Traffic Signal Control Using Machine Learning

Article *in* International Journal of Mechanical and Production Engineering Research and Development · September 2020

DOI: 10.24247/ijmperdjun20201040

5 authors, including:

Josin Hippolitus
SRM Institute of Science and Technology
4 PUBLICATIONS **4** CITATIONS
SEE PROFILE

Sanjay Sharathi
SRM Institute of Science and Technology
1 PUBLICATION **4** CITATIONS
SEE PROFILE

International Journal of Mechanical and Production Engineering Research and Development (IJMPERD)
ISSN(P): 2249–6890; ISSN(E): 2249–8001
Vol. 10, Issue 3, Jun 2020, 10849–10872

TRAFFIC SIGNAL CONTROL USING MACHINE LEARNING

JOSIN HIPPOLITUS A[1], NAMITA JOANNA VICTOR[2], RONAK BOGAVALLI[3], SANJAY S[4] & ATHTHEN PREMKUMAR[5]

[1]*Assistant Professor, Department of Mechatronics Engineering, SRM Institute of Science and Technology, India*

[2,3,4,5]*Department of Mechatronics Engineering, SRM Institute of Science and Technology, India*

ABSTRACT

Traffic signal has been a long-standing topic in urban traffic control. Ineffective and inflexible traffic control at urban intersections can often lead to an obstruction in traffic flow and will definitely lead to congestion of traffic. How we manage the traffic in a smart manner is a big challenge in urban traffic management. If we introduce new techniques with ways of solving this problem then it will benefit the urban areas. As traffic light systems are everywhere it would be tough to change the system and we also need to cross certain barriers to achieve this task and instead of changing the current system and bringing up new things if we just solve this issue by making some changes in the existing software then it would cut down a lot of other barriers too. All we need to do is just add some algorithms to main software. With recent advances in machine learning, especially reinforcement learning (RL), traffic signal control using advanced machine learning techniques represents a promising solution to tackle this problem. The performance of the proposed method is comprehensively compared with two traditional alternatives for controlling traffic lights. Simulation results indicate that the proposed method significantly reduces the total delay in the network when compared to the alternative methods. Adjacent traffic light intersections will work independently and yet cooperate with each other to a common goal of ensuring the fluency of the traffic flow within traffic network. The experimental results show that the Q-Learning algorithm is able to learn from the dynamic traffic flow and optimized the traffic flow.

KEYWORDS: Reinforcement Learning; Q Learning; Traffic Signal Management

Received: Jun 08, 2020; **Accepted:** Jun 28, 2020; **Published:** Sep 03, 2020; **Paper Id.:** IJMPERDJUN20201040

Original Article

1. INTRODUCTION

In the recent years due to technological advancements the automobile sector is also manufacturing a higher number of vehicles and every year almost 253 million vehicles are being manufactured and sold to customers. As the roads cannot expand, we need to look for efficient ways in order to manage the number vehicles on the road. A fully developed city has a limited ability to reconnect to its traffic, thus providing a real time solution to this problem may solve most of the problem. Providing a real time traffic control based on the density of vehicles could be the best solution so far. The main purpose of controlling traffic signal is to simultaneously increase the intersection capacity, decrease delays and guarantee the safety of people and vehicles. It can also help in reducing the unwanted time which people waste by staying at signals. Moreover, at peak hours and at special cases the traffic signals will be working on a real time-based approach instead of the older ways. After a lot of research, it is found out that the main purpose for the accumulation of traffic is due to fixed-time traffic control. In a fixed-time traffic control it has been observed that every part of the road is given an equal time for the movement of vehicles. But by giving an equal amount of time to every part of the road, the traffic congestion does not tend to reduce the amount of traffic. Our project deals with bringing in change with the current traffic signal system and moreover making it real time-based and to glow the green light for the part of the road containing a higher number of vehicles. According to the environment and the test cases the machine will bring out the

most effective solution for this problem and will get better after every try as we have used reinforced learning and the machine will also learn from the environment as it is a reward-based learning.

Figure 2 shows Conventional traffic signal timing plan management using statistical information of traffic lacks the ability to rapidly adapt into the dynamic traffic flow. Thus, the necessity for the development of intelligent traffic signal timing plan management is a need to continuously learn from the dynamic traffic environment for adaptability. The Q-learning algorithm gathers information of the past actions and tends to learn better and also learns from its environment. The implemented traffic light intersections will be able to learn from the current traffic light intersections and its environment for increasing its adaptability and tends to make a better decision in the future using the Q-learning algorithm [12].

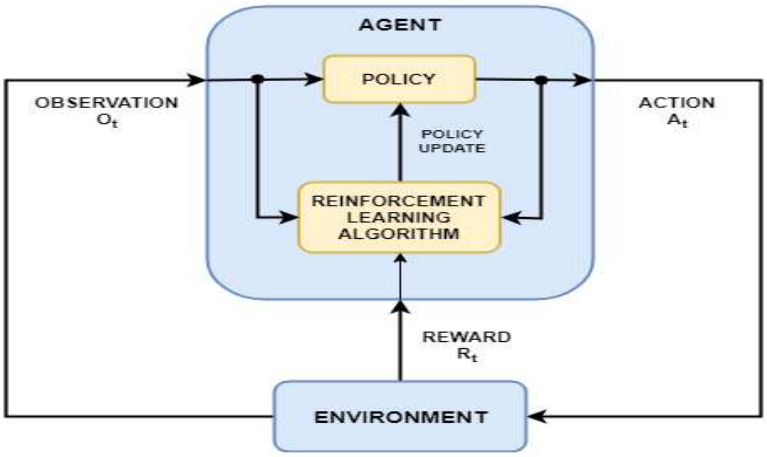


Figure 1: Structure of Reinforcement Learning.

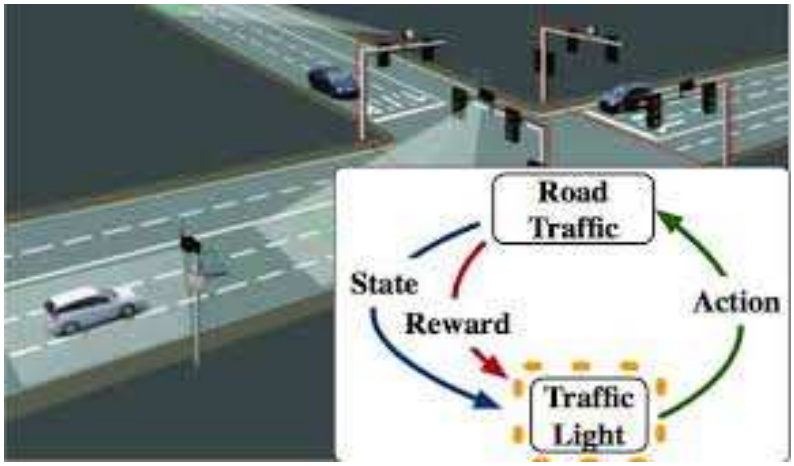


Figure 2

1.1. Description of the Project

1.1.1. Existing System

The traffic control lights currently are fixed in a sequence and with a time delay which follows a particular cycle while switching from one signal to other.

Sometimes this might cause a lot of congestion on the roads, mainly when there are many vehicles at some part of the road. The sequence of the traffic signal causes the green light to glow for the part of the road where there are very few vehicles.

1.1.2. Proposed System

Our project is a density-based traffic control system using reinforced learning to solve this problem.

We bring in a slight change to the traffic signal system by making it priority based when there is a huge amount of traffic and then switching it back to the normal sequence after there is less amount of traffic.

The system counts the number of vehicles on each part of the road and after the analysis the system takes an appropriate decision as to which road is to be given the highest priority and the longest delay for the corresponding traffic light.

1.2. Problem Formulation

The problem with the existing traffic system is that for every minute the vehicles at the 4-way junction (cross-roads) will be large in number due to this there is always a lot of congestion on such roads. Even though there are no vehicles at a particular side, the traffic signal will still glow green for a fixed time. This causes a lot of problems such as:

- The vehicles on the other part of the road have to keep waiting for their turn.
- The congestion on the roads after a point gets even worse.
- In some cases, most of the ambulances cannot move and get stuck in the congestion.

2. OBJECTIVES

This project aims to reduce traffic congestion and unwanted long time delays and provides a better approach to this by calculating the density of the traffic at each part of the road and simultaneously provides the best solution in order to reduce the congestion and in some cases to stop giving a green light indication to some roads where there are less number of vehicles.

3. OVERVIEW

The overview of this project is to implement Density-Based Traffic control using machine learning. In a nutshell, machine learning is all about learning a highly accurate predictive or classifier model, or finding unknown patterns in data, by leveraging learning algorithms and optimization techniques. It is a very vital topic for many key reasons because it provides the ability to obtain deep insights and despite the popularity of the subject, the true purpose of machine learning is obtained from the prediction models and data.

4. PURPOSE

The purpose of the current work is to analyze and study the counting and control system via machine learning. As there are already cameras on the roads, we could use them for retrieving the live data of vehicles and thus the vehicles could be identified by the cameras as the structure of vehicles have already been labeled before and this could be used for the vehicle counting.

5. SCOPE

The current work focuses on how to bring an effective solution to this problem:

- Reduce the traffic congestion.

- Avoid road blocks and prevent accidents from happening by effectively analyzing vehicles every now and then.
- Reduce waiting period for the vehicles and people and also save fuel.

5.1. Factors Affecting The System

A specific interchange on a major road is known as a junction. Traffic jams mostly occur at the traffic junctions mainly at the three-road and four-road junctions. When the flow of vehicles increases from all sides then it definitely causes traffic congestion and too many vehicles tend to wait for a long period of time till the traffic reduces and sometimes it may seem longer than is estimated.

5.2. Business and Peak Hours

Traffic congestion occurs mainly during peak hours every day mainly between 9-11 am in the morning and 5-7 pm in the evening. It is mainly because most of the employees travel to their workplaces during 9-11 am and travel back to their respective houses at 5-7 pm. It is because the volume of traffic or modal split generates the demand for space greater than the available street capacity.

6. REINFORCEMENT LEARNING

The idea that we learn by interacting with our environment is probably the first to occur to us when we think about the nature of learning. When an infant plays, waves his arms, or looks about, it has no explicit teacher, but it does have a direct sensor motor connection to its environment. Exercising this connection produces a wealth of information about cause and effect, about the consequences of actions, and about what to do in order to achieve goals.In general, we always learn by correcting our previous mistakes and we tend to get better or sometimes if we are awarded to finish a task, we tend to do it better. To know it better let's first break the two words.

Reinforcement: The action or process of reinforcing or strengthening.

Learning: The acquisition of knowledge or skills through study, experience, or being taught.

When we combine these two words that are 'REINFORCEMENT + LEARNING' we tend to understand this better. Reinforcement learning is a reward-based learning for which when every task is performed and if it is a success a reward is given to the performer if he fails to achieve it, he's not awarded anything. This helps the performer to increase its efficiency as he gets better. Similarly, we adapt to changes in and around us. Reinforcement learning teaches a performer how to adapt to changes with respect to the environment. Reinforcement learning is a very efficient form of learning because a performer is made to learn from was own mistakes.

Therefore, the person or the system performing this task automatically becomes better as it is done on a regular basis. Reinforcement learning is far different from machine learning (supervised learning). In supervised learning the system tries to find structures hidden in collections of unlabelled data. Reinforcement learning finds a reward signal instead of a structure. Uncovering structure in an agent's experience can certainly be useful in reinforcement learning, but by itself does not address the reinforcement learning agent's problem of maximizing a reward signal. We therefore consider reinforcement learning to be a third machine learning paradigm, alongside of supervised learning, unsupervised learning, and perhaps other paradigms as well. One of the challenges that arise in reinforcement learning, and not in other kinds of learning, is the trade-off between exploration and exploitation. To obtain a lot of reward, a reinforcement learning agent must prefer actions that it has tried in the past and

found to be effective in producing reward. Reinforcement learning takes the opposite tack, starting with a complete, interactive, goal-seeking agent. All reinforcement learning agents have explicit goals, can sense aspects of their environments, and can choose actions to influence their environments. One of the most exciting aspects of modern reinforcement learning is its substantive and fruitful interactions with other engineering and scientific disciplines. Reinforcement learning is part of a decades-long trend within artificial intelligence and machine learning toward greater integration with statistics, optimization, and other mathematical subjects [10]. For example, the ability of some reinforcement learning methods to learn with parameterized approximates addresses the classical "curse of dimensionality" in operations research and control theory. More distinctively, reinforcement learning has also interacted strongly with psychology and neuroscience, with substantial benefits going both ways. Of all the forms of machine learning, reinforcement learning is the closest to the kind of learning that humans and other animals do, and many of the core algorithms of reinforcement learning were originally inspired by biological learning systems. And reinforcement learning has also given back, both through a psychological model of animal learning that better matches some of the empirical data, and through an influential model of parts of the brain's reward system.

6.1. Interactions between Agent & Environment

An agent's environment may well include other agents. An agent together with its environment is called a world. An agent could be, for example, a coupling of a computational engine with physical sensors and actuators, called a robot, where the environment is a physical setting. It could be the coupling of an advice-giving computer--an expert system with a human who provides perceptual information and carries out the task. An agent could be a program that acts in a purely computational environment a software agent

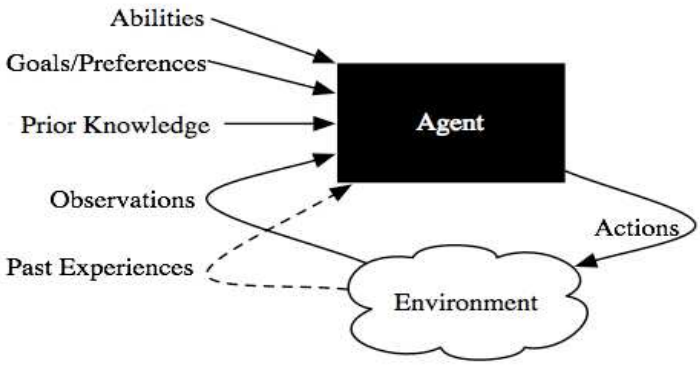


Figure 3: Interactions Between Agent & Environment.

6.2. Uncertainty about the Environment

Environmental uncertainty is when conditions are constantly changing within a business environment. As a result, management has little influence over factors that are outside of the company's control. For example, the economy could collapse at any time. This would impact the company. New technology could alter the landscape within a given business. One concern that some businesses have today is a shortage of skilled workers to do the work the company needs done in order to make its products. The company may have little control over these circumstances. Companies need to prepare for any changes that might occur so they can quickly respond to these changes.

6.3. Defined Goal

A goal should specify what we want to achieve, not how we want to achieve it. A goal must be outside the agent's direct

control thus outside the agent the agent must be able to measure success explicitly frequently during its lifespan. Its goal and usage are to build new or revise existing algorithms to learn from the given data in order to build better models that give higher predictions compared to the existing systems and find patterns and particularly with new data.

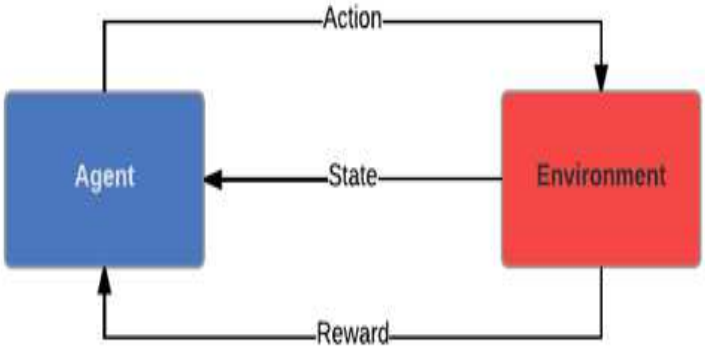


Figure 4: Description of Uncertainty.

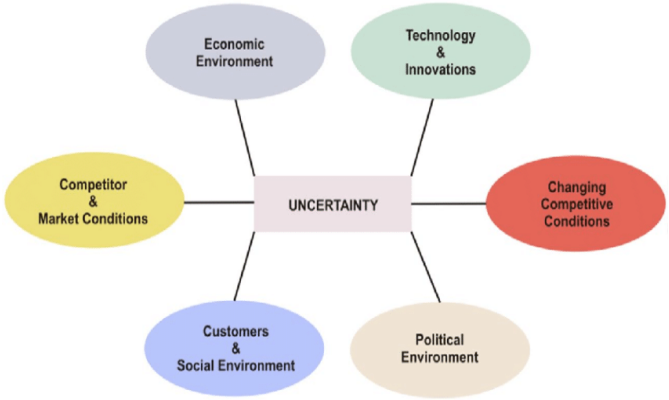


Figure 5: Linkage between Agent and Environment.

6.4. Experience Improves Performance

The typical framing of a Reinforcement Learning (RL) scenario: an agent takes actions in an environment, which is interpreted into a reward and a representation of the state, which are fed back into the agent.

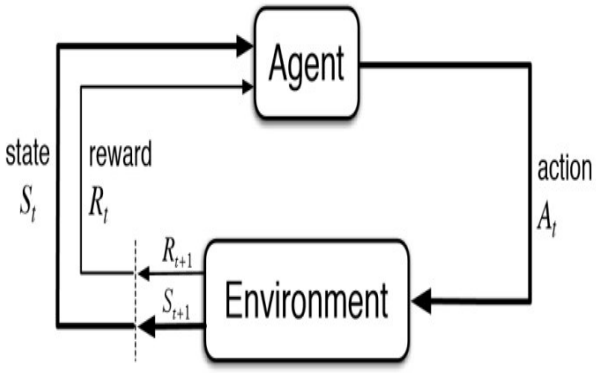


Figure 6: Reward Based Approach.

7. DESIGN OF THE SYSTEM

This project intends to design a system which uses deep neural network algorithm which is a subset of artificial intelligence, which will provide intelligence to the current traffic control system present at a four-way junction. This system is mainly aimed to replace the conventional timer traffic control system with our artificial intelligence system. Nowadays most cities are equipped with CCTV cameras on the roads and the junctions, the basic idea is to collect the live video from the CCTV cameras and detect the number of vehicles [11] on each lane and feed the data into another machine learning algorithm which according to the data of each lane changes the light phase of the signal. This system mainly aims to increase the traffic efficiency by increasing vehicle flow which will reduce waiting time for the vehicles.

Figure 7: Workflow of the System.

7.1. Detecting Vehicles

To detect the number of vehicles we used neural network algorithm as the basis of the design. Framework for the neural networks is must before starting to design the algorithm. We used Tensor Flow framework and Keras framework to create a neural network which will detect number of vehicles. A convolution neural network is used which is one type of neural network. The datasets will be fed into the designed neural network so to train the neural network in order to get highly accurate results.

7.2. Convolution Neural Network

The convolution neural network is one of the class of neural networks which was chosen to design an algorithm to detect vehicles, this was chosen instead of the fully connected neural network because there will be a compromise in spatial structure of the image (frame) because it will be impossible to connect neurons to all the neurons of the previous volume. The network is designed by adding layers; each layer has a different job, which is grouped up to give a desired output. These layers are used in feature extraction.

The Layers Are:

- Convolution layer
- Pooling layer
- Activation
- Fully connected layer

7.3. Convolution Layer

Figure 8 shows the layer consists of a filter used for feature extraction; the filter is a square matrix with weights. The size of the filter is much smaller than the image (frame) matrix. The filter will slide through the image matrix for a feature. The output neuron will be element wise multiplication of filter matrix and the image matrix in which the filter is and addition of all the elements. To get more features more filters should be added, more features will increase the accuracy for detecting vehicles.

Also, from Figure 8 it can be visualized how the filter multiplication and addition takes place and proceeds to next neuron.

Figure 9 shows the green color is the image matrix and the yellow color is the filter, it is visible that the filter slides through the image matrix and at each instant element wise multiplication happens and the elements are added. Likewise, each result is put in a matrix called feature map

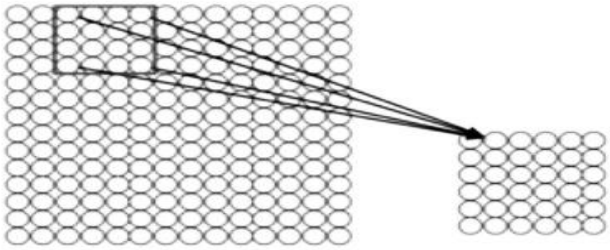


Figure 8: Filter Representation.

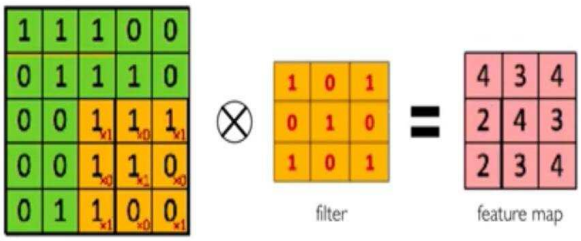


Figure 9: Filter Multiplication.

7.4. Activation Layer

In our design we used ReLU activation function, its abbreviation is rectified linear unit. It is highly used out of other activation functions. This function removes zero and gives only non-zero values, also it does not affect the convolution layer.

The Function Is Taken As:

$$y=\max(0,x) \quad (1)$$

7.5. Pooling Layer

To deal with different spatial resolution pooling is done, it is the non-linear Down sampling or reducing the dimensionality of the input layer, it is done after convolution. There are many functions among them we have chosen max pooling which is used most often. Patches are created in the input layers, no two patches intersect, and maximum value from each patch is

selected and formed a layer which can be passed to a convolution layer. By this method the spatial size reduces so the number of parameters reduces which makes the process faster.

7.6. Fully-Connected Layer

The above three layers will be repeated according to the design necessities, once that is done the final two-dimensional layer will be flattened that is converted to one-dimension. Now it will proceed like a fully-connected neural network with hidden layers and an output layer with the one-dimension input. Here high-level reasoning will be done. The number of hidden layers and neurons will be chosen according to design requirements.

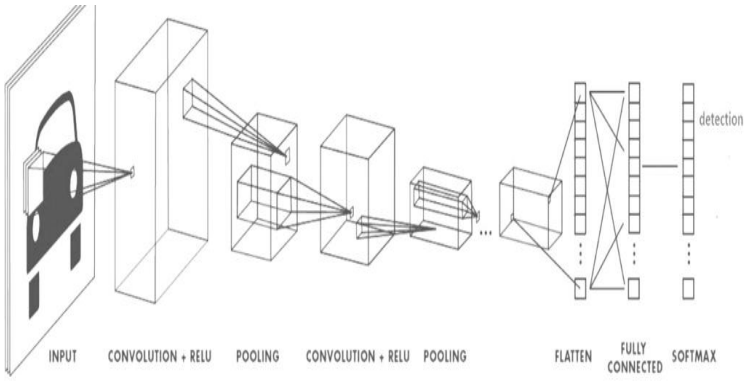


Figure 10: A Pictorial Representation of Neural Network Layers.

7.7. Additional

Figure 11 shows Addition to the above design we have two add-ons, one is we detect ambulance and the phase of the signal changes giving priority to ambulance first. When more than one ambulance nearing a junction then the one which is closest will get first priority then immediately the second one will be allowed.

Figure 12 shows the second is in each arm of the intersection where there will be multiple imaginary lines horizontal to the incoming road separates road into cells of unequal size, the cell closer to signal will be small and farther will be long. These cells describe the situation of traffic in each arm of the intersection.

All this information and the data collected will fed into the control algorithm which will make decisions accordingly increasing traffic efficiency and increased vehicle flow.

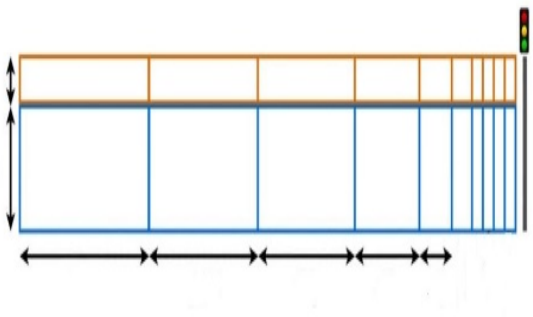


Figure 11: A Representation the Cells in Each Arm.

7.8. Control Algorithm Using Reinforcement Learning

This is our second part of our whole system design; in this a control algorithm [4] is designed to control the traffic signals at the junction. Reinforcement learning is completely different from the other two supervised and unsupervised learning. Reinforcement learning is one of the paradigms of machine learning is used to design the control algorithm; this will be used to take real action on the environment using an agent. There are set of variables in reinforcement learning which are not in supervised or unsupervised earning methods they are state *s*, agent *a*, and reward *r*. Agent is the central part in the reinforcement algorithm it is the neural network and it is the thing going to take actions on the environment. The environment is the place in which the agents operates or take actions.

Figure 13 shows here the four-way junction is the environment and neural network is the agent which will take action on it with respect to the state of the environment. An agent in real life is nothing but us(people) so the agent for this traffic problem will be a traffic policeman, who during high traffic looks in which side more vehicles are there and let them cross the junction. State is a concrete or immediate situation that the agent finds itself in, a situation. In which the agent perceives it could be in any form image, video, or sound. The agent which is the neural network is designed mainly for higher traffic conditions so even at high traffic problems this will work with very good accuracy and mainly will decrease congestions in the road during working days. The data will be feed into this algorithm directly, for visualizing purpose that real-time data can be simulated and fed into this control algorithm.

We will provide the agent the all possible actions it can make on the environment, so in each arm of the intersection there are two possible actions those are:

For Left-Lane Driving Countries

- Going straight and left
- Right

For Right-Lane Driving Countries

- Going straight and right
- Left

So, there are totally eight possible actions that can be performed by the agent over the environment. These actions will be performed over a time period according to the state of the environment.

For an action the state of the environment changes, suppose the phase of the signal of one arm of the intersection changes from red to green and then to red, due to this action the state of the environment will change with respect to the previous state so now the agent will take actions according to the present state of the environment. We have designed an agent which will take actions according. For every action the agent will get a reward from the environment

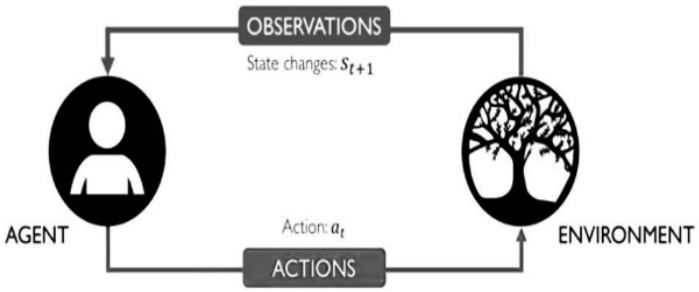


Figure 12: Control Algorithm.

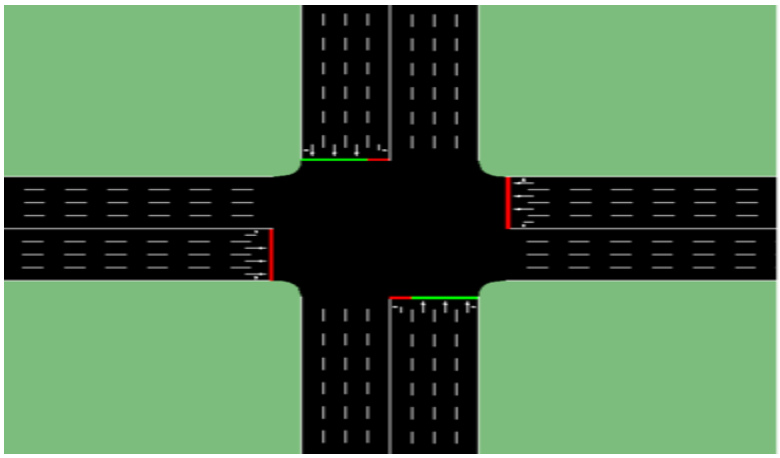


Figure 13: Simulated Environment.

7.9. Reward

The reward is nothing but a feedback that measures success or failure of the agent's action, for example the environment will give positive reward for good actions and negative reward for bad action of the agent over the environment [1]. The agent tries to maximize its reward every time. So, reward is an important factor in this design. Since the main objective of this project is to maximize vehicle flow, the reward should be given with a performance measure of traffic efficiency so the agent will know if its action is increasing or decreasing vehicle flow. In our project the measures to give reward are the following.

- Number of vehicles that have crossed the junction over a period of time t.
- The number of vehicles with speed less than 0.35 kilometer per hour
- The average waiting time of the number of vehicles waiting at a particular time.
- The average waiting time for set of vehicles closer to the junction should not cross the threshold waiting time. Threshold value varies with region

8. TRAINING AND SIMULATIONS

An explanation looking into the description of the agent comprising of the state, all the probable actions and the reward was given in the previous chapter. Here the strategies and methods that were used to ensure a continuous flow of traffic throughout the simulation of this four-way intersection will be discussed. For the simulation the software SUMO was used

to recreate a four-way junction. The way the agent operates per time step is shown in Figure 14.

The agent waits for the simulation to complete a specific number of steps before it proceeds to initiate its time step t. The agent first scopes for the state of the environment, while doing this it also calculates the delay times [13]. These delay times are essentially a waiting period between the previous state (t – 1) and the current state (t), so this delay will influence the reward function to be calculated. The reward is linked to the action that was taken at the time step t – 1. After this is completed, the newly learned information is collected and saved to a memory which can then be used to train the model to choose a more effective action in the future. In the final step, the agent selects a new action to be set to the environment so that the simulation can initiate a new episode [13]. The main goal of the agent is to retrieve a good reward from the actions chosen, this can only be done if the model is able to determine what the most beneficial course of action would be. In this chapter, the experience parameters used to train the model and guide its selection of action is discussed in depth.

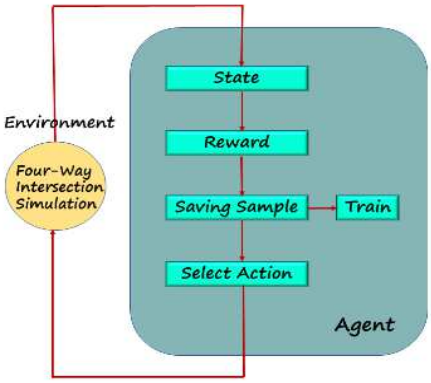


Figure 14: Operation of the Agent per Time Step.

8.1. Optimization of Experience Replay

Experience replay allows a reinforcement learning agent to recollect information from its memory and reprocess these past experiences. It uses various methods of sampling to help group up elements of its memory. It improves the learning rate and performance of the agent. The dataset that comprises of the experiences for each time step is known as the replay memory. The replay memory consists of every sample collected during the training session. Let m be a sample taken from the dataset, it is defined as a tuple (2). This tuple obtains information on the current state and the future state of the environment. The variables of the current state influence the variables of the future state.

$$m = \{s_t, a_t, r_{t+1}, s_{t+1}\} \quad (2)$$

Where, s_t refers to the state of the environment, a_t is the action of that state, r_{t+1} is the reward given to the agent as a result of the previous state-action pair and s_{t+1} is the next state of the environment. Using all of these variables, this tuple gives a summary of the agent's experience at a time step t. During a training session, a group of samples is gathered from the memory to train a neural network using the previously mentioned samples i.e., at every time step a random sample is taken from the memory to use to determine the state and the action of the environment. Figure 4.2 shows an illustration of the exchanges with the memory. This ultimately influences the reward value of the episode and also changes the learning rate. A new environment state is defined.

Figure 15 shows the replay memory is always set to a finite size limit within which all the samples are stored for playback. The size of the memory depends on the number of samples that can be stored in the replay memory. Here the

memory size is set to 40000 samples and it is not possible to store anything beyond this limit. The information retrieved from the memory is taken in batches [8], which are a group of randomized samples that are taken directly from the memory. These batches are also contained to a certain size limit. The size of a batch usually depends on the number of samples that are taken from the replay memory in one instance of training. If the memory at a particular time step is full then the first sample in the memory space is detached so that the incoming sample will have adequate space.

Replay memory has an important role in this method as it breaks any correlation between consecutive samples. The reason we randomize the recollected data is to avoid a high correlation of samples since this would lead to inefficient learning. When samples are taken sequentially, the model tends to learn only based on its consecutive experiences. This means that as the training process continues there is a tendency for the model to forget a past experience. To avoid such an occurrence the randomized samples, refresh the experiences so the agent will learn better.

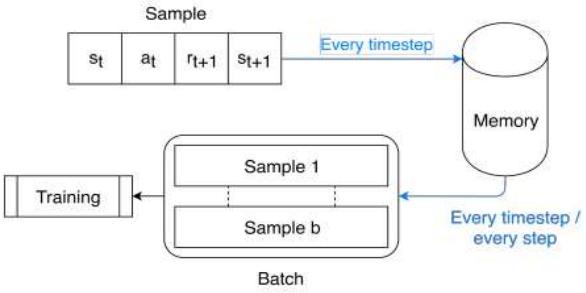


Figure 15: Sampling From Memory for Training.

8.2. Training Procedure

The learning process uses Q-learning to derive a function $Q(s_t,a_t)$ [9]. This function estimates the best course of action a_t it can take in a state s_t to increase the reward of the agent. The variables required to compute the Bellman equation are obtained from a sequence of steps that are executed for every episode of the training.

- A sample is taken from the dataset so that the agent can provide information on the reward and the next state of the environment (Let the sample be m).
- Every training instance depends on the four variables obtained from the sampling.
- The Q-table is initialized randomly and the agent interacts with the environment.
- Upon each interaction, the agent will observe the reward of its action and the state transition.
- The agent then computes its observed Q-value and updates its new estimate of $Q(s_t,a_t)$.

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

- r_{t+1} is taken as the reward estimated for the action a_t and $\max_A Q'(s_{t+1},a_{t+1})$ is derived from a prediction and is the maximum expected future reward.

  In Figure 18 we see that with every episode the reward value gets updated along with the learning rate and Q-value, this influences the next episode and the next action is chosen accordingly.

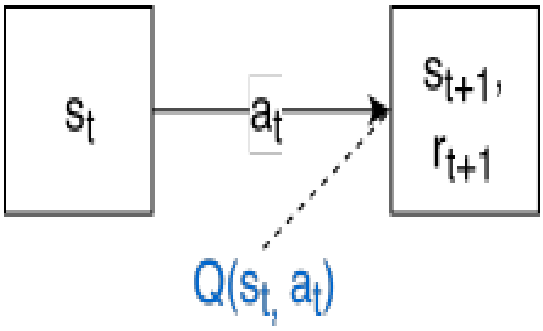


Figure 16: Calculation of Q-Value for a Single Sample.

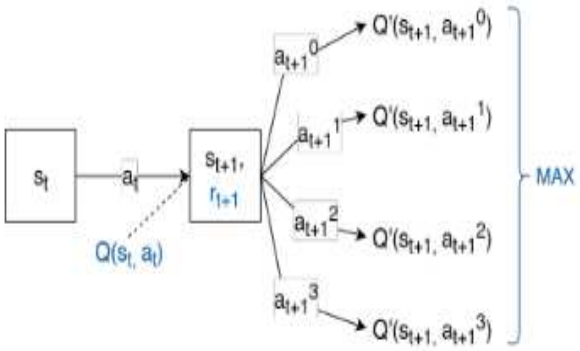


Figure 17: Updated Q-Value Using Bellman Equation.

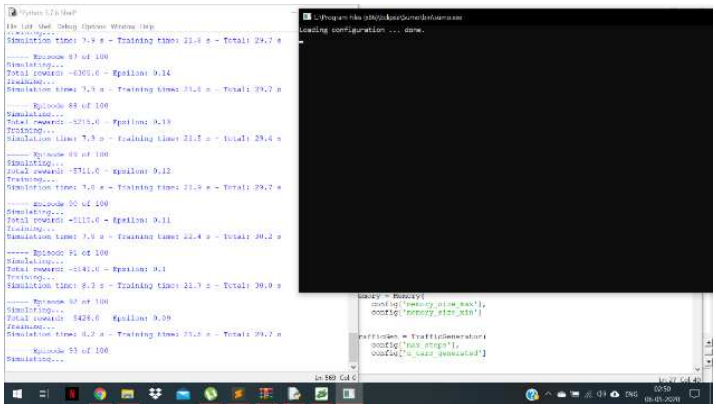


Figure 18: Real-Time Updates from Training.

8.3. Explore-Exploit Dilemma

In the training procedure, it is observed how the Q-value is estimated and updated using the Bellman equation. While this method works for simple algorithms, there is a problem faced in this model. The model is not exploring its options. It remains constant and gets greedy, only consistently repeating sequences out of which it knows for sure that it will get a good reward. This means that in the case that the model experiences a bad reward from a certain action then it will refrain from using that sequence but this results in the model possibly missing out on a much better reward that could perhaps be achieved from choosing this seemingly less desirable outcome. To solve this something called random exploration is used, that will make the model take a random action rather than the optimal action (This depends on its set probability).

Therefore, it eventually figures out that there is a better reward in this less ideal action. The probability to choose to explore or exploit is given by Є - greedy. So, the model can now randomly explore different actions so it may expand its knowledge and become more confident with its

$$Є_h = 1 - \frac{□}{H} \quad (3)$$

Figure 19 shows where $Є_h$ is the probability of the model to choose to explore, 1 - $Є_h$ is the probability of the model to choose to exploit, h is the episode that is currently training and H is the total number of episodes.

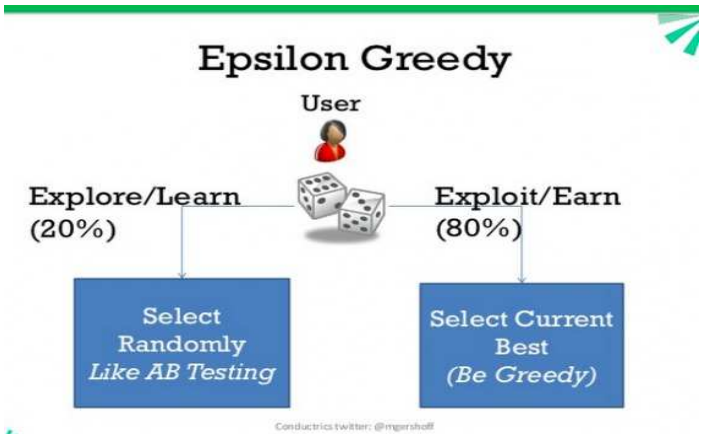


Figure 19: Epsilon Greedy.

8.4. Simulation in SUMO

The simulation for this project was executed on SUMO also known as Simulation of Urban Mobility. This platform is widely used for traffic simulations so that any changes to the traffic setup can be evaluated and implemented. This model uses a four-way intersection to determine the efficiency of this model in different conditions. SUMO has multiple modules that exist so that the required data for simulations can be made accordingly.

The TraCI (Traffic Control Interface) package was what made the simulation of road traffic possible and because of this the agent is able to interact with the environment and retrieve information about its state and make the required decision. Important elements such as traffic lights, the road and its lanes were all made using Net Edit which is a visual editor in SUMO. Net Edit is used to create various network scenarios which in this case is a four- way intersection with vehicles travelling North, South, East and West. This application also uses a GUI tool which helps graphically represents the simulation.

The Figure 20 shows the layout of the network that was made. The roads are sectioned into lanes. These lanes categorize the vehicles based on the direction in which they want to travel. For instance, a vehicle wanting to go straight will line up in the either of the middle lanes or the right most lane, a vehicle wanting to turn left will line up in the left most lane and a vehicle wanting to turn right will line up in the right most lane. The boundary at each arm acts as the signal lighting system in our simulation and will change according to the need. As such it is observed how the previously mentioned tools have helped create the static and dynamic elements of this four-way intersection simulation. Each arm consists of its own cells. This aids the system in its vehicle detection. When a vehicle enters a cell, it is detected by the system. The system then increases its count by one. Using this method, the system keeps track of how many vehicles are currently waiting for a green light. Keeping track of the vehicle count helps the system prioritize an arm that has had a

large number of vehicles waiting for a prolonged period of time [3]. Since the model is able to keep track of this system is able to avoid any and all traffic jams UN lessen countered with some unforeseen circumstances.

Figure 21 is a graph taken from running the simulation for one full test run. Here the x-axis is taken as the actions performed at every step and the y-axis is taken as the rewards given for each corresponding action. Rewards consist of both positive and negative values. The positive values symbolize a good reward. The aim of the model is to get as high a reward as it can. The negative values mean that the model received a bad reward. A bad rewards states that the actions of the model were not convenient and that it should have found a better action to implement on the environment. The observation from this graph is that the model mostly had a positive impact on the system that is to say that there were no problems regarding the possibility of a traffic jam. A few of the scenarios that were played out by the simulation could be further improved.

The model keeps track of exactly how many vehicles are present in each lane. The graph in figure 4.9 shows the number of steps of the sequence on the x-axis and the number of vehicles on the y-axis. The vehicles start of coming onto the roads at a low rate but that slowly increases and the model is presented with a high traffic situation. Every vehicle that lines up to stand at the signal is accounted for and the count for how many vehicles are waiting at a red light increases. Towards the end of the sequence the vehicles tend to lessen in number and continue that way until the end of the sequence.

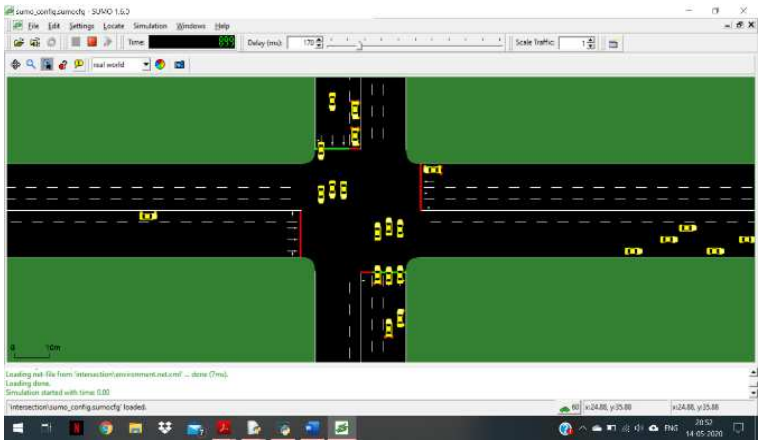


Figure 20: Four-Way Intersection Simulation on SUMO.

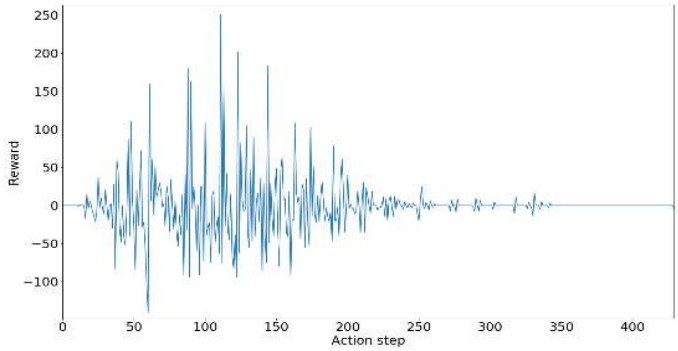


Figure 21: Reward Per Action.

8.5. Production of Traffic

SUMO is used to generate a steady flow of traffic in the simulation [13]. Since the simulation is meant to replicate a real-life road and all its scenarios SUMO needs to follow a distribution that will cater to these needs. The simulation that has

been used was set to replicate a low traffic situation and gradually mitigate into a high level of traffic. One episode is set to go through every possible scenario it can depending on the traffic generated. Each episode will have different sequences. This means that no two sequences will have the same order of traffic scenarios. This is to make sure that the model can learn sufficiently and the agent improves its performance. This way the model learns to avoid Congestions even during a high level of traffic.

The traffic generated resembles the histogram in Figure 22. This histogram shows how the vehicles start out slow, gradually increase, at one point it peaks and then it mellows down until the simulation is stopped. The different scenarios that were mentioned before depend on the starting point of the vehicle and its destination. As previously said every episode will have a different sequence of these scenarios so that the agent will learn better.

Now the probability for a car to need to go straight or turn left or right is the same in a traffic sequence that has a high level of vehicles and one that has a low level of vehicles. The probability of this is exactly 75 % of the vehicles will want to go straight and 25 % of the vehicles will either want to turn left or right. Now there is one more factor to take into consideration here and that is the direction that the vehicle will come from. A vehicle can be found coming from the North, South, East or West. In the simulation the probability that a vehicle could be coming from the North or the South is 90 % and the probability that a vehicle could be coming from the East or the West is 10 %. These four main scenarios will simulate alternately and repeat the sequence every four episodes. SUMO has a random generator and this will produce vehicles that have a different starting point and destination in each episode so every episode will not have the same sequence of vehicle types. This way every episode is different and the model is improved.

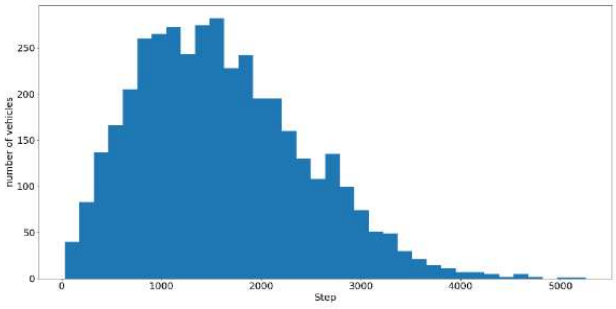


Figure 22: Histogram of Traffic Production in SUMO.

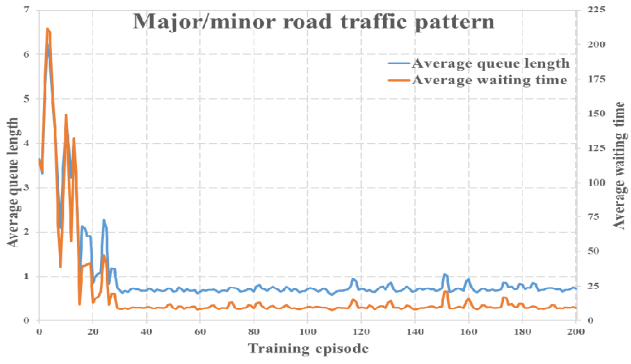


Figure 23: Major / Minor Road Traffic Pattern.

9. RESULTS AND DISCUSSIONS

The Performance of Our Reinforced Learning Approach is evaluated in Three Areas Are:

- Training convergence
- Comparison with benchmarks
- Generalization across different traffic patterns

Average queue length (number of halting vehicles per incoming lane) and average wait time used (wait time in second per incoming vehicle) as a performance criterion

These Figures show the training convergence of our reinforced learning approach under traffic patterns 1 - 3.

Traffic pattern 4 is used for testing purposes to validate the generalization ability of our reinforced learning approach.

At the beginning of the training process, the Q-learning network explores the control policy by selecting random action with high probability.

As training goes on, the Q-learning network gets positive or negative rewards depending upon the weather a corrective action has been taken to reduce the number of halting vehicles. The Q-learning Network gradually exploits the control policy and reduces the average queue length and average wait time [2].

Finally, the Q-learning Network succeeds the stabilized performance with respect to the average queue length and the average wait time.

These graphs show the evaluations on how the control policy are learned by our reinforced learning approach generalizes across different traffic patterns. The entry in the P1 row of the P3 column, it shows the average performance with the reinforced learning approach trained in the traffic pattern P1 and tested in traffic pattern P3. So overall, our reinforced learning approach generalizes well across the different traffic patterns with slight performance variations.

The reinforced learning approach are trained in traffic patterns P1-P3 which feature steady traffic flows and also performs well in the traffic pattern P4 which has a time varying traffic flow.

In patterns 1 – 4 from Figure 26, it shows the performance comparison of our reinforced learning approach with the benchmark of traffic signal control methods.

The box plots in Figure 26 are obtained by repeating each method 100 times in each traffic pattern. The bold line in the middle of the box is the median, the lower line of the box is the lower quartile and the upper line is the upper quartile.

Clearly, the reinforced learning approach is able to achieve a better performance in terms of average queue length and the average wait time in each traffic pattern in terms of the benchmarks.

Figure 26 Shows Even, when it is compared with the second-best benchmark, the performance improvements of the rein forced learning approach are still significant in all traffic patterns

The model performs simulation data to show that our algorithm learns good action policy that effectively reduces vehicle staying time, thus reducing vehicle delay and traffic congestion, and that our algorithm is stable in making control decisions, like not oscillating between good and bad action policies or even diverging to bad action policies [5]. The average values for the sum of staying time of all vehicles at the intersection are shown in Figure 30.

From this Figure, we can see that the average of the sum of vehicle staying time decreases rapidly as the agent is trained for more episodes and finally reduces to some small values which indicates that the agent learns good action policy from the training [6]. We can see that after 800 episodes, average vehicle staying time keeps stable at every small value, indicating that our algorithm tends to meet good action policy and algorithm stabilizing mechanisms and experience replay and target network, work effectively. The average values for the delay of vehicles at each separate road are presented in Figure 31.

From this Figure we see that the average vehicle delay at each road is reduced greatly as the agent is trained for more episodes, by indicating that our algorithm achieves adaptive and efficient traffic signal control. After the agent learns good action policy, the average vehicle delay reduces to small values and stays stable thereafter. From these stable values, we know that our algorithm learns a fair policy where average vehicle delays for roads with different vehicle arrival rates and does not differ too much. This is because long vehicle staying time, thus vehicle delay, at any road leads penalty to the agent, causing the agent to adjust its action policy accordingly [6]. Now, compare the vehicle delay performance of our algorithm with another two popular traffic signal control algorithms, the longest queue first algorithm and fixed time control algorithm, under the simulation" SUMO"'- An open source simulator.

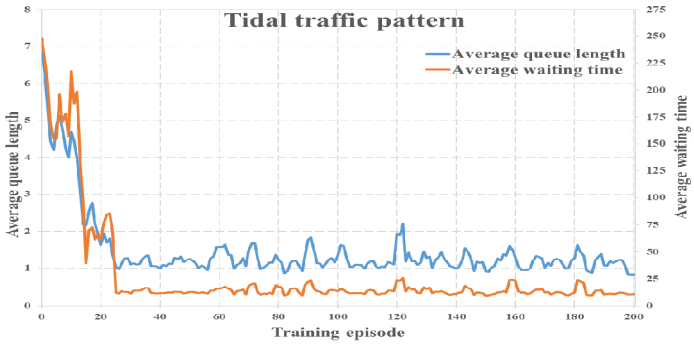


Figure 24: Through / Left-Turn Lane Traffic Pattern,

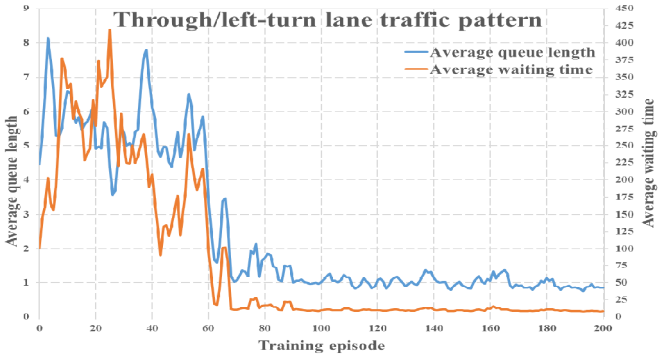


Figure 25: Tidal Traffic Pattern.

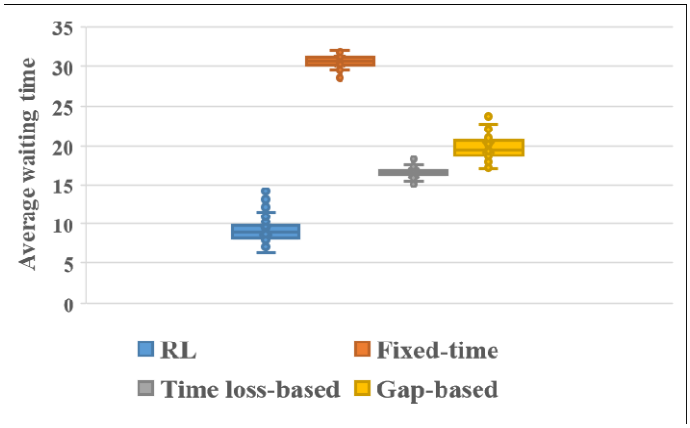


Figure 26: P1: Major / Minor Road Traffic Pattern.

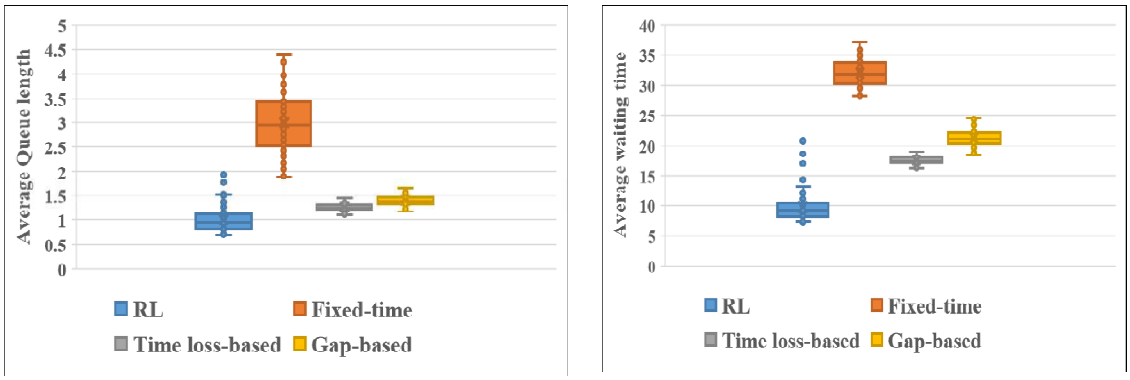


Figure 27: P2: Through / Left-Turn Lane Traffic Pattern.

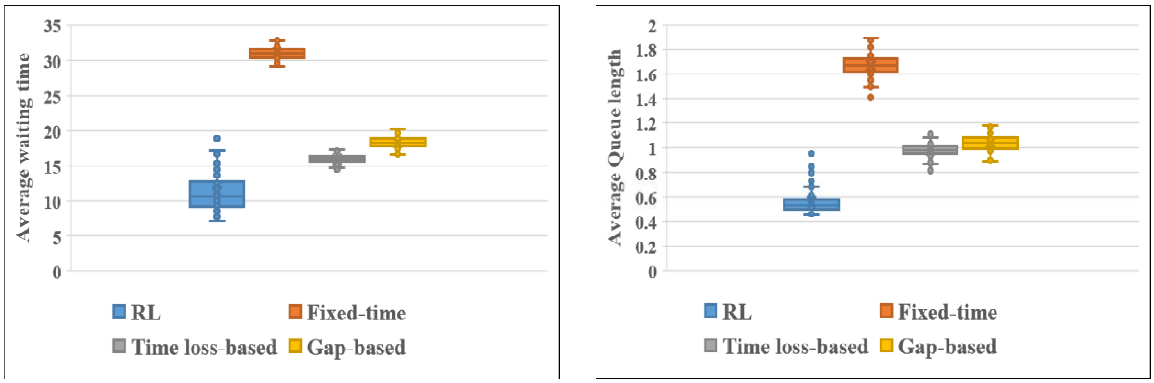


Figure 28: P3: Tidal Traffic Pattern.

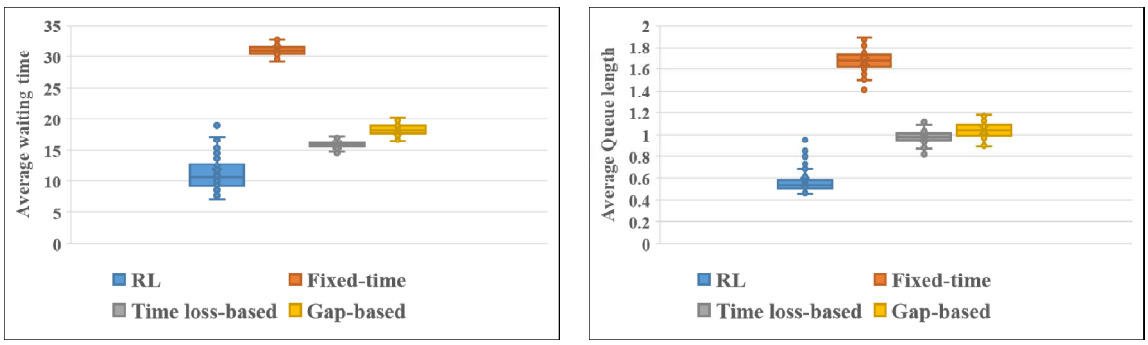


Figure 29: P4: Varying Demand Traffic Pattern.

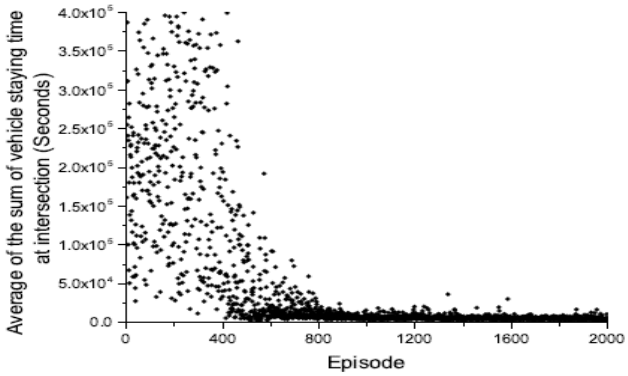
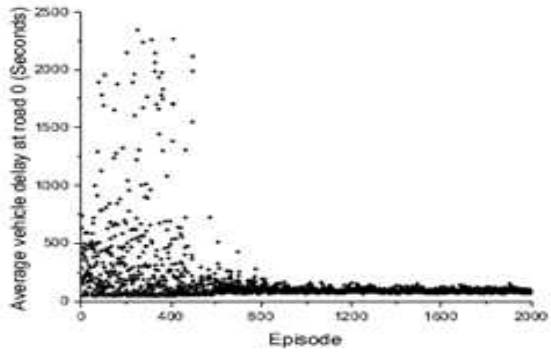
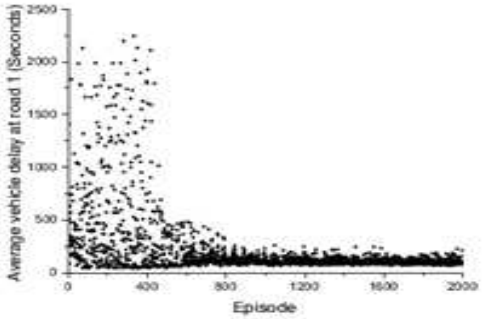


Figure 30: Average of Vehicles at the Intersection.



DSFSF

Figure 31



ROAD 1

Figure 32

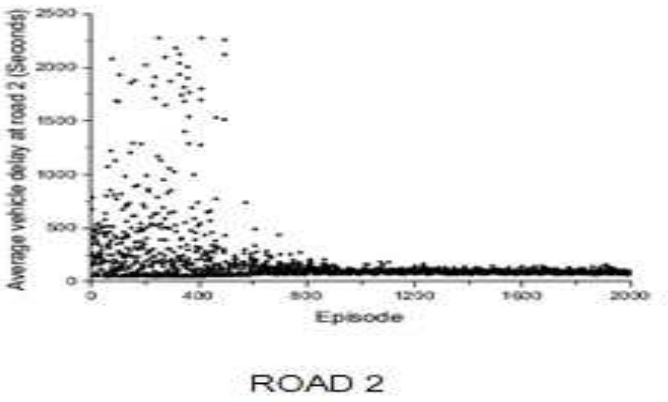


Figure 33

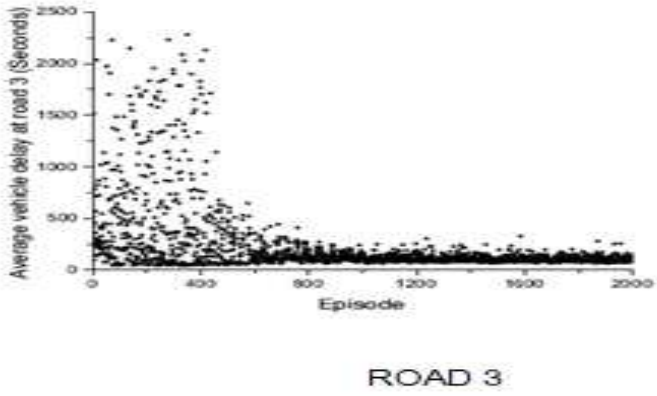


Figure 34: Average Time A Vehicle Remains Delayed.

10. CONCLUSIONS & FUTURE SCOPES

This project uses neural networks and reinforced learning to create an intelligent traffic signal controller [7] that takes into the consideration of the spatial-temporal characteristics of urban traffic flows. a Q-learning network is proposed to extract the information from the state space in order to derive the optimal signal control policy and to perform with large state space which consists of real-time vehicle position and speed [5].

Four traffic patterns are used in SUMO simulation. The simulation demonstrates the performance of our reinforced learning approach under different traffic patterns against the benchmark methods in terms of queue length and waits time.

There is scope for an improvement of the Q-learning network by changing the performance in convergence and stability. Advanced techniques such as dulling network and double Q-learning network can be employed. By extending the reinforced learning approach to more complex urban intersection settings such as an arterial or a multi-intersection network presents interesting challenges for exploration of the proposed methodology.

Future works are aimed at further improving achieved results in traffic signal control within a longer term, at investigating what would be the implications of introducing multiple reinforced learning agents within a road network and what would be the possibility to coordinate their efforts for achieving global improvements over local ones, and also the implications on the vehicle population, that could perceive the change in the infrastructure and adapt in turn to utilize additional opportunities and potentially negating the achieved improvements due to an additional traffic demand on the

improved intersections.

It is important to perform analyses along this line of work to understand the plausibility, potential advantages or even unintended negative implications of the introduction in the real world of this form of self-adaptive system.

The time constraints have prevented us from being able to analyze our approach when multiple intersections are present. It would be interesting to see if the same state-action pairs would be learned or if the presence of multiple nodes would cause these to change.

We generated simulations based on Q-learning and reinforced learning to test a four-intersection model however we have not had time to analyze this and learning Q values for multiple intersections would increase simulation time which itself already takes an hour to run.

Permitted more time, then we could expand this simulation scope and possibly consider implementing other state values apart from the vehicle position and velocity matrices. Allowing an intersection to see the states of its neibhours. so this increases in state space that could prove benefit for improving traffic flow, but can also greatly increase learning time.

REFERENCES

1. *Abdulhai, B., Pringle, R., & Karakoulas, G. J. (2003). Reinforcement learning for true adaptive traffic signal control. Journal of Transportation Engineering, 129(3), 278285 (cit. on pp. 24).*
2. *Araghi, S., Khosravi, A., Johnstone, M., & Creighton, D. (2013, October). Q-learning method for controlling traffic signal phase time in a single intersection. In 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013) (pp. 12611265) (cit. on pp. 36).*
3. *Balaji, P. G., German, X., & Srinivasan, D. (2010). Urban traffic signal control using reinforcement learning agents. IET Intelligent Transport Systems, 4(3), 177188 (cit. on pp. 31).*
4. *Chin, Y. K., Kow, W. Y., Khong, W. L., Tan, M. K., & Teo, K. T. K. (2012, November). Q-learning traffic signal optimization within multiple intersections traffic network. In 2012 Sixth UK Sim/AMSS European Symposium on Computer Modelling and Simulation (cit. on pp. 17).*
5. *Danthala, S. W. E. T. H. A., et al. "Robotic Manipulator Control by using Machine Learning Algorithms: A Review." International Journal of Mechanical and Production Engineering Research and Development 8.5 (2018): 305310.*
6. *Deepa, S., and R. Umarani. "Steganalysis on images based on the classification of image feature sets using SVM classifier." International Journal of Computer Science and Engineering (IJCSE) 5.5 (2016): 15-24.*
7. *Duwaer, D. A. (2016). On deep reinforcement learning for data-driven traffic control. LD Software, Eindhoven (cit. on pp. 34, 37).*
8. *Durgabai, R. P. L., and P. Bhargavi. "Pest Management using Machine Learning Algorithms: A Review." International Journal of Computer Science Engineering and Information Technology Research (IJCSEITR) 8.1 (2018): 1322.*

9. Gao, J., Shen, Y., Liu, J., Ito, M., &Shiratori, N. (2017). Adaptive traffic signal control: Deep reinforcement learning algorithm with experience replay and target network. arXiv preprint arXiv:1705.02755(cit. on pp. 35, 36).

10. Genders, W., &Razavi, S. (2018). Evaluating reinforcement learning state representations for adaptive traffic signal control. Procedia computer science, 130, 2633 (cit. on pp. 37).

11. Genders, W., &Razavi, S. (2016). Using a deep reinforcement learning agent for traffic signal control. arXiv preprint arXiv:1611.01142 (cit. on pp. 23).

12. Guo, M., Wang, P., Chan, C. Y., &Askary, S. (2019, October). A Reinforcement Learning Approach for Intelligent Traffic Signal Control at Urban Intersections. In 2019 IEEE Intelligent Transportation Systems Conference (ITSC) (pp. 42424247) (cit. on pp. 23).

13. Medina, J. C., &Benekohal, R. F. (2012). Agent-based traffic management and reinforcement learning in congested intersection network (No. 072IY03). NEXTRANS Center (US) (cit. on pp. 8).

14. Medina, J. C., &Benekohal, R. F. (2015, September). Vehicle detection design (coverage and accuracy) and the performance of congested traffic networks. In 2015 IEEE 18th International Conference on Intelligent Transportation Systems (pp. 179184). IEEE. doi:10.1109/itsc.2015.38 (cit. on pp. 13).

15. PATEL, AJAY M., A. PATEL, and HIRAL R. PATEL. "COMPARATIVE ANALYSIS FOR MACHINE LEARNING TECHNIQUES APPLIANCE ON ANOMALY BASED INTRUSION DETECTION SYSTEM FOR WLAN." (2013).International Journal of Computer Networking, Wireless and Mobile Communications (IJCNWMC) 3. 4, Oct 2013, 7786.

16. Wan, C. H., & Hwang, M. C. (2019). Adaptive Traffic Signal Control Methods Based on Deep Reinforcement Learning. In Intelligent Transport Systems for Everyone's Mobility (pp. 195-209). Springer, Singapore. doi:https://doi.org/10.1007/978-981-13-7434-0_11 (cit. on pp. 2).

17. Vidali, A., Crociani, L., Vizzari, G., & Bandini, S. A Deep Reinforcement Learning Approach to Adaptive Traffic Lights Management (cit. on pp. 21, 28).