Traffic Light Control using Deep Reinforcement Learning

**Traffic Light Control using Deep Reinforcement Learning**

A project report submitted in partial

fulfillment of the requirements for the degree of

Master of Science

By

FNU Divyani

Cochin University of Science and Technology

Bachelor of Technology, in Information Technology

May 2023

University of Colorado Denver

ABSTRACT

In today's world, urban areas have experienced exponential growth in population and traffic, which leads to greater traffic congestion, delays, increased air pollution, and fuel consumption. As a result, we see increased traffic congestion at intersections. To mitigate this problem, the introduction of intelligent traffic which is dynamic and responds better to real time traffic status is an effective solution. Our project aims to construct a model for a "Traffic Light Control System using Deep Reinforcement Learning," using neural networks and Q Learning techniques.

This Project Report is approved for recommendation to the Graduate Committee.

Project Advisor:

Dr. **Ashis Kumer Biswas**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

©2023 by FNU Divyani

All Rights Reserved

TABLE OF CONTENTS

[1. Introduction 1](#_Toc208640847)

[1.1 Problem 1](#_Toc208640848)

[1.2 Objective 1](#_Toc208640849)

[1.3 Approach 1](#_Toc208640850)

[1.4 Organization of this Project Report 1](#_Toc208640851)

[2. Background 2](#_Toc208640852)

[2.1 Key Concepts 2](#_Toc208640853)

[2.1.1 Key Concept 1 2](#_Toc208640854)

[2.1.2 Key Concept 2 2](#_Toc208640855)

[2.2 Related Work or Literature Review 2](#_Toc208640856)

[2.2.1 Area 1 2](#_Toc208640857)

[2.2.2 Area 2 3](#_Toc208640858)

**3. Project Setup…………………………………………………………………………6**

[4 Architecture (or Approach or Model …) 7](#_Toc208640859)

[4.1 High Level Design 8](#_Toc208640860)

[4.2 Implementation 9](#_Toc208640862)

[5. Methodology, Results and Analysis (or similar title) 10](#_Toc208640863)

[5.1 Methodology 13](#_Toc208640864)

[5.2 Results 14](#_Toc208640865)

[5.3 Analysis 14](#_Toc208640866)

[6. Conclusions 15](#_Toc208640867)

[5.1 Summary 16](#_Toc208640868)

[5.2 Contributions -or- Potential Impact 16](#_Toc208640869)

[5.3 Future Work 16](#_Toc208640870)

[References 17](#_Toc208640871)

[Appendix A. Title (Optional) 8](#_Toc208640872)

LIST OF FIGURES

Figure 1: **Traffic Lights and Intersection in SUMO** 3

Figure 2: **Reinforcement learning model** 4

Figure 3**: Q Learning** ……………………………………………………………………5

Figure 4: **DQN Archtechure** ………………………………………………………….13

# 1. Introduction

There are intersections and traffic signals on every street in the world, and with daily traffic congestion rising, the demand for intelligent and dynamic traffic signal control becomes more and more necessary. According to the Texas A&M Transportation Institute traffic congestion in the US costs an estimated $166 billion per year in lost productivity and wasted fuel. This project aims to provide a mechanism for controlling traffic signals in a dynamic manner using traffic simulation software “SUMO” and reinforcement learning.

## 1.1 Problem Statement

In the current world ,especially in metropolitan areas, population has significantly increased resulting in the increased number of cars more than ever .This has resulted in more traffic jams , delays and ultimately higher levels of air pollution and increased fuel consumption overall.Many times traffic problems are caused due to fixed timing durations of traffic control lights and inefficiency in adjusting to real time traffic load.That is why we need a more intelligent and effective traffic control system to tackle the existing problem.This system aims to build a traffic light control which adjusts and optimizes to real time traffic on the intersections.

## 1.2 Objective

The objective of this project is to build a Traffic light control using Deep Reiforcement Learning technique.

## 1.3 Approach

For solving the Traffic light control problem, we’ll be using the the Reinforcement learning approach with neural networks. I have taken real world maps and used that to create a realistic junction and traffic scenario. The street maps are taken from the website “OpenStreetMap” and then that map is visualized in the traffic simulation open-source software called SUMO (short for Simulation of Urban MObility). Using SUMO’s tool ‘randomTrips’ 200 random traffic trips were generated for each real-world map. Maps and traffic simulations are imported into Python code using a Python API called TraCI (short for Traffic Control Interface).

For training the traffic network & signal we use Q Learning method with neural networks as an approximate value function for the Q learning. For rendering the traffic simulation through Python “sumo-gui” application was used. The goal of the DQN (short for Deep Q network) will be to reduce the overall waiting time and queue length at every junction of the traffic network map. For evaluating the effiecieny and result of our approach we will plot a graph of average waiting time vs training epoch number.

## 1.4 Organization of this Project Report

Chapter 2 covers Background & Key concepts

Chapter 3 covers Project Setup

Chapter 4 covers Architechure

Chapter 5 covers Methodology.

Chapter 6 covers Conclusion

# 2. Background

## 2.1 Key Concepts

For the purposes of this project, the following major ideas have been used to study and explore the techniques of Traffic networks & Deep Q network.

### 2.1.1 SUMO (Simulation of Urban MObility)

"**S**imulation of **U**rban **MO**bility" (SUMO) is an open source, highly portable, microscopic and continuous traffic simulation package designed to handle large networks. It provides a realistic enavironment for traffic simulation and optimization. It is frequently utilized in the field of transportation research for designing and testing sophisticated traffic control algorithms as well as for researching and improving traffic flows in urban regions. SUMO provides a Python API TraCI (short for Traffic Control Interface) which basically connects Python code and SUMO software. Through this we can run traffic simulations and retrieve and modify all SUMO environment values in Python.

**2.1.2** **Traffic Lights and Intersection in SUMO**

A picture containing logo

Description automatically generated

In SUMO, traffic lights and intersections can be defined to create real-world traffic scenarios for simulation and visualisation. The figure shows how intersections and traffic lights are visualed in SUMO software. The red, green and yellow lights are showing the state of the traffic signal now. The values of network and traffic trips in SUMO are stored in a xml file format. So, for traffic signals, the values look like this:

Text

Description automatically generated

Individual traffic signals are represented with tag <tlogic> it has a unique ID and phase describes the phases of the traffic signal with their respective duration and states. The state string represents the combination of r, G, g, y where lower case alphabets represent red, green, yellow states, but the upper case ‘G’ represents the priority green signal. By manipulating these phases and duration we can control the traffic flow and optimize the wait times and queues.

**2.1.3 Reinforcement learning**

Using feedback from its own actions and experiences, Reinforcement Learning (RL), a form of machine learning

technique, enables an agent to learn in an interactive environment by trial and error.

Diagram

Description automatically generated

Fig 1 : https://www.researchgate.net/figure/a-Reinforcement-learning-architecture-b-Deep-reinforcement-learning-architecture\_fig7\_339873542

Key concepts of Reinforcement Learning:

1. Environment – It is the space in which our RL agent interacts with
2. State —describes the situation of the agent.
3. Reward — Feedback the agent receives from the environment.
4. Policy **—**Method to map agent’s state to actions.
5. Value function: It is the expected value of an agent in a certain state.

**2.1.4 Q Learning**

Q Learning is a type of reinforcement learning technique which uses bellman equation to train the agent in an environment. The bellman equation goes like this:

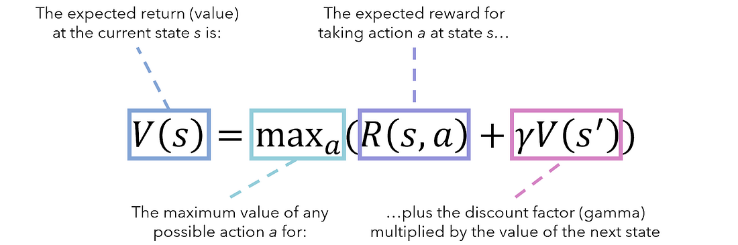


Fig 2 : [A Crash Course in Markov Decision Processes, the Bellman Equation, and Dynamic Programming | by Andre Ye | MLearning.ai | Medium](https://medium.com/mlearning-ai/a-crash-course-in-markov-decision-processes-the-bellman-equation-and-dynamic-programming-e80182207e85)

Where V(s) is the expected value on state s, R (s, a) is the reward that we get on state s, by performing action ‘a’ and γ V(s’) is the discounted value of the next state that agent will end up to. γ is the discount factor which determines how important the next stage is for value function. To implement Q learning in code we use Q tables, which is a lookup table that stores return values for all possible state, action.

**2.1.5 Deep Q-Network**

The problem with Q learning is that it takes discrete values and doesn’t work efficiently for continuous values. If we feed continuous values into this approach, the size of Q Table will be too big and complex to handle.

To tackle this problem, we use the Deep Q Learning approach which also works on continuous values. Here we are using neural networks as function approximator. The objective of function approximator is to generalize the estimation of the value at states that have comparable features using this set of attributes. Here the neural network takes the states value and tries to predict the Q values for it.

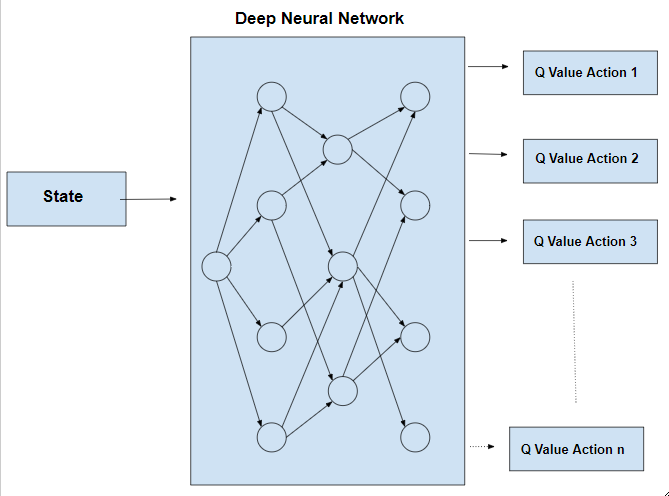


Fig 3: https://www.geeksforgeeks.org/deep-q-learning/

## 2.2 Related Work or Literature Review

There have been several related work and researchers who have tried to approach the same problem using SUMO software and reinforcement learning. The main two works which I found most useful while doing this project are listed below.

### 2.2.1 CityFlow: A Multi-Agent Reinforcement Learning Environment for Large Scale City Traffic Scenario

This paper uses multi-Agent reinforcement learning model to train different types of grid networks on street intersection. I have only used a single 1 x 1 type of network and a a single RL agent in my project. This paper uses different types of intersection networks to create a more generic model which can be used for vast types of street maps across the world.

### 2.2.2 Towards Real-World Deployment of Reinforcement Learning for Traffic Signal Control

This paper also does Deep Reinforcement Learning with SUMO as simulation software. They focus more on demand on the lane, and pedestrian queue length factor for shaping the model.

### 

# 3. Project setup

First install the SUMO software from link: https://sumo.dlr.de/docs/Downloads.php

OpenStreetMap is a free website from where we can download any type of street map across the world. To download the map:

1. Search the street or area by name in the search bar given on the website.

Map

Description automatically generated

1. Press Go and then select the region you want to import and export that region as .osm file.

After the map is now installed in OSM format, we must convert that map into SUMO appropriate format.

1. First copy the osmNetconvert.typ.xml from C:\Program Files (x86)\Eclipse\Sumo\data\typemap which will be inside SUMO main folder to our project folder.
2. Then we have to run the given command which will convert the osm file into SUMO compatible XML file.

Command: netconvert --osm-files map.osm -o test.net.xml -t osmNetconvert.typ.xml --xml- validation never

1. Generate the random 200 traffic for the given street map: To do that first we copy RandomTrips.py file from SUMO folder to our project folder then we have to run the given below command.

Command: python randomTrips.py – test.net.xml -r map.rou.xml -e 200 -1

This will create a trips.trips.xml file which contains information about the 200 traffic trips on the given map.

1. Then we created a map.sumo.cfg to merge the network map and trips file. Create the file like this, give respective path in net-file and route-files tags :

Text

Description automatically generated

After the SUMO setup is done install all the necessary API and libraries like TraCI using pip.

# 4 Architechure

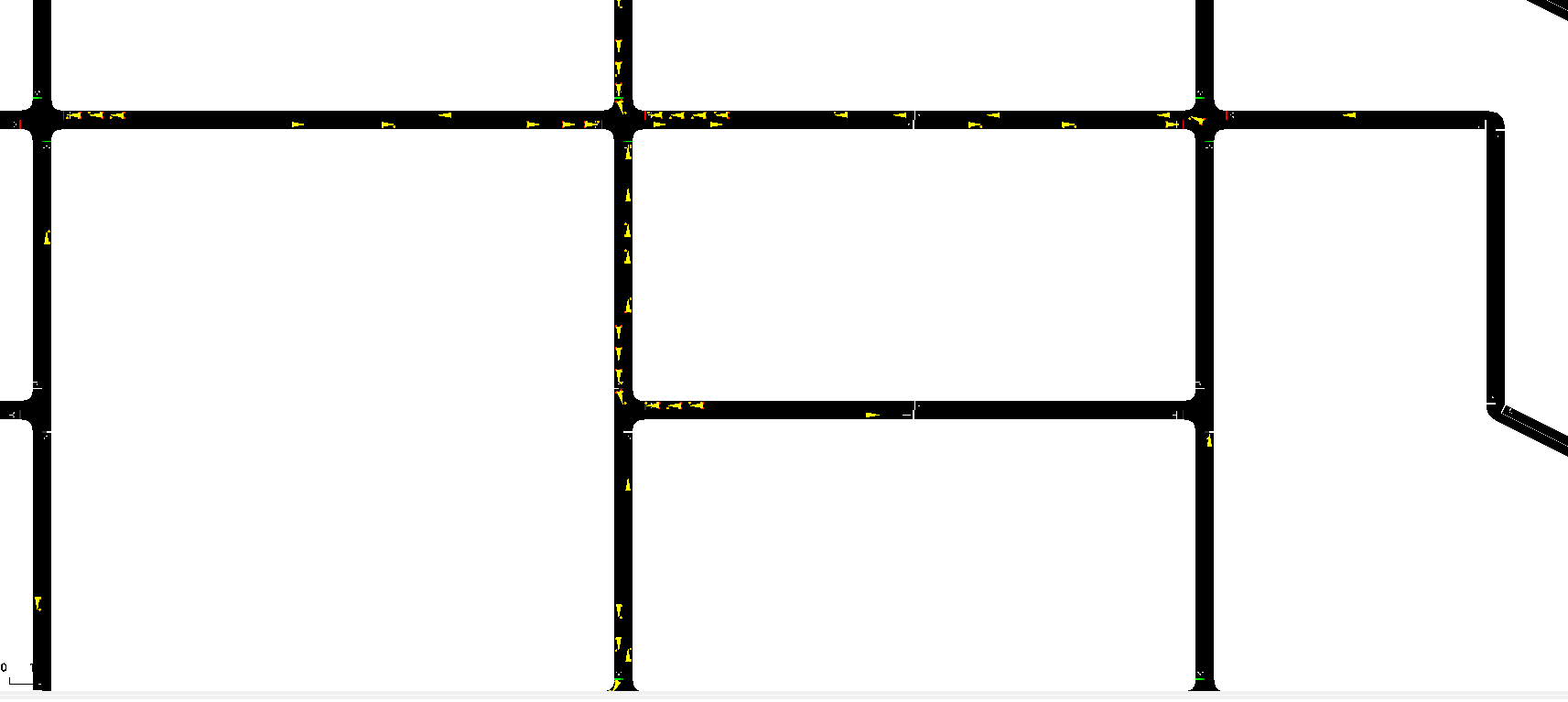
## 4.1 [High Level Design](https://docs.google.com/document/d/1hg9tp5W6UkVGjgJUHue8Df-1WCt4eOq9/edit" \l "heading=h.35nkun2)

We can devide the project design into several phases given below:

1) Data & Network Collection: In this phase we collect the network maps from real-world streets and generate realistic traffic trips for it. This is the data that we worked on and modified for the entirety of the project.

Then we identified the environment, states and action space, training epochs, testing and reward criteria.

2)Environment: The environment for this project is the SUMO traffic map and traffic trips included in it. This includes the junctions, traffic lights, cars etc present on the map.



3)State/Observation Space: The state of the environment is described by the total waiting time by all cars for each lane at the traffic signal and queue length at each lane/traffic signal.

4)Actions: The possible action taken by the agent will be to change the phases of the traffic signal and modify the duration of the signal.

5)Training: The training for the agent will be done by the Deep Q Learning model using Keras library for neural network implementation. The architechure for Deep Q Network look like:

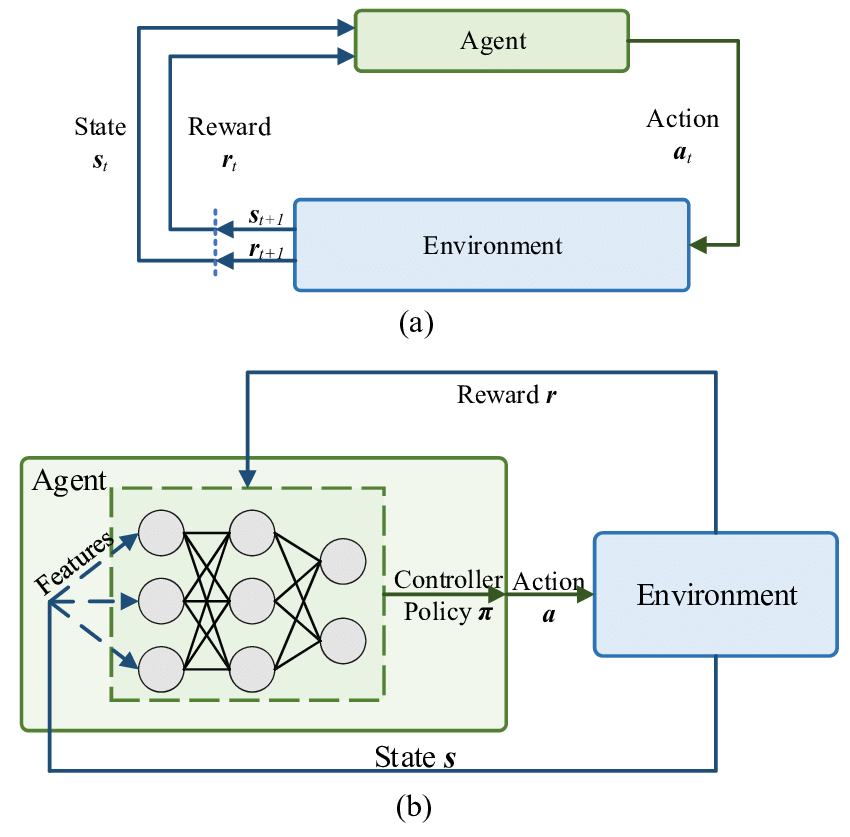


Figure 4: link : https://www.researchgate.net/figure/a-Reinforcement-learning-architecture-b-Deep-reinforcement-learning-architecture\_fig7\_339873542

## 4.2 Implementation

We can devide the implementation into several phases given below:

Set up the simulation environment using SUMO and TraCI. The step for the setup is described in section 3.

To setup SUMO simulation in Python, we must provide path for exe file and simulation config file:

Text

Description automatically generated

Then gain information about the traffic signals from the xml file like phases, durations using parse XML

Text

Description automatically generated

Then defined the observation and action space like this based on the traffic light attributes.

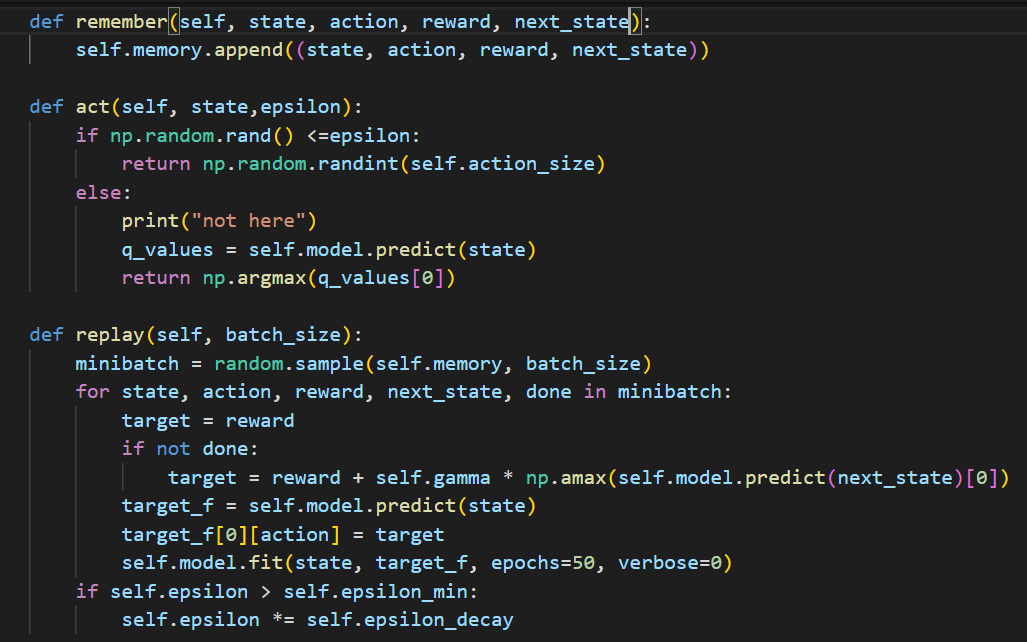
A screenshot of a computer

Description automatically generated with medium confidence

Then defined the neural network which is going to be usedText

Description automatically generated for DQN like this: This code block creates a neural network model using the Keras API of TensorFlow library. The model has three fully connected layers. The first two layers have 64 nodes with the Rectified Linear Unit (ReLU) activation function, while the output layer has nodes equal to the number of actions with a linear activation function.

The define the rest of the DQN class like this:



Act function is taking the current state of the environment and predicting action in a Exploration – Exploitation manner. In the exploration phase, we are selecting random actions and letting the agent explore more states, action pairs. In the exploitation phase we are choosing the action in a greedy manner, like choosing the action which yields the best reward at the given time step.

Remember function stores all the action, states, reward for all the transitions in the memory.

The replay phase takes the all the parameters from the memory in batches, makes the Q values as target variable and states as an input variable and tries to learn and predict the Q values accordingly.

# 5. Methodology, Results and Analysis (or similar title)

After training the model, to test it we ran the testing simulation and saw the total waiting time for all the cars in the network. We saw the overall decrease in the total waiting times for the cars on the network.

## 4.1 Testing Methodology

For the testing methodology, we will run the simulation again using the learning agent this through the model. Here we will put the epsilon value as 0, so that we are able to choose greedly and use the learning.

To compare the progress we first ran the simulation before training, collect the waiting times and queue length and then do the training and run the simulation again after training like this :

Text

Description automatically generated

**4.2 Results**

Note: The time is in miliseconds

Before Training

Graphical user interface

Description automatically generated

After Training

A screenshot of a computer

Description automatically generated with medium confidence

## 4.3 Analysis

Based on the results, it appears that the implementation of the DQN agent in traffic simulation has led to a slight decrease in wait time for vehicles at intersections. We were succefully able to run and visualize our desired result with SUMO software. The Deep Reinforcemrnt learning works for continuous values and can solve the dimensionality problem for simple Q Learning through Q tables.

# 6. Conclusions

## 5.1 Summary

Based on the results, it appears that the implementation of the DQN agent in traffic simulation has led to a slight decrease in wait time for vehicles at intersections.

To achieve the results, we collected the data from OpenstreetMap and then used SUMO software for visualization and trained the model using Deep Reinforcement learning method. For testing we ran the traffic simulation on SUMO with TraCI API before and after the agent training and compared the results.

Also, we saw that the results and our project success depend on the data, quality of maps we collect, how well we understand the traffic simulation and intersection logic. Based on this we say we could’ve improved on these pointers and could achieve even better results.

## 5.2 Contributions/Potential Impact

We know that traffic congestion at the intersection is a worldwide phenomenon and how it is often caused by non-dynamic nature of traffic signals. If we were to implement these dynamic traffic signals more in real life street intersections, we could probably see less traffic congestion on the intersections, which means less fuel consumption and ultimately less pollution.

## 5.3 Future Work

In this project I only considered the basic type of intersection, which has 4 lanes and constructed my project according to it. I would consider different types of intersections, like the one with 6 lanes. Also in the future work, we could consider the state of pedestrians on the street as well. For achieving this type of solution, we would have to work with multi agent RL models.

# References

[1] Zhang, H., Feng, S., Liu, C., Ding, Y., Zhu, Y., Zhou, Z., Zhang, W., Yu, Y., Jin, H., & Li, Z. (2019, May 13). *CityFlow: A multi-agent reinforcement learning environment for large scale city traffic scenario*. arXiv.org. Retrieved May 1, 2023, from https://arxiv.org/abs/1905.05217

[2] Müller, A., Rangras, V., Schnittker, G., Waldmann, M., Friesen, M., Ferfers, T., Schreckenberg, L., Hufen, F., Jasperneite, J., & Wiering, M. (2022, January 11). *Towards real-world deployment of reinforcement learning for Traffic Signal Control*. arXiv.org. Retrieved May 1, 2023, from https://arxiv.org/abs/2103.16223

[3]

**A Crash Course in Markov Decision Processes, the Bellman Equation, and Dynamic Programming**

byAndreYe

[4] Machine Learning for Intelligent Optical Networks: A Comprehensive Survey - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/a-Reinforcement-learning-architecture-b-Deep-reinforcement-learning-architecture\_fig7\_339873542 [accessed 2 May, 2023]

[5] Deep Q Learning by Alind Gupta

Link : [Deep Q-Learning - GeeksforGeeks](https://www.geeksforgeeks.org/deep-q-learning/)

This is the final page of a Project Report and should be a blank page