A Project Report

On

Intelligent Traffic control system

Submitted in partial fulfilment of the requirement of

University of Mumbai

For the Degree of

Bachelor of Engineering

in

COMPUTER ENGINEERING

Submitted by

Kranti Shingate
Komal Jagdale
Yohann Dias

Supervised by

Ms Dakshayni

Department of Computer Engineering

Fr. Conceicao Rodrigues Institute of Technology
Sector 9A, Vashi, Navi Mumbai - 400703

UNIVERSITY OF MUMBAI

2019-2020

APPROVAL SHEET

This is to certify that the project entitled

"Intelligent Traffic control system"

Submitted by

Kranti Shingate	**101652**
Komal Jagdale	**101669**
Yohann Dias	**101667**

Supervisors : ______________________

Project Coordinator : ______________________

Examiners : 1. ______________________

2. ______________________

Head of Department : ______________________

Date :

Place :

Declaration

We declare that this written submission for B.E. Declaration entitled " **Intelligent Traffic control system**" represent our ideas in our own words and where others' ideas or words have been included. We have adequately cited and referenced the original sources. We also declared that we have adhere to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any ideas / data / fact / source in our submission. We understand that any violation of the above will cause for disciplinary action by institute and also evoke penal action from the sources which have thus not been properly cited or from whom paper permission have not been taken when needed.

Project Group Members:

1. Kranti Shingate, 101652

2. Komal Jagdale, 101669

3. Yohann Dias, 101667

Abstract

Now-a-days, due to increased automobile production we are facing numerous traffic problems. People are unable to reach their destination on time due to huge traffic.The system used for directing traffic is not dependent on real time scenario of a intersection.Traffic Light Control System with preset timers are widely used to invigilate and control the flow of automobiles through the junctions of many roads.They aim to realize smooth motion of vehicles on roads.However,the synchronization of multiple traffic light systems at adjacent intersections is a complicated problem given the various parameters involved.To handle such traffic we need to either expand road networks or we need some adaptive traffic control system which handle such traffic intelligently.So here,we propose a system which handle traffic intelligently which adapts according to the density of traffic by automatically increasing or decreasing traffic signal time by using Experience Replay mechanism.In this system,we propose deep Reinforcement Learning algorithm for extracting features to make a decision. Keywords: Reinforcement Learning (RL), Traf ic Light Control System (TLCS),Experience Replay mechanism

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The First Gas-lit Traffic Lights The purpose of a traffic signal is to regulate the flow of automobiles, traffic signals came into existence long before automobiles were invented. The idea for developing traffic signals began in the 1800's[1], and on December 10, 1868, the first gas-lit traffic lights were installed outside the Houses of Parliament in London.This model was proposed by a British railway engineer, J.P Knight. It was implemented to control the traffic of horse carriages in the area, and to allow pedestrians to safely cross the roads.But this gas-fueled lights needed to be manually controlled by a police officer.Due to this gas-lit lights,there were some incidents of lights exploding at night and injuring police officers who were controlling them.
The First Electric Traffic Lights With the invention of automobiles, the traffic on the roads has increased significantly, so there was a need for a better traffic system.In 1912, an American policeman, Lester Wire, who was concerned with the increasing traffic, came up with the idea of the first electric traffic light. Based on Wire's design, the lights were first installed in Cleveland,Ohio, on August 5, 1914, at the corner of 105th and Euclid Avenue.The first electric traffic light had only red and green lights; it did not have a yellow light like modern-day traffic signals. Instead of a yellow light, it had a buzzer sound that was used to indicate that the signal would be changing soon.
The First Four-way and Three-colour Traffic Lights In the year 1920, a policeman named William Potts in Detroit, Michigan invented the first four-way and three-colored traffic lights. Apart from red and green, a third color – amber (or yellow) – was introduced. Detroit became the first city to implement the four-way and three-colored traffic lights. In the 1920's, several automated traffic signals were installed in major cities around the world. The modern traffic light still uses this famous T-shaped model with three different colors.
The Computerization of Traffic Lights In the 1960's, with the invention of computers, traffic lights started to become computerized. Over time, computers improved, and the traffic lights subsequently improved, and they could now monitor traffic and change the lights accordingly.Based on the software, the traffic of a city could now be predicted and accordingly controlled.At present, traffic all over the world can be monitored, which gives an idea about the traffic at a certain time, which city has the most traffic, and what the peak hours of traffic are, so the lights can be controlled accordingly.

1.2 Motivation

Traffic overcrowding problem is continuously growing all over the world and it has become a burden for commuters.One more issue in traffic jam is delay of red light.Traffic congestion can also be promoted by large red light delay.This delay problem is provoked because lights in the traffic control are codified and it is not dependent on real traffic.Therefore solution to this problem is employing reinforcement learning (RL) – a machine learning framework which attempts to approximate an optimal decision-making policy.The system provides solution to reduce traffic in metropolitan cities by taking into consideration real time traffic scenario and reinforcement learning algorithm to improve over time.

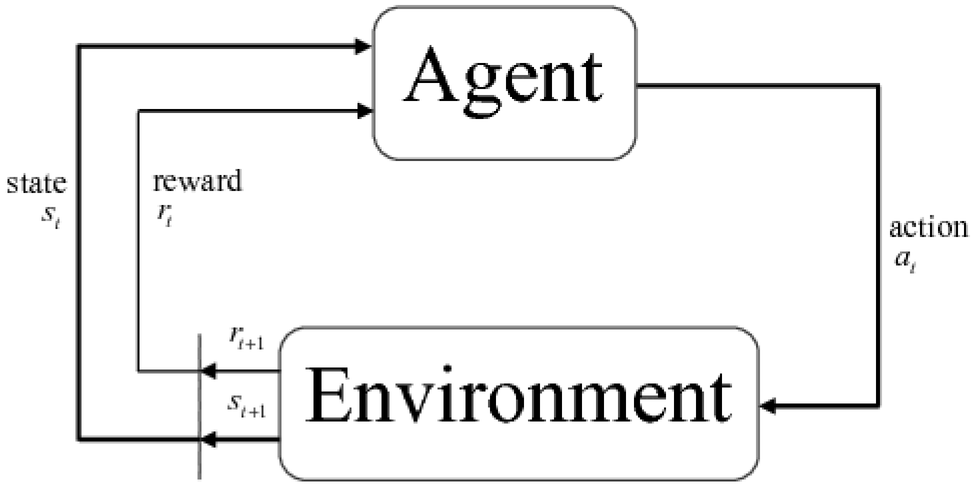


Figure 1.1: Reinforcement learning

1.3 Aim and Objective

Aim is to develop a traffic control system which handles and direct the traffic intelligently using reinforcement learning algorithm and objective to achieve a smooth transportation of vehicles and to reduce environmental issues like raised air pollution,wastage of fuel and risk of accident.

1.4 Report Outline

The report is structured as follows: Chapter 2 gives an idea of the systems existing for traffic control and a brief information on the papers referred. Chapter 4 presents design of system using reinforcement learning. Chapter 5 gives the results obtained. In Chapter 6 we conclude our report and present future scope of the system.

Chapter 2

Study of Traffic Control System

2.1 About the Technique

Traditional Traffic Control system

i. Traffic signals with preset timers

Under fixed time operation the traffic signals will display green to each approach for the
same time every cycle regardless of the traffic conditions. This may be best suited for heavily congested areas but for low traffic density the sequence is not so beneficial as there are no vehicles waiting.

ii. Adaptive Traffic Signal Control System

The Adaptive Traffic Signal Control System signalized intersections in the Bhubaneshwar city gets input from sensors embedded in road and synchronizes the group of traffic signals accordingly.This signaling system is run on solar power.The system is infeasible and costly since it requires system embedded in roads .

iii. LQF(longest queue first) scheduling

The signal-scheduling algorithm minimizes the queue sizes at each approach to the intersection.The goal is to lower vehicle delay as compared to a current state signal control method. A focus is given by giving preference vehicles (such as emergency vehicles or large trucks). As the system concentrates on reducing the queue length, stability of the system is a major concern.

2.2 Various Available Technique

Traffic signal control is a task that is well suited for RL techniques. In this context, one or more autonomous agents have the goal of maximizing the efficiency of traffic flow that drives through one or more intersection controlled by traffic lights. In this section, the most widely used approaches to design RL components (state, action, reward) in the context of traffic signal control are described.

The use of RL for traffic signal control is motivated by several reasons. First, if trained properly RL agents can adapt to different situations such as road accidents or bad weather conditions. Second, RL agents can self-learn without supervision prior knowledge of the environment. Third, a model of the environment that describes every variable of the environment is not needed since the agent learns using the system performance metric i.e. the reward.

RL techniques applied to traffic signal control address the following challenges:

- **Inappropriate traffic light sequence :** Traffic lights usually choose the phases in a static, predefined policy. This method could cause the activation of an inappropriate traffic light phase in a situation that could cause an increase in travel times.
- **Inappropriate traffic light durations :** Every traffic light phase has a predefined duration which does not depend on the current traffic conditions. This behavior could cause unnecessary waitings for the green phase.

2.3 Related Works

Traffic control system using reinforcement learning
Intelligent Traffic Signal Control:Using Reinforcement Learning with Partial Detection [2]RL algorithm used for partially observable. ITS is based on DSRC i.e Dedicated short range communication.The system is good for those vehicles which have wireless communication system so that only those vehicles are detected and system makes decision accordingly.However, system gives better performance for the detected vehicles than undetected vehicles.

Reinforcement learning-based multi-agent system
The multi-agent system[3] for network traffic signal control introduces the use of a multi-agent system and reinforcement learning algorithm to obtain an efficient traffic signal control.In this two types of agents are used i.e central agent and an outbound agent.The outbound agents schedule traffic signals (LQF algorithm) and central agent learns a value function(Q-learning) driven by its local and neighbour's traffic conditions.At low arrival rates, LQF scheduling algorithm performs slightly better than the multi-agent Q Learning system.Adaptive traffic signal control, which adjusts traffic signal timing according to real-time traffic, has been shown to be an effective method to reduce traffic congestion.

Deep Reinforcement Learning Algorithm with Experience Replay and Target Network

The system works on adaptive traffic signal control make responsive traffic signal control[4] decisions based on human-crafted features (e.g. vehicle queue length). However, human-crafted features are abstractions of raw traffic data (e.g., position and speed of vehicles), which ignore some useful traffic information and lead to sub optimal traffic signal controls. In this paper, they propose a deep reinforcement learning algorithm that automatically extracts all useful features (machine-crafted features) from raw real-time traffic data and learns the optimal policy for adaptive traffic signal control. To improve algorithm stability, experience replay and target network mechanisms are adopted.

Reinforcement Learning for Intelligent Traffic Light Control

It mainly emphasizes on determining the feasibility and value of applying a model-less[5] temporal difference reinforcement learning algorithm to traffic light control.The main drawback of the implementation is environment involves four-way intersections but allows traffic to flow in either horizontal or vertical not both.

Multi-Agent Reinforcement Learning for Intelligent Traffic Light ControlA set of multi-agent model-based Reinforcement Learning system[6] formulated under Markov Decision process model for traffic light control. The system does not rely on heuristics equations but learns the optimal control by improving its experience on interacting with the environment.Future scope includes adding public transport which should get priorities for crossing roads, since they carry more passengers.

Table 2.1: Comparative study of traffic control system using reinforcement learning

No.	Title	Methodology	Limitations
1.	Intelligent Traffic Signal Control: Using Reinforcement Learning with Partial Detection	RL algorithm for partially observable ITS based on DSRC.(Dedicated Short Range Communications).	Better performance for the detected vehicles than undetected vehicles.
2.	Reinforcement learning-based multi-agent system for network traffic signal control	The outbound agents schedule traffic signals (LQF algorithm) and the central agent learns a value function(Q-learning) driven by its local and neighbours traffic conditions.	At low arrival rates, LQF scheduling algorithm performs slightly better than the multi-agent Q Learning system. Require lane system.
3.	Adaptive Traffic Signal Control:Deep reinforcement learning algorithm with experience replay and target network	Reinforcement learning algorithm using experience replay and target network	Gives accurate result for machine -crafted features only
4.	Reinforcement Learning for Intelligent Traffic Light Control	To determine the feasibility and value of applying a model-less temporal difference learning algorithm to traffic light control.	Environment involves four-way intersections but allows traffic to flow in either horizontal or vertical not both.
5.	Multi-Agent Reinforcement Learning for Intelligent Traffic Light Control	A set of multi-agent model-based Reinforcement Learning system for traffic light control	Should add public transport which should get priorities for crossing roads,since they carry more passengers

Chapter 3

Proposed System

3.1 Problem Statement

The chance of improvement in traffic flow that drives through an intersection controlled by traffic lights will be investigated using artificial intelligence techniques. The analysis will be conducted with a simulation where an agent manages the choice of which traffic light's phase activate with the objective of optimizing the traffic efficiency. In order to choose the best light phase in every situation, some learning mechanism is required by the agent.

The learning techniques used in this project are related to reinforcement learning and deep learning. The entire system that comprehends the agent, its elements, and the learning techniques in this thesis is called **Traffic Light Control System (TLCS)** and is described in this chapter.

In order to design a system based on the reinforcement learning framework, it is necessary to define the environment, a state representation, an action space, a reward function and the agent learning techniques involved.

In Table 3.1 are listed the terms used in this chapter

Table 3.1: Notations

Notation	Meaning
TLCS	Traffic Light Control System.
A	Set of actions.
a	A single action.
s	State.
r	Reward.
TL	Set of traffic lights.
IDR	Intersection Discretized Representation.
NSA	North-South Advance.
NSWA	North-South Left Advance.
EWA	East-West Advance.
EWLA	East-West Left Advance.
twt	Total waiting time.
wt	Waiting time.
veh	Vehicle.
t	Timestep, the precise moment when the agent interact with the simulation.
h	An episode.
H	Total number of episodes.
B	Batch.
b	a sample contained in the batch.

The goal of this project is to develop a system for controlling traffic generated in a 4-way intersection with four incoming lanes and four outgoing lanes per arm.
Car follows the possible directions defined by the incoming lanes:
i. left-most lane (left-turn only)
ii. right-most lane (right-turn and straight)
iii. two middle lanes (only for going straight)

In this project the environment is represented by a 4-way intersection, where 4 lanes per arm approach the intersection from the compass directions and lead to 4 lanes per arm leaving the intersection. A set of traffic lights T L manages the incoming traffic into the intersection. The TLCS is composed by a single agent that interacts with the environment using a state s, an action a and a reward r. A deep Q-learning neural network is the learning mechanism of the agent.
Figure 3.1 shows a summary of the TLCS.

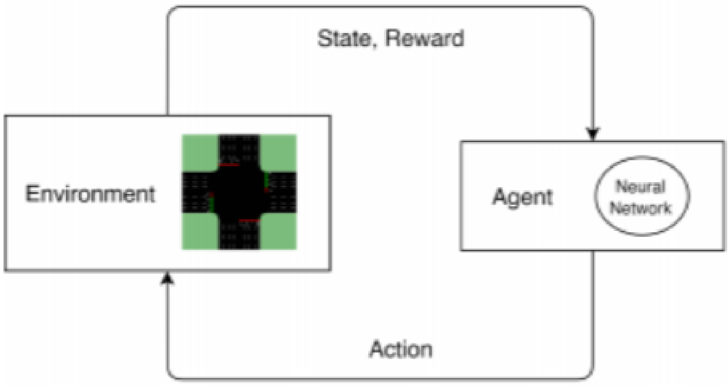


Figure 3.1: Generic working process of the TLCS

The problem is defined as follows: given the state of the intersection s, what is the traffic light phase a that the agent should choose, selected from a fixed set of predefined actions A, in order to maximize the reward r and optimize the traffic efficiency of the intersection.

3.2 Scope

The system provides solution to reduce traffic in metropolitan cities by taking into consideration real time traffic scenario and reinforcement learning algorithm to improve over time.The system is bound to take into consideration only a single 4-way intersection and not a multiagent system considering more than one intersection. The algorithm used learns over a period of time so the initial stages of system might not give optimal results for the detected traffic.Real time system may show variations than the simulation as traffic detection for every fraction of a second is crucial in taking decision.

To efficiently reduce the traffic, we impose the following:

i. Four lanes in each of the four arms of intersection.

ii. only single intersection is considered.

3.3 Proposed System

3.3.1 The environment of the simulation

The simulated environment of this project is the intersection represented in Figure 3.2. It consists of a four-way intersection (Figure 3.2) where there are 4 lanes approaching the intersection from the compass directions connected to 4 lanes leaving the intersection. Each arm of the junction is 750 meters long from the vehicle origin to the intersection's stop line. On every arm, the four lanes used for entering the junction indicate the possible directions that a car can take. When a vehicle approaches the junction, it selects the desired lane based on its destination:

- Turn left: select only the left-most lane.
- Go straight ahead: select the two central lanes or the right-most lane.
- Turn right: select only the right-most lane.

The traffic light system

Traffic lights in the environment are indicated by a color on the stop line of every entrance lane, which represents the status of the traffic light for that lane on a precise timestep. For example, Figure 3.2 shows a green light for vehicles coming from the north or south direction that want to go straight or turn right, and red for everyone else.

Every traffic light in the environment works accordingly to the following rules:

1. The color phase transition is always the following: red-green-yellow-red.
2. The duration of every traffic light phase is fixed. The green time is always 10 seconds and the yellow time is always 4 seconds. Consequently, the duration of the red phase is defined as the amount of time since the last phase change.
3. For every timestep, at least one traffic light is in yellow phase or green phase.
4. It is not possible to have every traffic light in the red phase simultaneously.

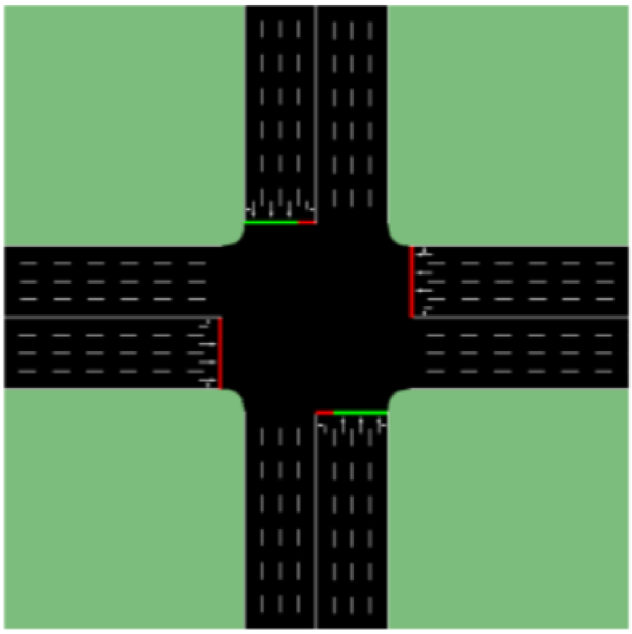


Figure 3.2: A zoomed view of the intersection without vehicles

3.3.2 The state representation

The **state of the agent** describes a representation of the situation of the environment in a given timestep t and it is denoted with st. To allow the agent to effectively learn to optimize the traffic, the state should provide sufficient information about the distribution of cars on each road.

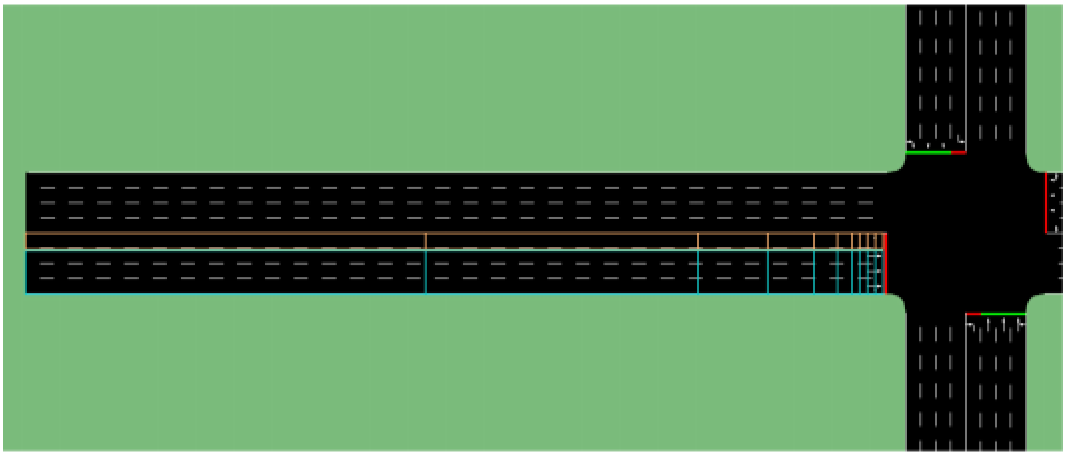


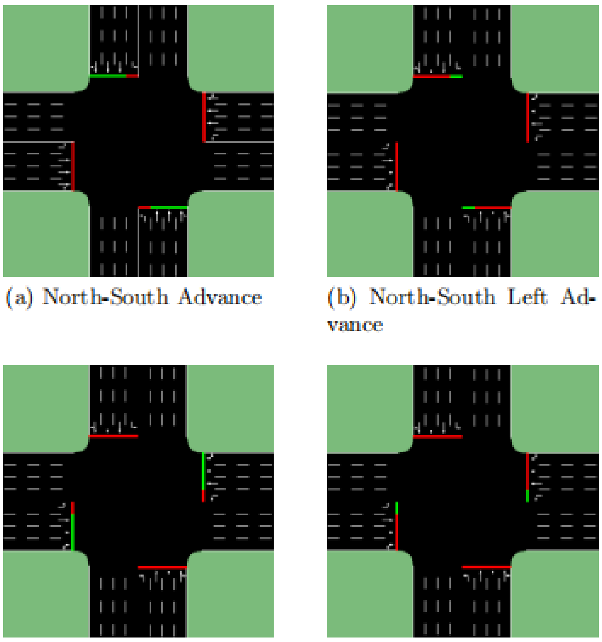
Figure 3.3: The state representation in the west arm of the intersection

In each arm of the intersection, incoming lanes are discretized in cells that can identify the presence or absence of a vehicle inside them. Figure 3.3 shows the state representation for the west arm of the intersection.
It must be noted that a particular cell does not necessarily describe the situation in a single lane. As seen in Figure 3.4, in fact, the 3 lanes dedicated to going straight and turning right share the same cells since they share the same traffic light, while the lane dedicated to turning left has a separate set of cells.
Along the length of a lane there are 10 cells, which means that in every arm of the intersection there are 20 cells and in the whole intersection 80 cells.

3.3.3 The action set

The action set identifies the possible actions that the agent can take.Figure 3.4 shows the 4 actions. The agent is the traffic light system, so doing an action translates to turning green some traffic lights for a set of lanes and keep it green for a fixed amount of time. The green time is set at 10 seconds and the yellow time is set at 4 seconds. In other words, the task of the agent is to initiate a green phase choosing from the predefined ones. The action space is defined in the set. The set represents every possible action that the agent can take. Every action a of set is described below :

- **North-South Advance (NSA):** the green phase is active for vehicles that are in the north and south arm and wants to proceed straight or turn right.
- **North-South Left Advance (NSLA):** the green phase is active for vehicles that are in the north and south arm and wants to turn left.
- **East-West Advance (EWA):** the green phase is active for vehicles that are in the east and west arm and wants to proceed straight or turn right.
- **East-West Left Advance (EWLA):** the green phase is active for vehicles that are in the east and west arm and wants to turn left



(a) North-South Advance

(b) North-South Left Advance

(c) East-West Advance

(d) East-West Left Advance

Figure 3.4: Action set

3.3.4 The reward function

In reinforcement learning, the reward represents the feedback from the environment after the agent has chosen an action. The agent uses the reward to understand the result of the taken action and improve the model for future action choices. Therefore, the reward is a crucial aspect of the learning process. The reward usually has two possible values: positive or negative.A positive reward is generated as a consequence of good actions, a negative reward is generated from bad actions.

In this application, the objective is to maximize the traffic flow through the intersection over time. In order to achieve this goal, the reward should be derived from some performance measure of traffic efficiency, so the agent is able to understand if the taken action reduce or increase the intersection efficiency. In traffic analysis, several measures are used, such as throughput, mean delay and travel time.

The candidate measures to generate the reward were the following :

- **Queue length:** the number of vehicles with a speed of less than 0.1 m/s.
- **Total waiting time:** the sum of individual waiting times of each car in the environment in timestamp t. Each waiting time is defined as the amount of time a vehicle has a speed of less than 0.1 m/s.
- **Throughput:** the number of vehicles that crosses the intersection over a defined period of time.

3.3.5 The agent's learning mechanism

The learning mechanism involved in this thesis is called Q-Learning.

Q-Learning

Q-Learning is a form of model-free reinforcement learning. It consists of assigning a value, called the Q-value, to an action taken from a precise state of the environment. Formally, in literature, a Q-value is defined as in equation.

$$Q(st, at) = Q(st, at) + (rt{+}1 + \ \cdot \max AQ(st{+}1, at)\ )\ Q(st, at))$$

where Q(st, at) is the value of the action at taken from state st. The equation consists on updating the current Q-value with a quantity discounted by the learning rate . Inside the parenthesis, the term rt+1 represents the reward associated to taking action at from state st. The subscript t+1 is used to emphasize the temporal relationship between taking the action at and receiving the consequent reward. The term Q(st+1, at) represents the immediate future's Q-value, where st+1 is next state in which the environment has evolved after taking action at in state st. The expression maxA means that, among the possible actions at in state st+1, the most valuable is selected. The term is the discount factor that assumes a value between 0 and 1, lowering the importance of future reward compared to the immediate reward.

In this system, a slightly different version of the equation (3.9) is used and it is presented in equation

$$Q(st, at) = rt{+}1 + \ \cdot \max AQ'(st{+}1, at{+}1)$$

Where the reward rt+1 is the reward received after taking action at in state st. The term Q0(st+1, at+1) is the Q-value associated with taking action at+1 in state st+1, i.e. the next state after taking action at in state st. As seen in equation (3.9), the discount factor denote a small penalization of the future reward compared to the immediate reward.

Chapter 4

Design Of the System

4.1 Requirement Engineering

4.1.1 Requirement Elicitation

Increment in urbanization has prompted many traffic congestion issues. The primary necessity is to control traffic at the intersection based on ongoing traffic. Traffic police accomplish this by physically controlling the traffic. Be that as it may, this turns into a tedious task. So there is a need for traffic signal which modifies as per the traffic conditions at that time. The traffic signal goes about as a framework that alters the green signal time for the arm having more traffic thereby reducing the traffic at the intersection.

4.1.2 Software lifecycle model

Agile software model is used for developing software applications where project implementation is done iteratively or incrementally. This model helps to make changes or modification as per the user requirement for different traffic scenarios . The cycle stages are executed in parallel. The system explained in the report has been developed based on the agile framework model.

4.1.3 Requirement Analysis

The state is the traffic signal's perception of the environment in an arbitrary time step. This state space representation is obtained using SUMO (Simulation of urban mobility). The simulation generates 1000 cars for each episode. The cars arrival timing are defined according to a Weibull distribution [7]. 75 percent of vehicles spawned will go straight, 25 percent will turn left or right. Every vehicle have the same probability to be spawned at the beginning of every arm. On every episode the cars are generated randomly so is not possible to have two equivalent episode in term of vehicle's arrival layout.

4.1.3.1 Use Case Diagram

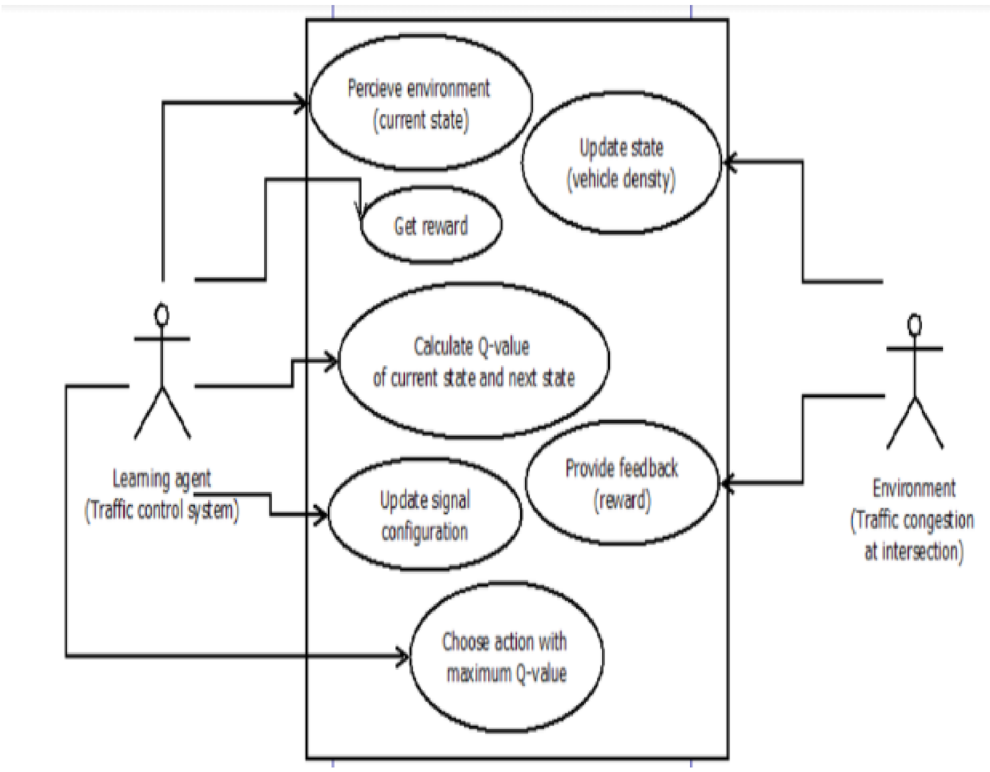


Figure 4.1: Use Case Diagram
The learning agent performs the tasks of perceiving the environment, getting reward, calculating q-value and updating the traffic signal using the state information and feedback provided by the environment.

4.1.3.2 Sequence Diagram

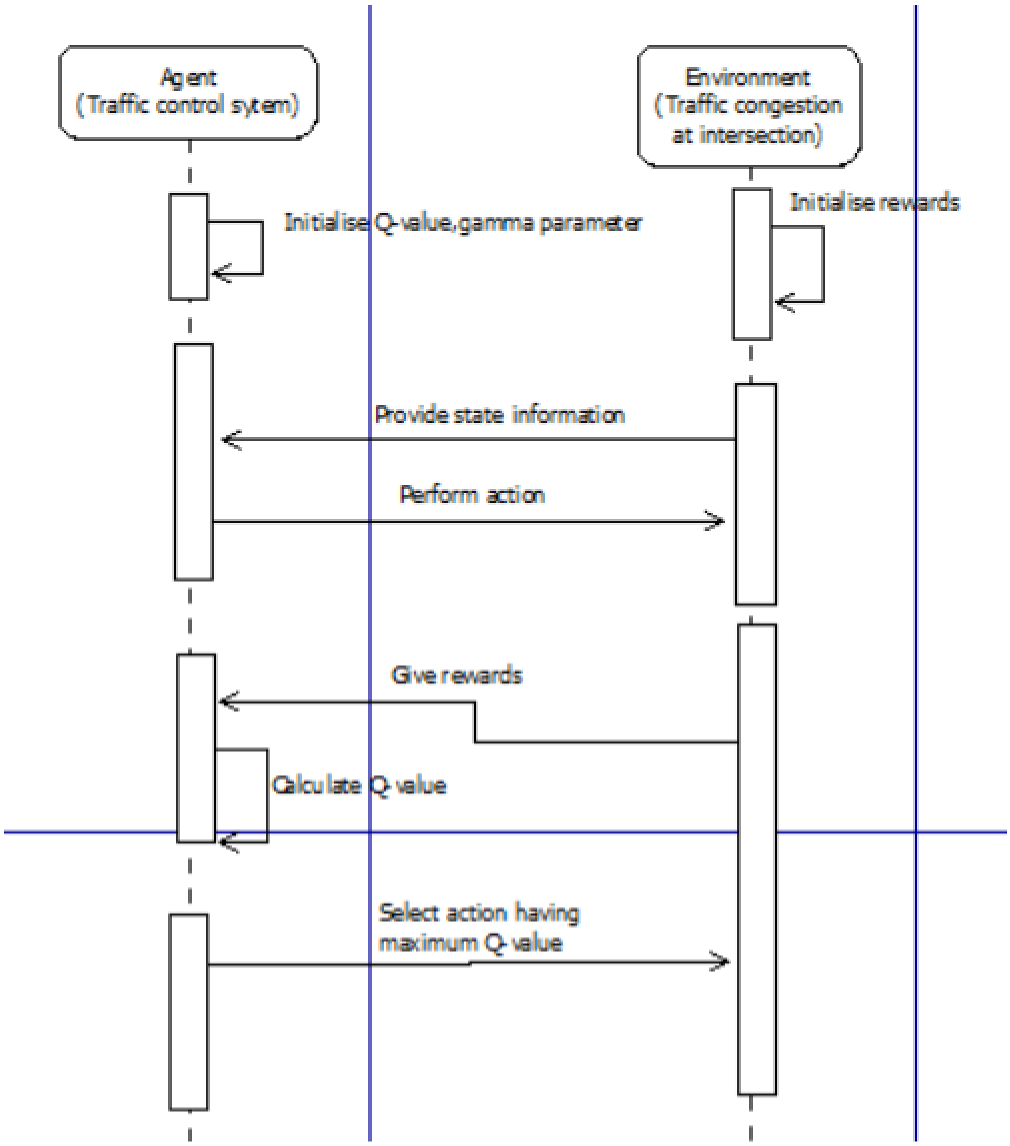


Figure 4.2: Sequence Diagram
Sequence diagram of various operations within the proposed system.

4.1.3.3 Cost Analysis

The current scope of the project only deals with the simulation of the traffic light system. So no cost segment is involved. The future scope might incorporate sensors for getting real time traffic, processor for running algorithm, traffic control system and other definite segments.

4.1.3.4 Hardware and software requirement

Software Requirements

1. Operating System: Windows 7, Windows 8 or Windows 10
2. Python 3.6
3. SUMO Traffic simulator 1.0.1
4. Tensor flow 1.11.0

Hardware Requirements

1. Processor (CPU) with 2 gigahertz (GHz) frequency or above
2. A minimum of 4 GB of RAM
3. Monitor Resolution 1024 X 768 or higher
4. A minimum of 20 GB of available space on the hard disk

4.2 System architecture

4.2.1 Block Diagram

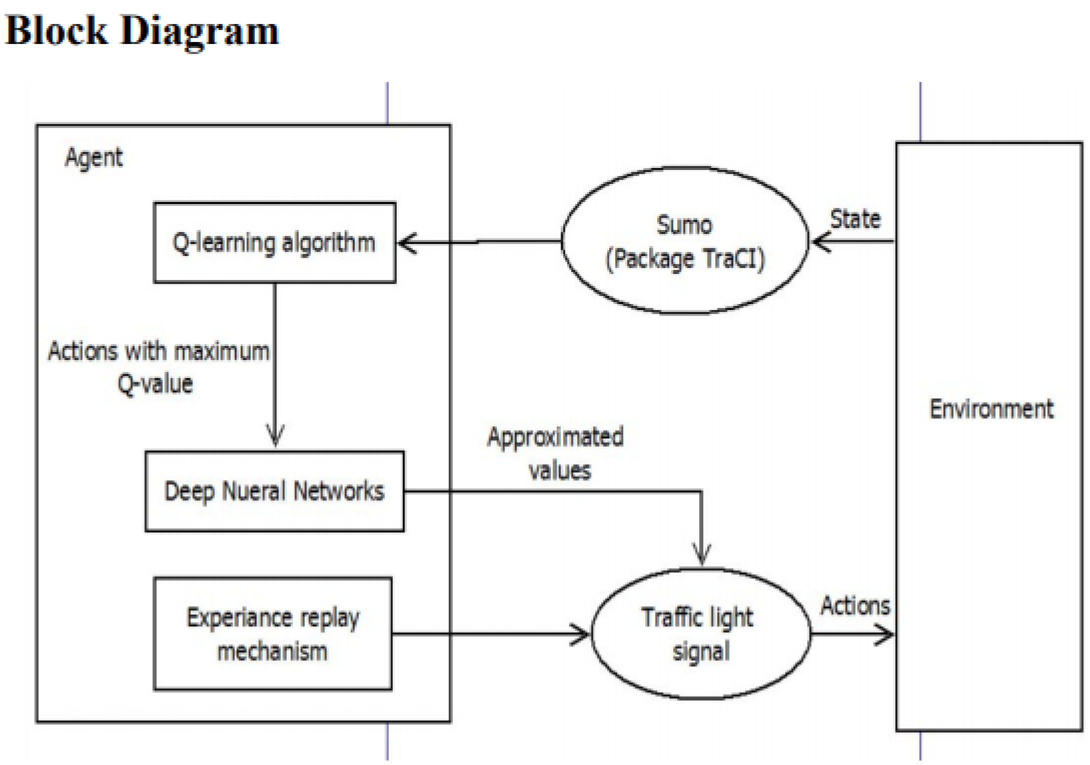


Figure 4.3: Block Diagram
Traffic light signal takes actions based on approximated values from deep neural networks and experience replay mechanism.

Chapter 5

Result and Discussion

5.1 Screenshots of the System

We have created an environment as a 4-way intersection (Figure 5.1) with four incoming lanes and four outgoing lanes per arm.
Car can follow the possible directions defined by each incoming lane -

- left-most lane dedicated to left-turn only
- right-most lane dedicated to right-turn and straight
- two middle lanes dedicated to only going straight.

Traffic light system is placed at the left-most lane and others three lanes share the same traffic light.

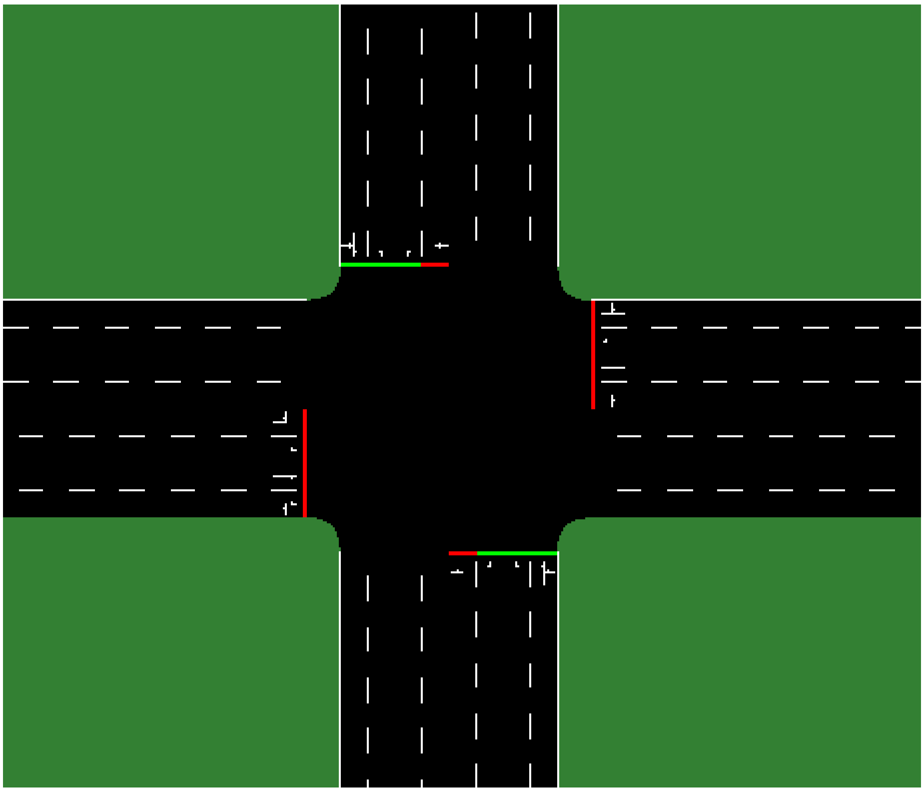


Figure 5.1: 4-way Intersection

For every episode, 1000 cars are generated by choosing random source and destination.Figure 5.2 shows the traffic generation.Out of the four lanes, two lanes are dedicated for straight and one for straight and right, so there are 75 percent chances that the cars will go straight (three out of four lanes) and a 25 percent probability of taking a left turn or right turn using the leftmost lane.The traffic signal has a duration of 10 seconds for each phase and before changing to green phase it has to change from red to amber for 4 seconds.

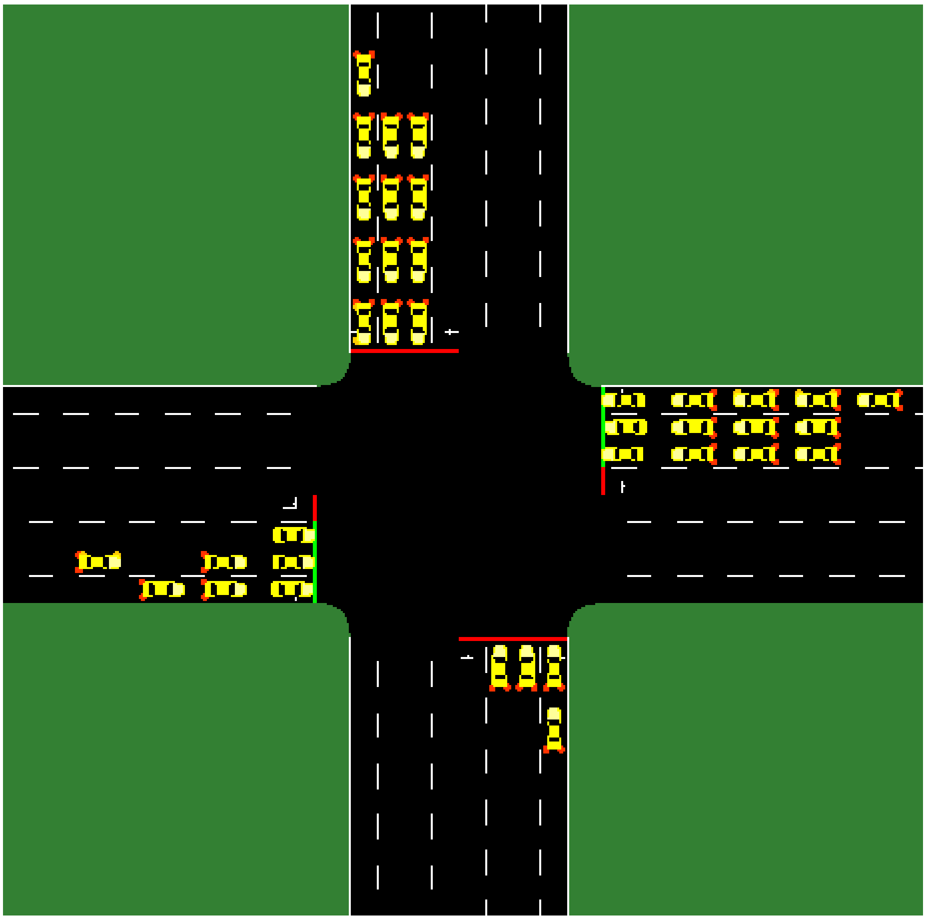


Figure 5.2: Traffic Generation

Action:
choice of the traffic light phase from a 4 possible predetermined phases, which are the described below. Every phase has a duration of 10 seconds. When the phase changes, a yellow phase of 4 seconds is activated.
For every state,reward and calculated Q-value the agent can take 4 actions:

North-South Advance:
Green for lanes in the north and south arm dedicated to turn right or go straight.(Figure 5.3)
North-South Left Advance:
green for lanes in the north and south arm dedicated to turn left. (Figure 5.4)
East-West Advance:
green for lanes in the east and west arm dedicated to turn right or go straight.
East-West Left Advance:
Green for lanes in the east and west arm dedicated to turn left.

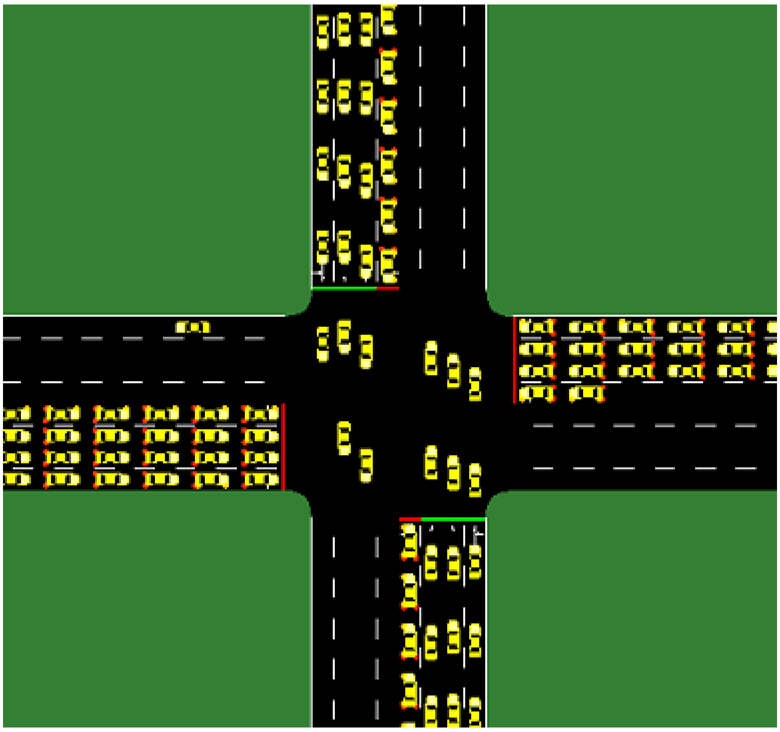


Figure 5.3: North-South Advance

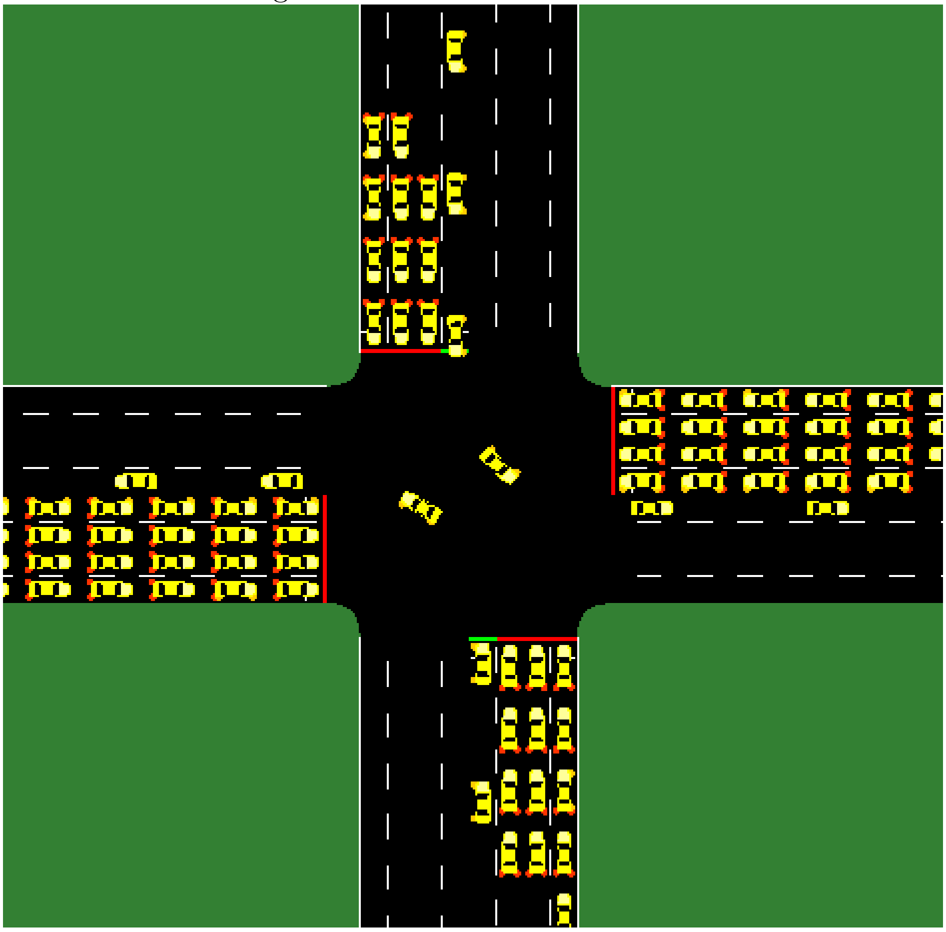


Figure 5.4: North-South Left Advance

Figure 5.5: East-West Advance

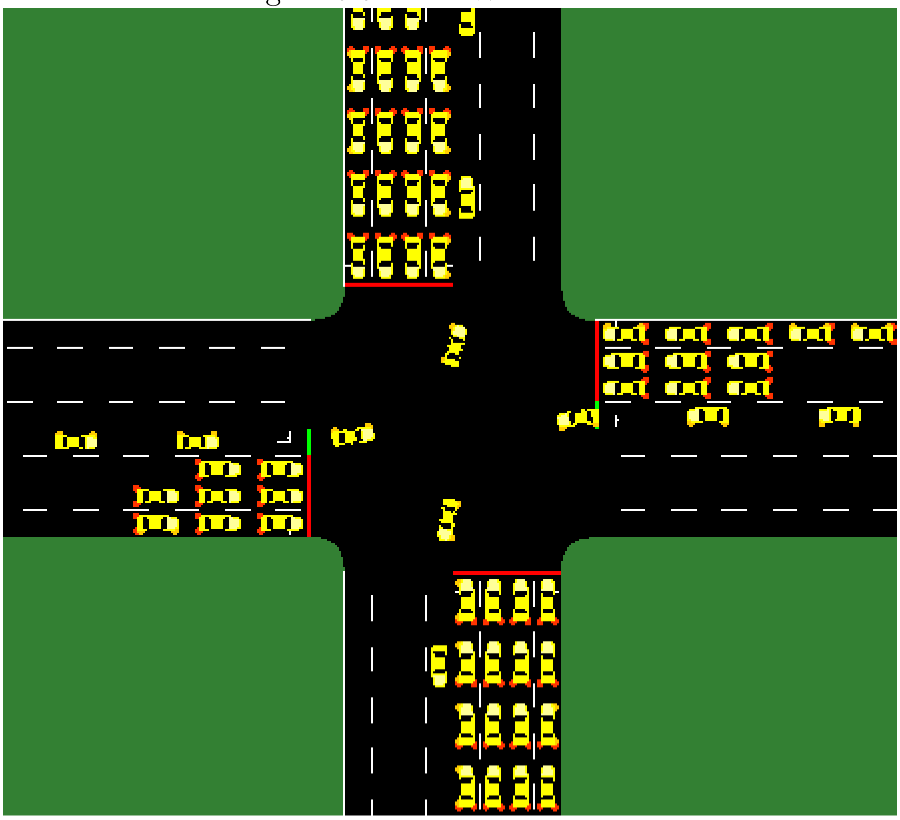
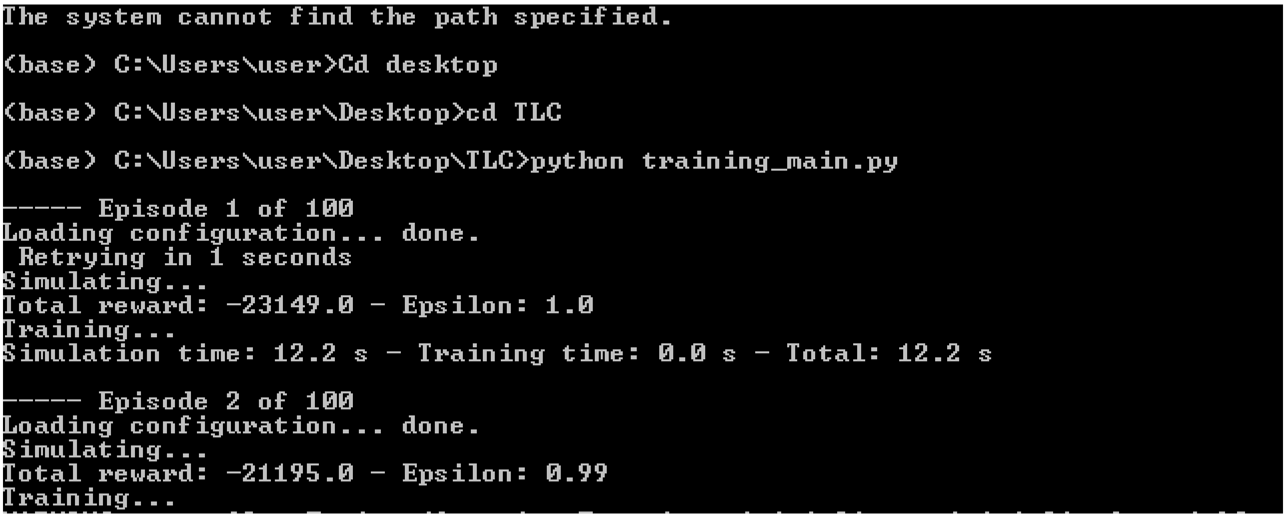
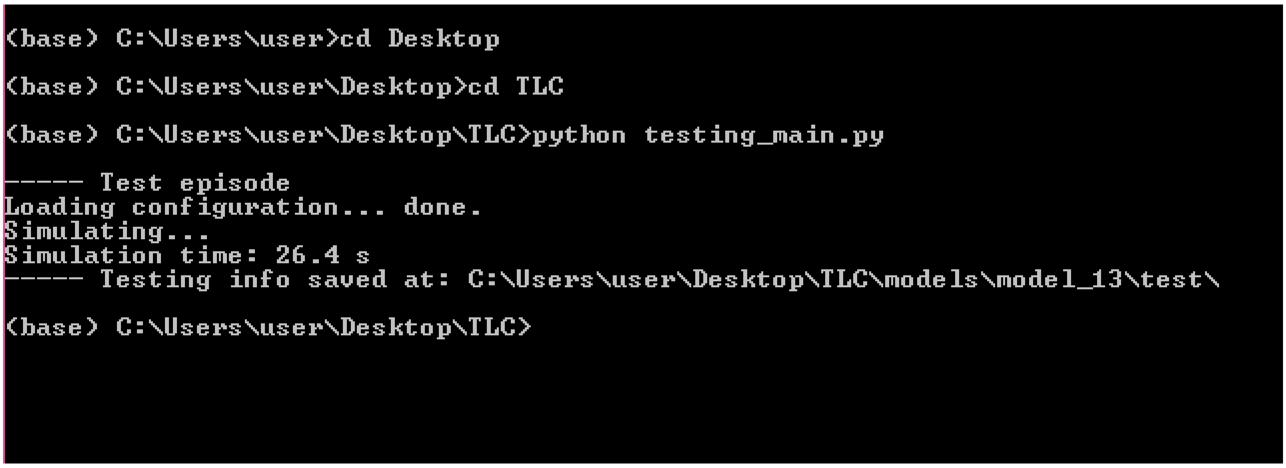


Figure 5.6: East-West Left Advance



```
The system cannot find the path specified.

(base) C:\Users\user>Cd desktop

(base) C:\Users\user\Desktop>cd TLC

(base) C:\Users\user\Desktop\TLC>python training_main.py

----- Episode 1 of 100
Loading configuration... done.
 Retrying in 1 seconds
Simulating...
Total reward: -23149.0 - Epsilon: 1.0
Training...
Simulation time: 12.2 s - Training time: 0.0 s - Total: 12.2 s

----- Episode 2 of 100
Loading configuration... done.
Simulating...
Total reward: -21195.0 - Epsilon: 0.99
Training...
```

Figure 5.7: Training Process



```
(base) C:\Users\user>cd Desktop

(base) C:\Users\user\Desktop>cd TLC

(base) C:\Users\user\Desktop\TLC>python testing_main.py

----- Test episode
Loading configuration... done.
Simulating...
Simulation time: 26.4 s
----- Testing info saved at: C:\Users\user\Desktop\TLC\models\model_13\test\

(base) C:\Users\user\Desktop\TLC>
```

Figure 5.8: Testing Process

Performance of Agent before Training and testing :

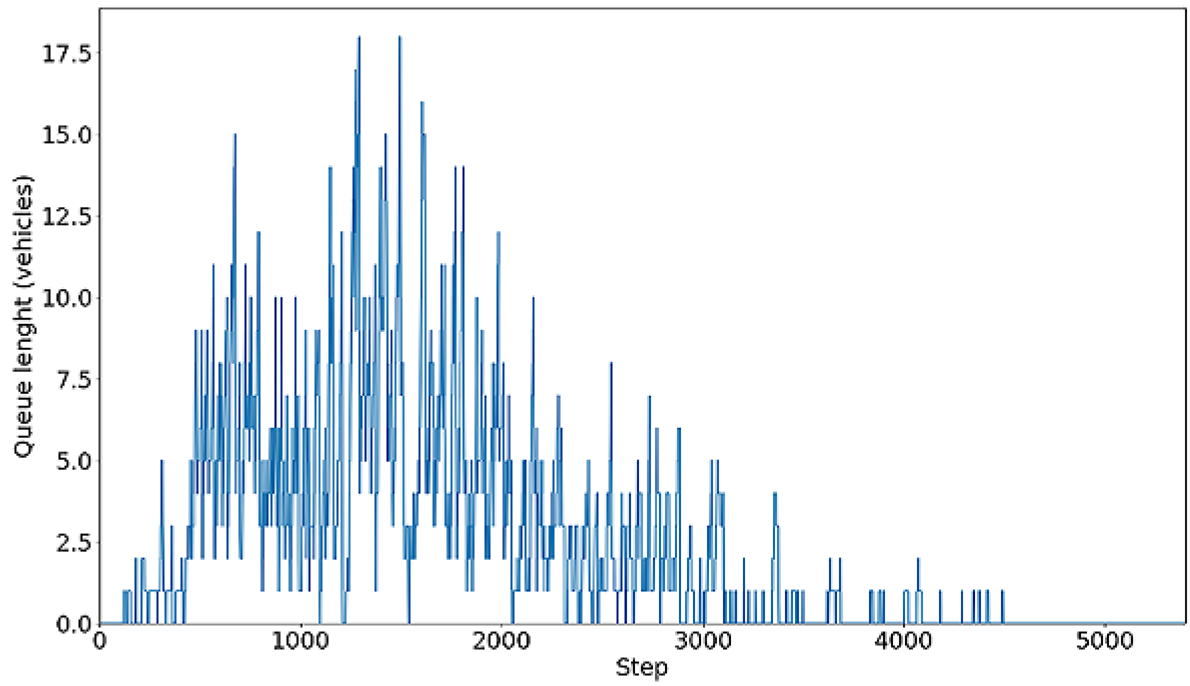


Figure 5.9: Queue length

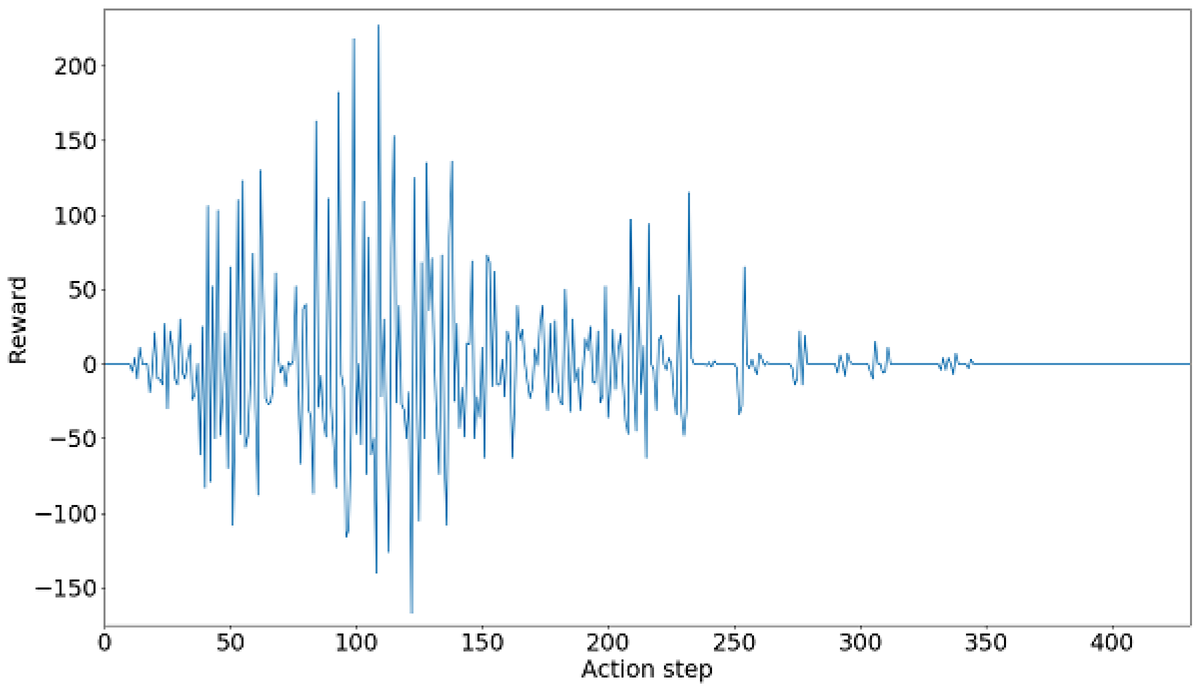


Figure 5.10: Rewards

Performance of Agent after Training and testing :

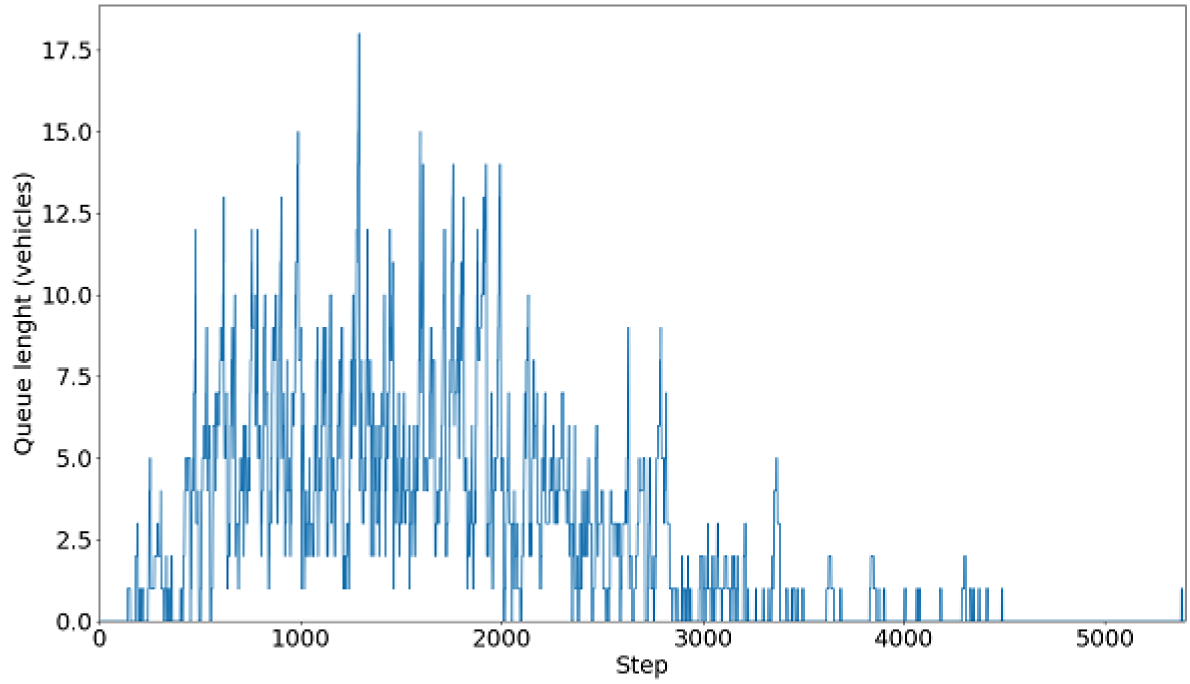


Figure 5.11: Queue length

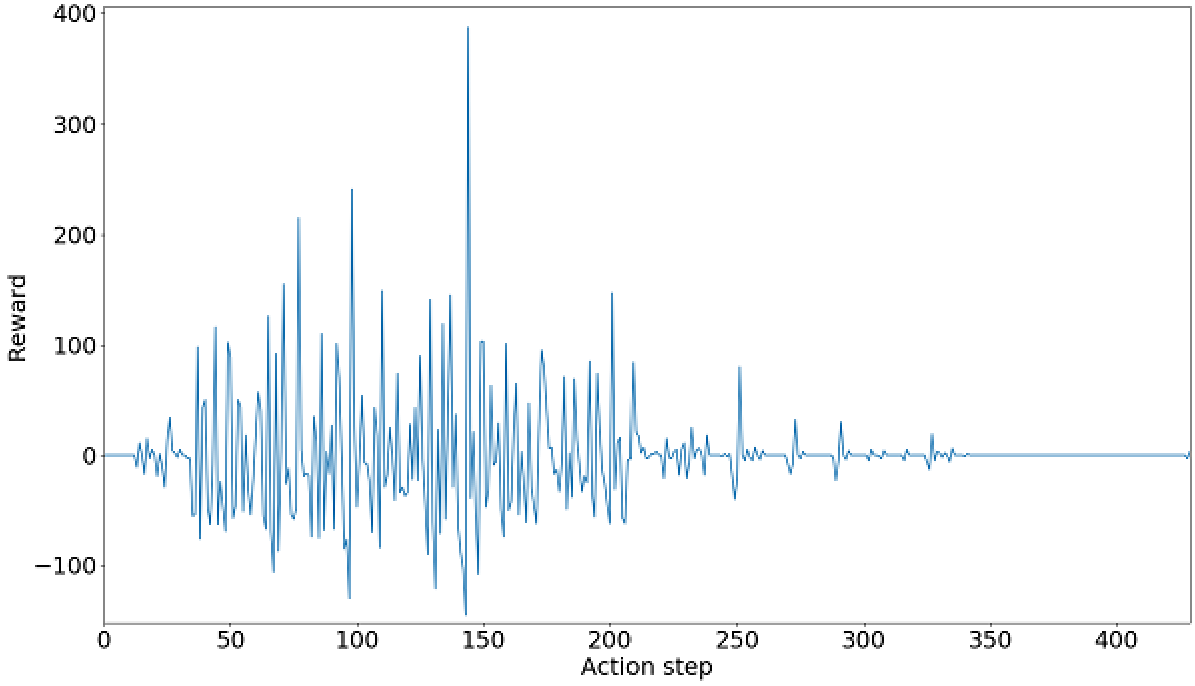


Figure 5.12: Rewards

5.2 Sample Code

training-main.py

```
from __future__ import absolute_import
from __future__ import print_function

import os
import datetime
from shutil import copyfile

from training_simulation import Simulation
from generator import TrafficGenerator
from memory import Memory
from model import TrainModel
from Visualization import Visualization
from utils import import_train_configuration, set_sumo, set_train_path


if __name__ == "__main__":

    config = import_train_configuration(config_file='training_settings.ini')
    sumo_cmd = set_sumo(config['gui'], config['sumocfg_file_name'], config['max_ste
    path = set_train_path(config['models_path_name'])

    Model = TrainModel(
        config['num_layers'],
        config['width_layers'],
        config['batch_size'],
        config['learning_rate'],
        input_dim=config['num_states'],
        output_dim=config['num_actions']
    )

    Memory = Memory(
        config['memory_size_max'],
        config['memory_size_min']
    )
```

testing-main.py

```python
from __future__ import absolute_import
from __future__ import print_function

import os
from shutil import copyfile

from testing_simulation import Simulation
from generator import TrafficGenerator
from model import TestModel
from Visualization import Visualization
from utils import import_test_configuration, set_sumo, set_test_path


if __name__ == "__main__":

    config = import_test_configuration(config_file='testing_settings.ini')
    sumo_cmd = set_sumo(config['gui'], config['sumocfg_file_name'], config['max_ste
    model_path, plot_path = set_test_path(config['models_path_name'], config['model_

    Model = TestModel(
        input_dim=config['num_states'],
        model_path=model_path
    )

    TrafficGen = TrafficGenerator(
        config['max_steps'],
        config['n_cars_generated']
    )

    Visualization = Visualization(
        plot_path,
        dpi=96
    )

    Simulation = Simulation(
        Model
```

training-simulation.py

```
import traci
import numpy as np
import random
import timeit
import os

# phase codes based on environment.net.xml
PHASE_NS_GREEN = 0  # action 0 code 00
PHASE_NS_YELLOW = 1
PHASE_NSL_GREEN = 2  # action 1 code 01
PHASE_NSL_YELLOW = 3
PHASE_EW_GREEN = 4  # action 2 code 10
PHASE_EW_YELLOW = 5
PHASE_EWL_GREEN = 6  # action 3 code 11
PHASE_EWL_YELLOW = 7


class Simulation:
    def __init__(self, Model, Memory, TrafficGen, sumo_cmd, gamma, max_ste
        self._Model = Model
        self._Memory = Memory
        self._TrafficGen = TrafficGen
        self._gamma = gamma
        self._step = 0
        self._sumo_cmd = sumo_cmd
        self._max_steps = max_steps
        self._green_duration = green_duration
        self._yellow_duration = yellow_duration
        self._num_states = num_states
        self._num_actions = num_actions
        self._reward_store = []
        self._cumulative_wait_store = []
        self._avg_queue_length_store = []
        self._training_epochs = training_epochs
```

testing-simulation.py

```
import traci
import numpy as np
import random
import timeit
import os

# phase codes based on environment.net.xml
PHASE_NS_GREEN = 0  # action 0 code 00
PHASE_NS_YELLOW = 1
PHASE_NSL_GREEN = 2  # action 1 code 01
PHASE_NSL_YELLOW = 3
PHASE_EW_GREEN = 4  # action 2 code 10
PHASE_EW_YELLOW = 5
PHASE_EWL_GREEN = 6  # action 3 code 11
PHASE_EWL_YELLOW = 7


class Simulation:
    def __init__(self, Model, TrafficGen, sumo_cmd, max_steps, green_du
        self._Model = Model
        self._TrafficGen = TrafficGen
        self._step = 0
        self._sumo_cmd = sumo_cmd
        self._max_steps = max_steps
        self._green_duration = green_duration
        self._yellow_duration = yellow_duration
        self._num_states = num_states
        self._num_actions = num_actions
        self._reward_episode = []
        self._queue_length_episode = []


    def run(self, episode):
        """
        Runs the testing simulation
        """
```

model.py

```python
import os
os.environ['TF_CPP_MIN_LOG_LEVEL']='2'  # kill warning about tensorflow
import tensorflow as tf
import numpy as np
import sys

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras import losses
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import plot_model
from tensorflow.keras.models import load_model


class TrainModel:
    def __init__(self, num_layers, width, batch_size, learning_rate, input_dim, 
        self._input_dim = input_dim
        self._output_dim = output_dim
        self._batch_size = batch_size
        self._learning_rate = learning_rate
        self._model = self._build_model(num_layers, width)


    def _build_model(self, num_layers, width):
        """
        Build and compile a fully connected deep neural network
        """
        inputs = keras.Input(shape=(self._input_dim,))
        x = layers.Dense(width, activation='relu')(inputs)
        for _ in range(num_layers):
            x = layers.Dense(width, activation='relu')(x)
        outputs = layers.Dense(self._output_dim, activation='linear')(x)

        model = keras.Model(inputs=inputs, outputs=outputs, name='my_model')
        model.compile(loss=losses.mean_squared_error, optimizer=Adam(lr=self._le
```

memory.py

```
import random
class Memory:
    def __init__(self, size_max, size_min):
        self._samples = []
        self._size_max = size_max
        self._size_min = size_min


    def add_sample(self, sample):
        """
        Add a sample into the memory
        """
        self._samples.append(sample)
        if self._size_now() > self._size_max:
            self._samples.pop(0)  # if the length is greater than the siz


    def get_samples(self, n):
        """
        Get n samples randomly from the memory
        """
        if self._size_now() < self._size_min:
            return []

        if n > self._size_now():
            return random.sample(self._samples, self._size_now())  # get
        else:
            return random.sample(self._samples, n)  # get "batch size" nu


    def _size_now(self):
        """
        Check how full the memory is
        """
        return len(self._samples)
```

traffic-generator.py

```
import numpy as np
import math

class TrafficGenerator:
    def __init__(self, max_steps, n_cars_generated):
        self._n_cars_generated = n_cars_generated  # how many cars per ep
        self._max_steps = max_steps

    def generate_routefile(self, seed):
        """
        Generation of the route of every car for one episode
        """
        np.random.seed(seed)  # make tests reproducible

        # the generation of cars is distributed according to a weibull di
        timings = np.random.weibull(2, self._n_cars_generated)
        timings = np.sort(timings)

        # reshape the distribution to fit the interval 0:max_steps
        car_gen_steps = []
        min_old = math.floor(timings[1])
        max_old = math.ceil(timings[-1])
        min_new = 0
        max_new = self._max_steps
        for value in timings:
            car_gen_steps = np.append(car_gen_steps, ((max_new - min_new)

        car_gen_steps = np.rint(car_gen_steps)  # round every value to in

        # produce the file for cars generation, one car per line
        with open("intersection/episode_routes.rou.xml", "w") as routes:
            print("""<routes>
            <vType accel="1.0" decel="4.5" id="standard_car" length="5.0"
            <route id="W_N" edges="W2TL TL2N"/>
            <route id="W_E" edges="W2TL TL2E"/>
```

5.3 Testing

5.3.1 Unit Testing

The goal of unit testing to separate each part of the program and test that the individual parts are working correctly and as intended.
Test Objective: Proper working of the simulator

Table 5.1: Unit Testing

Test Condition	**Input Specification**	**Output Specification**	**Success/Fail**
Working of traffic and traffic light signal	Running .sumocfg file	Cars generated and working traffic signal	Success

5.3.2 Integration Testing

Combine the unit tested module one by one and test the functionality of the combined unit.
Test Objective: To obtain reward, simulation and episode number.

Table 5.2: Integration Testing

Test Condition	**Input Specification**	**Output Specification**	**Success/Fail**
Obtaining reward for each episode	Running training_main.py	Image files of Reward,Simulation time and episode number generated	Success

5.3.3 Blackbox Testing

In BlackBox Testing, we just focus on inputs and output of the software system without bothering about internal knowledge of the software program.
Test Objective: To obtain comparative graphs of queue length and waiting time.

Table 5.3: Blackbox Testing

Test Condition	**Input Specification**	**Output Specification**	**Success/Fail**
Reduction in queue length and waiting time	Running testing_main.py	Image files of graph generated	Success

Chapter 6

Conclusion & Future Scope

In this report, the study of how an intelligent traffic control system using reinforcement learning algorithm can be used to tackle current traffic problems is explained.The learning agent of this system is designed with state representation that identifies the position of vehicle in environment and makes decisions according to real time traffic.Based on the decision agent gets reward which is further used by the agent to make appropriate decisions to reduce traffic on the basis of it's rewards.The system can be designed as a multi-agent system to take decisions for more than one intersection at a time.Also public transport,emergency vehicles like fire brigade,ambulance should be given higher preference.The system requires lane system for its working so an improvisation in the system for functioning on roads without lanes can be implemented.

References

[1] "The history and evolution of traffic lights." https://www.scienceabc.com/innovation/ready-steady-go-the-evolution-of-traffic-lights.html.

[2] R. Zhang, A. Ishikawa, W. Wang, B. Striner, and O. Tonguz, "Intelligent traffic signal control: Using reinforcement learning with partial detection."

[3] I. Arel, C. Liu, T. Urbanik, and A. Kohls, "Reinforcement learning-based multi-agent system for network traffic signal control."

[4] J. Gao, Y. Shen, J. Liu, M. Ito, and N. Shiratori, "Adaptive traffic signal control: Deep reinforcement learning algorithm with experience replay and target network."

[5] "Reinforcement learning for intelligent traffic light control."

[6] "Multi-agent reinforcement learning for intelligent traffic light control."

[7] "Weibull distributions and their applications." https://www.researchgate.net/publication/37628953_Weibull_Distributions_and_Their_Applications.

Acknowledgement

Success of a project like this involving high technical expertise, patience and massive support of guides, is possible when team members work together. We take this opportunity to express our gratitude to those who have been instrumental in the successful completion of this project. We would like to show our appreciation to **Ms. Dakshayni** for their tremendous support and help, without them this project would have reached nowhere. We would also like to thank our project coordinator **Mrs. Rakhi Kalantri** for providing us with regular inputs about documentation and project timeline. A big thanks to our HOD **Dr. Lata Ragha** for all the encouragement given to our team. We would also like to thank our principal, **Dr. S. M. Khot**, and our college, **Fr. C. Rodrigues Institute of Technology, Vashi**, for giving us the opportunity and the environment to learn and grow.

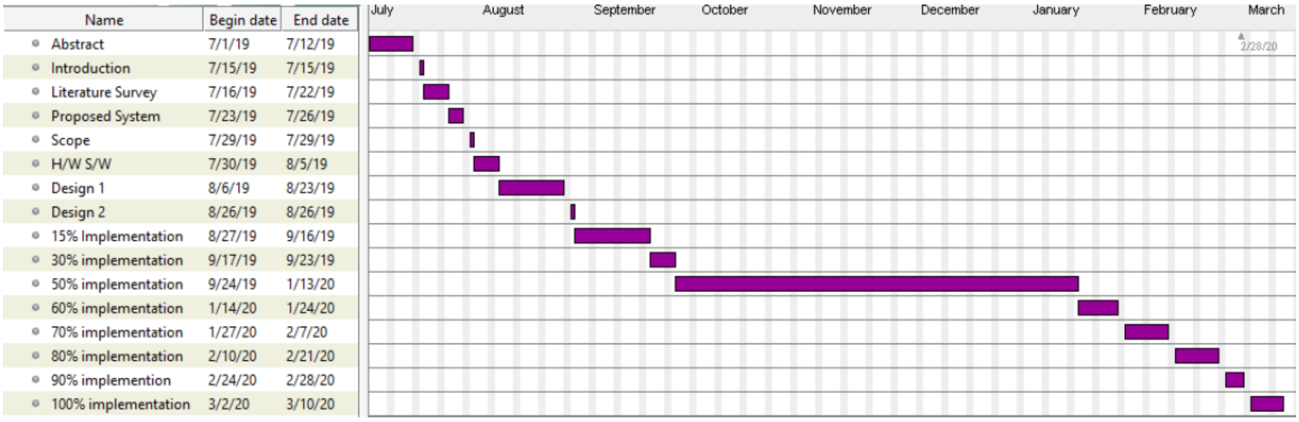
Project Group Members:

1. Kranti Shingate 101652

2. Komal Jagdale 101669

3. Yohann Dias 101667

Appendix A : Timeline Chart



Name	Begin date	End date
Abstract	7/1/19	7/12/19
Introduction	7/15/19	7/15/19
Literature Survey	7/16/19	7/22/19
Proposed System	7/23/19	7/26/19
Scope	7/29/19	7/29/19
H/W S/W	7/30/19	8/5/19
Design 1	8/6/19	8/23/19
Design 2	8/26/19	8/26/19
15% Implementation	8/27/19	9/16/19
30% implementation	9/17/19	9/23/19
50% implementation	9/24/19	1/13/20
60% implementation	1/14/20	1/24/20
70% implementation	1/27/20	2/7/20
80% implementation	2/10/20	2/21/20
90% implemention	2/24/20	2/28/20
100% implementation	3/2/20	3/10/20

Figure 6.1: Timeline chart

Appendix B : Publication Details

6.0.1 Publication details

Adaptive Traffic Control System using Reinforcement Learning, IJERT Publication,Volume -9, Issue -2, February 2020
https://www.ijert.org/adaptive-traffic-control-system-using-reinforcement-learning