduce the average queue length much quicker than the ANN as the number of episodes increases.

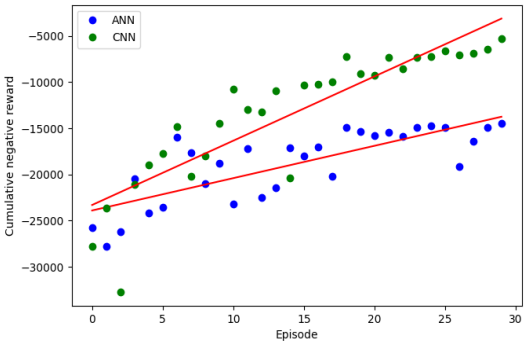


Fig. 7. Cumulative negative reward vs. Episode (ANN and CNN)

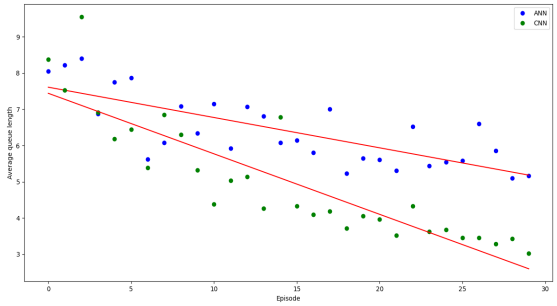


Fig. 8. Average queue length vs. Episode (ANN and CNN)

The reason why CNN performs much better compared to ANN is because of its ability to extract important features using DTSE. This allows CNN to develop high-level state representations. Moreover the CNN takes multiple data inputs, in our case the position and velocity matrix compared to the position matrix alone in the ANN. This allows CNNs to achieve better results with higher accuracy. Although CNN takes longer to train, it gives efficient results much quicker.

It was also seen that the models performed better in smaller intersections than the bigger ones (Fig. 9 & Fig. 10).

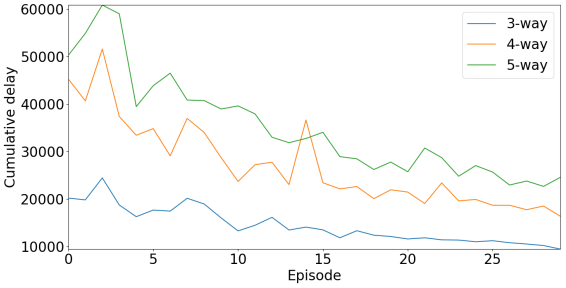


Fig. 9. Cumulative delay(s) vs. Episode

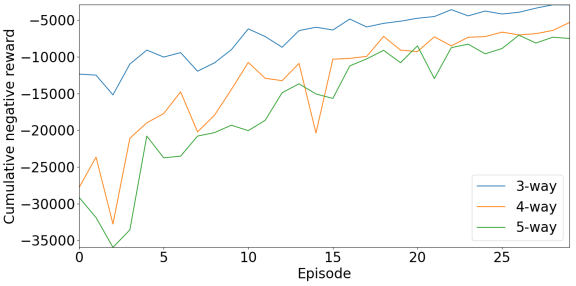


Fig. 10. Cumulative negative reward vs. Episode (3-way, 4-way and 5-way)

This is mainly due to the fact that in smaller intersections, the state space representation is smaller compared to larger intersections and the model needs to choose from a smaller action set making it easier and quicker to train.

In complex intersection networks, when one model is undertrained, we found that it performs worse when compared to the fully trained models. The queue_length keeps on increasing for the partially trained model because of the congestion in the undertrained network. This can be seen from the plot below (Fig. 11).

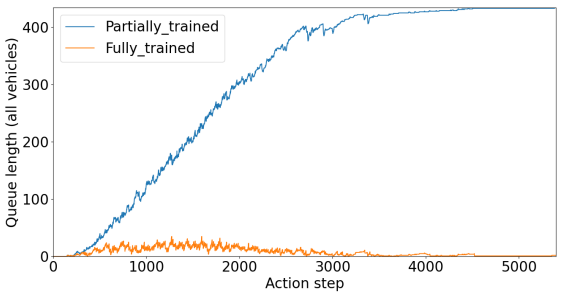


Fig. 11. Queue length vs. Action Step (Fully trained vs partially trained)

The above figure (Fig. 11) shows that while training for complex networks, each model has to be trained fully to achieve better results. Even if one of them is undertrained, the whole network will perform worse, since each individual network is interconnected and queue_length in one network impacts the whole network.

V. Conclusion

The work carried out presented a RL approach to monitor and control traffic lights in a much more effective manner wherein they are able to adjust dynamically to real time traffic and take appropriate actions. This work was carried out on a traffic simulator which provides a realistic environment to carry out complex simulations. The agents were trained using the DQN mechanism and its performance was analysed using various parameters. It was important to properly understand the application context and apply them to machine learning models to achieve proper results.

This work has shown that proper use of Machine learning approaches can provide effective and excellent results. Reinforcement learning, despite its limited application in the real world, is still one of the hot topics in the field of Artificial Intelligence and could potentially have a great impact in the future.

There are however many things that can be improved in this work. Future works could include improving the results achieved and expanding it to a much larger network. The training of the agents could be improved by using different techniques such as target network, clipping rewards and skipping frames. We can also try using different reward and Q functions to try and improve the model further. Proper analysis is important to understand the advantages and possible disadvantages that it can bring on introduction to the real world.

Acknowledgment

The authors would like to thank all the authors of the various sources that are used in this project. The authors would also like to thank PES University for their support and encouragement throughout this project and acknowledge the support of friends and family.

References

[1] Andrea Vidali, Luca Crociani, Giuseppe Vizzari, Stefania Bandini, “A Deep Reinforcement Learning Approach to Adaptive Traffic Lights Management”.

[2] Wade Genders , Saiedeh Razavi, “Using a Deep Reinforcement Learning Agent for Traffic Signal Control”.

[3] Hua Wei, Guanjie Zheng, Huaxiu Yao, Zhenhui Li, “IntelliLight: A Reinforcement Learning Approach for Intelligent Traffic Light Control”.

[4] Arel C. Liu1 T. Urbanik, A.G. Kohls, “Reinforcement learning-based multi-agent system for network traffic signal control”

[5] SUMO documentation : https://sumo.dlr.de/docs/ Copyright © 2001-2021 German Aerospace Center (DLR) and others.

[6] R. S. Sutton, A. G. Barto et al. Introduction to reinforcement learning. MIT press Cambridge, 1998, vol. 135.

[7] Monireh Abdoos, Nasser Mozayani, and Ana LC Bazzan. 2013. Holonic multi-agent system for traffic signals control. Engineering Applications of Artificial Intelligence 26, 5 (2013), 1575–1587.

[8] Baher Abdulhai, Rob Pringle, and Grigoris J Karakoulas. 2003. Reinforcement learning for true adaptive traffic signal control. Journal of Transportation Engineering 129, 3 (2003), 278–285.

[9] T. L. Thorpe and C. W. Anderson, “Traffic light control using sarsa with three state representations,” Citeseer, Tech. Rep., 1996.

[10] M. Wiering et al., “Multi-agent reinforcement learning for traffic light control,” in ICML, 2000, pp. 1151–1158.

[11] Y. K. Chin, N. Bolong, A. Kiring, S. S. Yang, and K. T. K. Teo, “Q-learning based traffic optimization in management of signal timing plan,” International Journal of Simulation, Systems, Science & Technology, vol. 12, no. 3, pp. 29–35, 2011.

[12] M. Abdoos, N. Mozayani, and A. L. Bazzan, “Holonic multi-agent system for traffic signals control,” Engineering Applications of Artificial Intelligence, vol. 26, no. 5, pp. 1575–1587, 2013.

[13] I. Arel, C. Liu, T. Urbanik, and A. Kohls, “Reinforcement learning-based multi-agent system for network traffic signal control,” IET Intelligent Transport Systems, vol. 4, no. 2, pp. 128–135, 2010.