Adaptive Traffic Signal Control for Urban Intersections Using Reinforcement Learning: A SUMO Simulation Case Study on Gulshan-2

by

MD Shahadat Hossain Shamim
22101174
Tawhid Chowdhury
22101182
Sadid Arman Deep
22101042
Riazul Hoque Bhuban
23341126

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
BRAC University
February 2025

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at BRAC University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Tawhid Chowdhury	Sadid Arman Deep
22101182	22101042
Riazul Hoque Bhuban	MD Shahadat Hossain Shamim
23341126	22101174

Approval

The thesis titled "Adaptive Traffic Signal Control for Urban Intersections Using Reinforcement Learning: A SUMO Simulation Case Study on Gulshan-2" submitted by

1. Tawhid Chowdhury (22101182)
2. Sadid Arman Deep (22101042)
3. Riazul Hoque Bhuban (23341126)
4. MD Shahadat Hossain Shamim (22101174)

of Fall, 2025 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on January 31,2026.

Examining Committee:

Supervisor:
(Member)

Amitabha Chakrabarty, PhD

Professor
Department of Computer Science and Engineering
BRAC University

Co-Supervisor:
(Member)

Md. Golam Rabiul Alam, PhD

Professor
Department of Computer Science and Engineering
BRAC University

Thesis Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD

Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD

The Chairperson
Department of Computer Science and Engineering
BRAC University

Abstract

Traffic congestion is an ever-growing concern in rapidly increasing population and increase of vehicles on roads. On top of that, our road condition, pedestrian behavior, and driving behavior makes the situation much worse. As Bangladesh has started adapting to the fixed-time traffic light system, finding the optimal adaptive traffic light management system is required to mitigate the issue. Keeping the lack of intra-vehicular communication devices in mind, we have studied different traffic light controlling systems and how these perform on Bangladeshi roadways. In our study, we have used traffic simulator to simulate Gulshan-2 intersection, one of the major congestion points in Dhaka city. Based on traffic data, we have created a vehicular network system to find out how the existing traffic light models per- form on mitigating the traffic jam using an adaptive traffic management system. In our study, we simulate the Gulshan-2 intersection using a realistic traffic dataset derived from field data and analyze three control models, Fixed-Time, Q-Learning, and Deep Q-Network (DQN). The results show that in ideal traffic condition, when no anomalies are present; the Q-Learning controller reduced average vehicle delay by approximately 60-70% and improved throughput by 35-45% compared to the fixed-time system, demonstrating its superior adaptability to dynamic traffic flows. However, when we add real-life anomalies such as potholes and jaywalking, the baseline controllers' performance drops. By acknowledging the drawbacks of QL controller in abnormal scenarios, we have introduced a hybrid model CRQL (Congestion Responsive Queue Learning) and made some improvements. In abnormal situation CRQL performs better in regard to throughput, which is a 76.9% improvement over Fixed-Time, 37.4% improvement over Q-Learning and 215.2% improvement over DQN. This reflects a overall performance improvement under disruptive and non-stationary environment.

In short, the study helps to see how existing models perform under real-life anomalies, which factors are significant to create Ad hoc model for our extreme pedestrian patterns and road conditions and how our proposed solution; the CRQL ad hoc or hybrid model performs to address the problems.

Keywords: IoT-based Network, Machine Learning, Deep Learning, Adaptive Traffic Signal Management, Traffic jam solution, SUMO

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the background of traffic congestion in Bangladesh, outlines the motivation and problem statement of the study, and defines the objectives, scope, and challenges addressed in this thesis.

1.1 Background

Traffic congestion is a huge curse to our nation as it wastes a huge amount of time and fuel, causes mental distress, and pollutes the environment. According to [7], every day around 3.2 million work hours are wasted due to traffic jams. Collectively, 19 million working hours per day are lost, which is worth Tk 1.53 billion [11]. The traffic jam creates agitation, and the drivers rush to make up the wasted time, which causes accidents. According to the [38] around 34 people (per 100,000 people) died due to road accidents in the year 2023. The population is ever-growing, but the roads are limited. The current scenario of traffic index in Bangladesh since mid-2019 is alarming [37] see figure (1.1). The rate of increase of vehicles in both Dhaka city and overall Bangladesh is increasing [36] see figure (1.2, 1.3). The amount of fuel waste is also very high. Around 40 percent of fuel is wasted [14]. Due to being a hot, humid, and overpopulated country, people stuck in traffic jams feel physical and mental discomforts [6]. Dhaka is the capital of Bangladesh, which is the world's most densely populated city and currently ranks 8th in context to population [40]. This city is ranked 3-5 in global traffic ranking [37]. In the paper [23], it was mentioned that a lack of public transportation, inadequate road infrastructure, an unregulated increase in the number of private cars, and poor traffic management are some of the things that make traffic on the roads in Dhaka so bad. Moreover, the average traffic speed is extremely low mentioned in [25], it was mentioned to be only 4.8 km/h. One study by the Center for Policy Dialogue (CPD) showed that traffic jams in Dhaka waste 23 minutes per hour on average [31]. Keeping the lack of resources in mind, reconstructing all the roads and infrastructure is difficult for a country like Bangladesh. So, we should think of a solution that can be applicable without changing much of the roads and infrastructure.

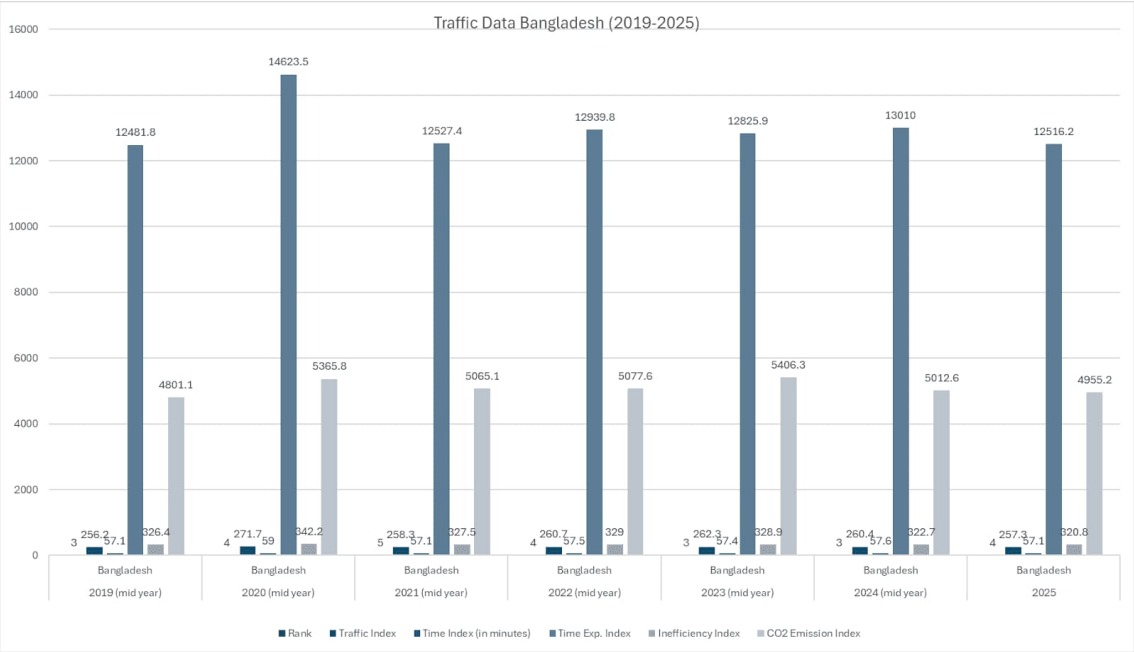


Figure 1.1: Traffic Data Bangladesh (2019-2025)

1.2 Rationale of the Study or Motivation

To mitigate the situation, Bangladesh has already considered reinstalling traffic lights of 22 intersections of Dhaka [39]. This creates a scope to study how the traffic lights should operate and how the introduction of adaptive traffic signal control as part of an Intelligent Traffic Management System (ITS) can help to save time and reduce congestion-related losses. As the traffic congestion problem is rapidly increasing and Bangladesh is slowly shifting towards smarter solutions, we have studied which algorithms perform better. As these algorithms often neglect real-life problems, we studied how performance changes when traffic flow disruptive behaviors such as road wear, potholes and jaywalking are introduced into the system. This study is motivated by the need for a real-world, situation-aware, robust Intelligence Traffic System that is reliable in traffic anomalies that disrupt the traffic flow. We aim to study the popular baseline system and propose a hybrid model that performs better under real-world anomaly situations.

1.3 Problem Statement

There has been numerous research regarding traffic congestion in past years. Most of the works presented the causes of traffic congestion and the issues it creates. Some researchers have tried out different Machine Learning and Deep Learning models including Q-Learning and Deep-Q-Networks to control traffic lights to see which models work the best. However, Bangladeshi roads and their drivers are unpredictable, which may lead the model to behave abnormally. We have studied how these models behave in different real-life road and vehicular scenarios. The problem is to design and evaluate a traffic signal control strategy that remains robust and maintains good performance when anomalies are present.

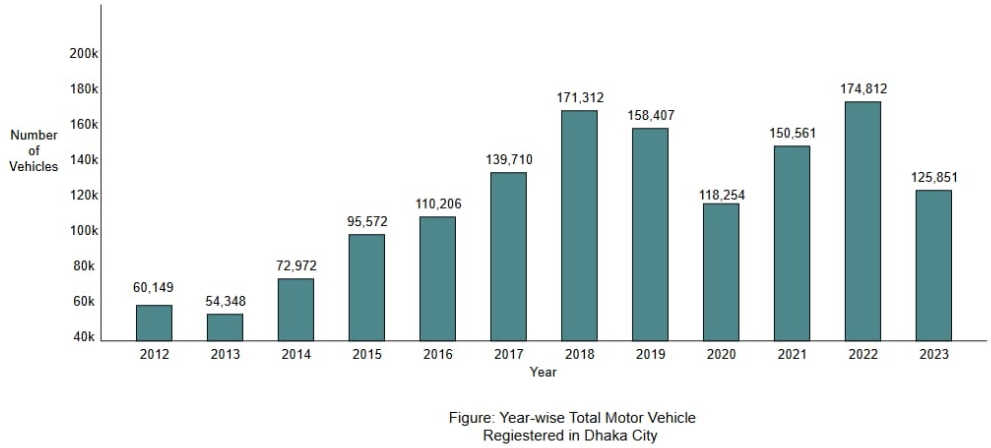


Figure: Year-wise Total Motor Vehicle Regiestered in Dhaka City

Figure 1.2: Year-wise Total Motor Vehicle Registered in Dhaka City

1.4 Objective

The objectives of this research are:

- To construct a realistic SUMO-based simulation environment for the Gulshan-2 intersection.
- To implement and evaluate a traditional Fixed-Time controller.
- To implement and evaluate Q-learning and DQN traffic signal controllers.
- To introduce realistic anomalies (potholes and jaywalking) into the simulation and observe how baseline performance changes.
- To propose and evaluate a hybrid model (CRQL) that improves performance under anomalies.

1.5 Methodology in Brief

To address this issue from a computer science point of view, we studied how modern adaptive traffic light algorithms such as Fixed Timing, Queue Learning (a form of Machine Learning), and Deep Queue Network (a form of Deep Learning) work with the help of the SUMO (Simulation of Urban MObility) simulator. We have used the in-built IoT system of SUMO to detect queue lengths and keep track of traffic flow to run the algorithms and study how they mitigate traffic congestion.

Most of the traffic lights across the world typically use fixed timing-based traffic lights [2]. The popular algorithms work on a cycle-based approach where a predetermined time is used for each cycle to alternate between inbound and outbound lanes, switching between stop (red) and go (green) signals. This approach works for cities with low traffic or disciplined drivers, but it fails in unexpected conditions such as accidents, extreme jams, or empty roads.

Some of the modern adaptive systems like SCOOT (Split Cycle Offset Optimization

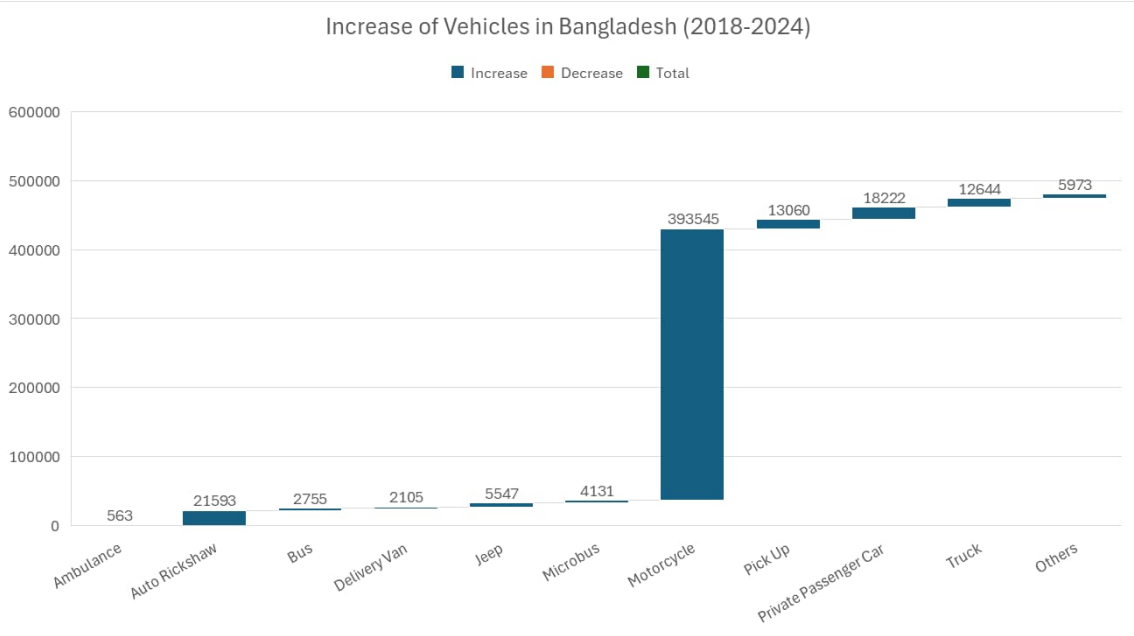


Figure 1.3: Increase of Vehicles In Bangladesh (2018-2024)

Technique) and SCAT (Sydney Coordinated Adaptive Traffic System) have been introduced to make real-time adjustments. SCOOT optimizes, whereas SCAT uses heuristic feedback [2].

Q-Learning, a reinforcement learning method, can adapt to real-life traffic scenarios through continuous interaction with the environment and feedback in the form of rewards or punishments [22]. This property allows it to manage dynamic traffic efficiently. Reinforcement learning is widely applied in fields like autonomous driving [19], decision planning [10], [30], and control execution [9], [24].

Deep-Q-Network (DQN), an extension of Q-Learning, uses deep neural networks to estimate Q-values and is better suited for complex, large-scale environments like urban traffic. The paper [34] shows that DQN outperforms QL in both control and no-control conditions. In our study, we have evaluated its performance compared to QL and fixed-timing approaches.

In summary, we build a SUMO simulation of the Gulshan-2 intersection using OSM-based network generation and GIS (Geographic Information Systems). We control the traffic lights using TraCI. We implement four control approaches: Fixed-Time, Q-Learning, DQN, and the proposed CRQL model. We run simulations with and without anomalies and measure average delay, cumulative reward, queue length, and throughput. Comparative results are analyzed to validate the contribution.

1.6 Scopes and Challenges

Scope: This study is conducted on one intersection of one of the most densely populated Dhaka City's Gulshan-2 intersection

Challenges:

- Designing real-world irregularities such as potholes and jaywalking

- Reinforcement learning instability under non-stationary disruptions
- Ensuring fair comparisons between multiple controllers
- Designing a hybrid model to mitigate flaws of existing models in the real-world traffic situation

1.7 Thesis Outline

The thesis is outlined in the following way:

In chapter 2, we have discussed the related existing works and stated the research gap we have found. By reviewing the papers, we have seen conventional fixed time based traffic signal controller is widely used. However, modern reinforcement learning based traffic controllers using Machine Learning and Deep Learning are widely studied. We have observed that these models are evaluated under ideal conditions. But in developing countries, there are various anomalies in traffic environment. There is not much of a study in this regard.

The chapter 3 defines the functional and non-functional requirements of the proposed traffic control system and outlines the simulation constraints. It also discusses societal, environmental, ethical, and economic impacts relevant to adaptive traffic signal control in Dhaka. The chapter concludes with project planning, risk management, and reproducibility considerations to ensure fair and realistic evaluation.

Chapter 4 describes the methodology of our study. We set up the environment in SUMO simulator, extract the dataset and input the parameters. Then we evaluate the simulation under ideal conditions using Fixed Time, Q-Learning and DQN controllers. Next, we insert anomalies in the simulation environment and see that the performance degrades. By addressing the performance drop, we propose a hybrid congestion aware Q-Learning traffic controller(CRQL).

In Chapter 5, we evaluate the model performance by evaluating the average delay, total queue length, cumulative reward and most importantly throughput. We observe that CRQL is robust in abnormal traffic conditions. It performs effectively under our simulated abnormal traffic environment of Gulshan-2.

Chapter 2

Related Work

This chapter reviews existing literature on traffic signal control systems, with a focus on conventional methods and machine learning–based adaptive approaches, and identifies research gaps relevant to real-world traffic abnormalities.

2.1 Preliminaries

Traffic congestion has become an unbearable challenge in modern cities due to an increase in the traffic vehicles and poor road construction. It not only kills our time but also contributes to environmental pollution and affects human health. With the increase in traffic, it is now becoming increasingly difficult to find a solution for a better traffic control system due to the limitations of roads. This situation demands a better and more efficient traffic management system to take the best output from our roads and drivers. To achieve this, a data-driven traffic control system is needed that can adapt to real traffic flow.

Traffic conditions and safety can be enhanced with the use of Intelligent Transportation Systems (ITS), which combine sensors, upgraded data, and computing technologies. One of the primary characteristics of ITS is its ability to dynamically modify traffic signals to achieve optimal results based on real-time data. Such adaptive and autonomous control mechanisms should be made feasible by Machine Learning (ML) and Deep Learning (DL) techniques, especially Q-Learning and Deep Q-Networks (DQN), which have demonstrated significant potential in recent years.

Usually, all cities use the fixed time signal system that only works statically, as it is pre-programmed. That's why it can not give better output in a scenario that is unpredictable to it. This leads to unnecessary traffic congestion, which could have been avoided with a better and adaptive traffic control system that can learn from real data and give the flexibility to make real-time decisions based on it.

Our reviews mostly focus on the ML and DL based adaptive traffic signal control algorithms to examine how they differ from the usual fixed-time traffic controls in the output. We emphasize the adaptiveness of these algorithms to get a clear idea

of how they can learn from situations and apply them to their uses.

The review is structured as follows: a brief introduction to IoT, Machine Learning, and Deep Learning in general, and then we finally dive deeper into Reinforcement Learning. In Reinforcement Learning, we discuss the different algorithms that have been implemented by researchers.

2.2 Review of Existing Research

2.2.1 IoT

Through interconnected devices, the Internet of Things (IoT) brings new capabilities to traffic systems by managing real-time data at every level of operation. IoT systems in intelligent transportation use sensors with GPS and cameras to collect traffic data that helps control traffic lights and direct vehicle movements. According to [17], smart road systems process traffic data to optimize road usage and reduce traffic jams.In the paper [33], the authors demonstrated IoT systems improve traffic flow by making instant adjustments to reduce traffic bottlenecks. However, this progress faces continuous difficulties. There are three main challenges that exist in this field, protecting user data from unauthorized access, making different smart devices work together and expanding the system to support increasing urban populations.

IoT technology goes beyond traffic control by using prediction models to detect and stop traffic build-up in advance. In, [8] they explored how to use advanced neural network and machine learning models to study future traffic patterns. By examining IoT-generated data sets experts can help traffic managers take preventive actions before congestion happens. These models show good outcomes but they require better methods because they cannot handle changing traffic properly and have missing data. Different Internet of Things technology elements like V2X(Vehicle to everything) allow cars to work with road systems and pedestrians by sharing data easily. Additionally, V2X(Vehicle to everything) improves traffic flow but setting up this technology in busy city areas needs ongoing development to deliver dependable and real-time results.
Simulation tools like Simulation of Urban MObility (SUMO) have been widely used to test this type of IoT-based traffic solutions. To illustrate, [34] compared Q-learning and Deep Q-Network (DQN) as ramp signal controllers with the help of SUMO, demonstrating that real-time data, like in the case of the IoT devices, could help to improve the traffic performance greatly. Likewise, [15] designed the RESCO benchmarking toolkit based on SUMO that allows to experiment with reinforcement learning algorithms in realistic sensor-driven conditions, as described by IoT-like data settings. Based on these studies, it is shown that IoT data streams when integrated with simulation and reinforcement learning can offer potent adaptive traffic management tools.

2.2.2 Machine Learning and Deep Learning

Machine Learning (ML) and Deep Learning (DL) have changed the way of managing traffic systems. It provided many advanced algorithms and methods for predicting traffic, flow of the traffic, optimizing the route, and managing traffic signals. These advanced algorithms and methods help to identify the problems and make decisions in real-time by using datasets. As traffic congestion is a world-wide problem, by integrating ML and DL into traffic systems offers new insights and creative solutions for enhancing traffic management. Which increases the efficiency and safety of life.

Machine Learning (ML) techniques are considered statistical models that are utilized to make classifications and predictions based on the data provided [29]. Machine learning (ML), a branch of Artificial Intelligence focuses on creating prediction algorithms that aren't tailored for a specific task but rather rely on the equitable identification of patterns in large datasets. In Machine Learning there are many algorithims and models. And these are classified mainly into three categories. They are Supervised Learning, Unsupervised Learning, and Reinforcement Learning(Each category has been used in traffic control with varying degrees of success for improving prediction, classification, and optimization tasks.

In [20], the authors tried to address the important issue related to traffic congestion. The number of vehicles is increasing exponentially day by day. As a result, fuel consumption has increased, taking more than usual time for traveling, road accidents have increased, and pollution levels have elevated drastically. Which affected the quality of life of the general population. Traditional methods of traffic control systems failed to solve these problems in real-time. However, to eradicate this, they proposed the FITCCS-VN (Fusion-based Intelligent Traffic Congestion Control System for Vehicular Networks). This system can monitor and control traffic flow efficiently and in real time. The proposed system (FITCCS-VN) collects real-time traffic data from vehicular networks using Internet of Vehicles (IoV) enabled vehicles. Then they processed these datas with machine learning algorithms, especially Support Vector Machine (SVM) and Artificial Neural Network (ANN) to predict the level of traffic congestion. The FITCCS-VN system using these ML techniques demonstrated an accuracy of 95% in predicting traffic congestion, with a miss rate of 5%. Which outperformed all the previous approaches. Additionally, authors added that this system includes a web-based application for drivers, which will help them to know the current traffic flow and navigate through alternative routes. Furthermore, the authors suggested for future research they can use additional data sources, for example weather conditions, any social events. This may improve the accuracy. Additionaly, exploring the use of advanced machine learning techniques, federated learning, Alexnet and deep learning models could provide more accurate and robust traffic predictions.

In the paper [12], "Traffic Prediction for Intelligent Transportation System using Machine Learning," the authors address the challenges of accurately predicting the traffic flow in the face of increasing vehicle numbers and the exponential growth of traffic data. In their study they analyze the large data for the transportation sys-

tem using deep learning techniques, genetic algorithm, machine learning, and soft computing significantly less complicated. Additionally, image processing techniques to analyze and collect real-time traffic data from sources such as GPS, cameras, and sensors. Their approach mainly focuses on identifying traffic congestion hotspots and predicting the traffic flow with Decision Tree (DT), Random Forest (RF), and Support Vector Machine (SVM). After using these algorithms, Random Forest has the highest accuracy, with 91% predicting the traffic congestion and flow patterns. Therefore, the proposed algorithm has solved many problems like large data issues. Furthermore, the huge dimensions of the dataset are reduced, which avoids overfitting of the model[12]. The authors stated that this work can be improved in future through web-based implementation, like in paper [20]. Also, they added that implementing more advanced algorithms can improve their accuracy.

This research [6] is conducted in Melbourne, Victoria, Australia. The authors aimed to improve the traffic prediction at intersections. Accurately predicting is necessary for intelligent transportation systems (ITS) for enhancing mobility and reducing accidents. They have worked with various types of data. For instance, they collected real-time data with sensors at 4598 intersections in Melbourne with 15-minute intervals, a road accident dataset that consists of multiple attributes (id, location, time, date, road number, type, etc.), and roadwork data. All these data were publicly available and collected from the Victorian Government and VicTraffic. After collecting the study employed two learning schemes which are batch learning (Gradient Boosting Regression Tree, Random Forset, and Extreme Gradient Boosting) and online learning (Fast Incremental Model Trees with Drift Detection). Among the batch learning algorithms, Extreme Gradient Boosting (XGBoost) performed best in terms of accuracy. On the other hand Fast Incremental Model Trees with Draft Detection (FIMT-DD) outperformed all batch learning methods. This FIMT-DD can handle dynamic traffic patterns which make the prediction more accurate. In future work, they might explore ensemble models of FIMT-DD for vehicular network.

This [3] study is designed to improve the short-term travel time forecast for freeways using the Gradient Boosting Regression Tree method. The travel time data used in this paper were from INRIX and collected on five TMC segments along I-95 in Maryland. The authors used Gradient boosting regression tree (GBM) as it can handle discrete data. They took different types of parameters for GBM such as, (learning rate, number of trees) for getting better output. Therefore, GBM model was compared with Random Forest and ARIMA models. GBM outperformed both of these models, especially in multi-step-ahead prediction. The sudden change was well captured by the GBM model, which yielded an interpretable model with the most important variables comprising recent travel times and time of day. However, in future, this work can be extended by incorporating spatial correlations and optimizing model parameters for better computational efficiency.

Deep learning, an advanced level of machine learning has served as a milestone in enabling self-learning methods where models learn patterns directly from data. Its ability to process and analyze vast datasets has made it a cornerstone in solving complex simulation-based problems. Among its several uses, deep learning has proven

enormous promise in tackling traffic congestion issues. By leveraging sophisticated neural networks, deep learning has enhanced the performance, applications, and development of intelligent traffic systems (ITS).

By understanding the behavior of traffic congestion and identifying pivotal points within traffic systems, sophisticated deep-learning methods can provide effective solutions to alleviate these issues. One essential step in this process is acquiring accurate and real-time data from the roads. To enhance the quality and utility of the dataset, Time series analysis is crucial for collecting, processing, and interpreting this raw traffic data. Because its concept is straightforward and solely depends on traffic flow time series, time series analysis is quite popular in the actual world. As in paper [13], there are currently three kinds of traffic flow methods based on time series analysis such as model-driven methods, data-driven methods, and combinatory models which are used in inspecting. Traditional approaches to analyzing time series data include methods such as Arima, Seasonal Arima, Markov Chains, Bayesian Networks, Kalman Filters, etc. These methods rely on strict assumptions, such as having a predefined structure, data following a normal distribution, or the time series being stationary (unchanging statistical properties over time). However, real-world traffic data is often nonlinear and unpredictable, which makes these traditional methods less effective for traffic flow prediction. Despite this, some modifications to these models can lead to improved results and allow them to capture the complexity and variability of traffic patterns to make them efficient models. For example, In paper [5] they tried the ARIMA model in combination with Kalman Filter for predicting traffic congestion based on the previous data, and this algorithm this model was used in the four road segments in Beijing which resulted in an effective and reliable traffic state prediction.

The time series basically contains the events that happened over time in the field and it can be assessed to find the patterns and flow in the field, this feature helps to work in predictive models to analyze past events to predict the future in the roads such as peak congestion times, road accidents, unusual accidents, etc. Its data is being used to train with deep learning models such as Long short-term (LSTM) networks, Convolutional Neural Networks (CNN), Backpropagation Neural Networks (BPNN) to advance traffic prediction. Recurrent Neural Networks (RNN) are vastly used for pattern recognition and prediction but it has a significant drawback in that it fails to learn long-term dependencies because of its vanishing gradient which is why it forgets the far past event in its sequences. Thus, Long Short-Term Memory (LSTM) comes up with the solution as it has a memory cell containing three main gates forget gate, an input gate, and an output gate. As the name suggests these gates help to forget, remember, or update the information in the models which is why it can analyze the event that happened long ago. This feature helps them to work with complex data that RNN could not do. For these unique features, this model has gained attention in the field of traffic congestion detection. In paper[13], this LSTM model outperformed ARIMA and BPNN models in predicting traffic accuracies having RMSEs mean LSTM 14.4438, better than ARIMA's 61.1699 and BPNN's 26.8773 shows that LSTMs are capable of capturing higher accuracies given the normal conditions for the memorial functions. In the paper [16], an improved

LSTM-based model integrates the LSTM and BiLSTM (Bi-Directional LSTM) networks that avail the power of forward LSTM in handling sequential data and reverse LSTM in learning long-term dependencies. This combined approach makes the model much more effective in reducing the prediction error by allowing the first layer LSTM to learn and predict the input time series and refining it more through the BiLSTM network. Finally, the model's performance was assessed using real-world traffic data and it showed improved accuracy and stability compared to both standalone LSTM and BiLSTM. Research indicates that the LSTM model performs well in predicting vehicle speed trends on roads and showcasing strong predictive accuracy for periodic and structured data. However, its performance diminishes when dealing with non-periodic and highly discrete data, and capturing irregular patterns becomes challenging. To overcome such cases, we have seen that paper[28] proposed a hybrid deep learning method combining multiple models to improve the accuracy of traffic flow, the new model combining PSO and Bi-LSTM has been developed for short-term traffic prediction, and the PSO-Bi-LSTM is more powerful than PSO - LSTM because Bi-LSTM architecture captures past and future contextual information and is appropriate for modeling a complex temporal sequence such as traffic flow data. Moreover, the Bi-LSTM has less of a problem with vanishing gradients compared to LSTM, where the gradients become very small resulting in the network being inefficient.

A convolutional Neural Network (CNN) is a deep learning model that is designed to effectively analyze gird- like data structures such as videos, and images and is effective in pattern identification like edges, textures, and objects within visual data using convolutional layers in a hierarchical manner for feature processing and extraction. It can be convenient in analyzing spatial patterns like traffic density across roads to optimize routing and congestion. In paper [32], CNN, LSTM, and Gated Recurrent Unit (GRU) were integrated to predict traffic blockages where real-time datasets were taken from Chicago Traffic Tracker (CTT) and roads were transformed into RGB-coded images as input for GRU to detect the blockage parameters. Then, CNN was given the work of extracting the features and redefining the dataset to make it more befitting for LSTM to work with and this preprocess helped to improve the performance of LSTM models with a prediction of 98.87% shows the fruitfulness of the trio GRU, CNN and LSTM for working complex dataset.

Moreover, The attention-based- deep neural network has shown great promise in tackling traffic congestion-related issues in a complex scenario. This mechanism can help to improve the quality to identify patterns and more accurately prediction. In the paper [27], an attention-based hybrid network (ADHN) was proposed combining two Convulational Long Short-Term Memory (ConvLSTM) to catch the better flow in the roads and predict it accurately. The result of this paper shows that this model is promising and can be useful to use with LSTMs. Also, the driver plays a vital role in designing the traffic, it is crucial to understand the behavior of the driver to train the model more effectively. In paper [21], attention based deep neural network was used to improve driver's distraction behavior recognition by using channel and spatial attention mechanisms and the result shows that this model achieved a Top-1 accuracy of 98.48%on datasets of American University in Cairo (AUC) and Hunan

University (HNU) which is more than the existing models.

The combination of time series analysis with deep learning models provides an intelligent foundation for developing robust Intelligent Traffic Systems (ITS). These systems can collect real-time data from sensors, cameras, and GPS devices to actively change traffic signals, recommend alternative routes, or even predict long-term traffic trends that contribute to more efficient and sustainable urban mobility solutions.

Reinforcement Learning

There has been growing research interest in the adaptive traffic signal control (ATSC) as traditional fixed-time or pre-timed controllers are not dynamically appropriate to handle highly dynamic and urban traffic demands. Traditional methods including the fixed time scheme of Webster or coordinated adaptive signal schemes including SCOOT and SCATS are based on periodic or historic traffic data. Practically, though, city traffic is dynamic, and it varies due to accidents and weather as well as day-to-day variations. Reinforcement learning (RL) and deep reinforcement learning (DRL) is also a promising alternative in recent years and can actively learn control policies based on data or simulations. The frameworks considered in the given literature review are directly connected with the research by [34] who compared two frameworks, Q-learning and Deep Q-Networks (DQN) to the traffic signals control in an on-ramp situation. We compare the similar works by [18] [26] and [15] in our discussion, which also use reinforcement learning techniques to traffic lights, but with varying representations of the state space, action space, and algorithm design.

The classical solution of RL application in the traffic signal control was Q-learning. It is an action value function in tabular form, which is updated with the interaction between the agent and the environment. The conceptual simplicity and interpretability of it is its primary strength, which is why it was first applied to traffic lights. Its practical implementation is not without difficulties. However, the state space of an actual traffic intersection is highly large, and includes variables like queue lengths and lane occupancy, phase history and even driver behaviour. The table form is soon unwieldy and the curse of dimensionality is soon evident. To address this, [34] used the Q-learning to control metering with a limited state representation and phase duration actions which were controllable. Although this was a useful starting point it did not work on more complex cases.

The Deep Q-Network (DQN) can overcome these disadvantages by using a deep neural network (DNN) to approximate the Q-function instead of the explicit Q-table. The convergence can also be accomplished more with stabilization methods, like replay buffers and target networks, and hence DQN can run in large and high-dimensional state spaces. Their version of DQN outperformed tabular Q-learning significantly, according to [34] theirs provided a much higher level of performance with an estimated ten percent higher average vehicle speed and a reduction in the number of vehicles occupied in a lane by a factor of three. Their results align with that study by [18] that trained DQN models on both isolated intersections and coordinated intersections. The ability of DQN to deal with complex conditions

as well as its expansion to multi-intersection corridors is pointed out in the study of Park. They find that DQN has a statistically significant reduction in delays and throughput over either tabular reinforcement learning or classical optimization methods.

One of the main design questions in the context of the traffic control reinforcement learning is the definition of state, action, and reward. The compact state definition in [18] was the longest queue length in each approach per signal cycle. This also decreased the state space and speeded up training but at the expense of spatial detail. [34] used a more detailed approach and provided a description of the spatial distribution of vehicles in the form of a discretized lane occupancy matrix. Such representations of a high-resolution allow the agent to anticipate the congestion and also the trends of vehicle accumulation. [26] went one step further to generalize it to continuous state representations that can be used in actor-critic algorithms. By their RESCO benchmarking toolkit, [15] focused on state design flexibility by providing the flexibility of features of lanes (e.g. queue length, arrival rate, waiting times) to conduct comparisons between states fairly under different sensing assumptions.

The design of action space also differs considerably among researches. [18] designed their agent to select one of a limited set of pre-computed signal timing plans to use in the following cycle. This made the cycle-level action space deployment-friendly by ensuring that resulting policies were always related to viable traffic signal plans.[34] instead simulated actions as continuing with the same stage or going to a different one, with more time resolution. This was further extended by their Q-learning implementation by tuning the green phase durations. [26] also experimented with continuous action spaces by using Deep Deterministic Policy Gradient (DDPG), which generated continuous outputs, e.g. green splits or offsets. This allowed agents to coordinate signals within arterial networks, and resulted in the emergent behaviour of a greenwave, where signals coordinately move to allow vehicles to flow through.[15] provide support of all these possibilities in the RESCO toolkit, which is why it could be considered a valuable benchmarking environment.

The design of rewards is very crucial to learned behavior. [18] treated the number of stops and delay into one measure as their reward, which was average stop delay.One of the congestion management tools, [34] introduced the ratio of the mean travel time (reward), whereas [26] embraced more severe congestion as a punishment (squared queue length) (punishment).

In the RESCO toolkit, it is stressed that to make sound conclusions, a variety of reward functions should be tested: delay, pressure, or throughput, etc. Improperly designed rewards may lead to unplanned policies, e.g. to cut back delay at a single intersection against fairness between approaches. Thus, the design of rewards is crucial to Q-learning and DQN.

As it was shown by [18], DQN can be deployed to both coordinated and isolated intersections. They reduced instability with its cycle-level action design and enhanced interpretability, demonstrating that DQN was able to compete with optimization in fixed-time strategies in realistic VISSIM simulations. Their work showed that DQN can be used in practice in real-world traffic systems since the actions it chooses are always possible and reasonable to traffic engineers. According to both studies [18], [34], their findings suggest that DQN is more flexible to real-time traffic compared

to Q-learning or fixed plans.

In addition to DQN, [26] reviewed the actor-critic algorithms, especially DDPG. As DQN can only optimize discrete actions, actor-critic can be used to produce continuous actions, making them highly applicable in large networks where timing is required to be diversified. [26] demonstrated not only the high-performance of DDPG agents in grid networks but also found emergent greenwave coordination, which is among the most sought-after findings in traffic control in arteries. This finding proves the possibility of actor-critic techniques to be equal or even better than strategies developed by humans. DDPG and other such strategies might thus prove more successful than DQN in its extrapolation of local ramp metering to a larger scope of corridor or city-wide traffic control.

A lack of standard benchmarks is one of the persistent problems with RL-based traffic control research. Customized simulation experiments were frequently used in early studies and cross-study comparisons were virtually impossible. This was tackled by [15] who proposed a reinforcement learning traffic signal control benchmark, RESCO. SUMO-based RESCO has realistic city scenarios and RL algorithms implementations. They have evaluated a significant number of algorithms that have worked well in a simple experiment to fail under realistic conditions like partial observability or sensing noise. Interestingly in these benchmarks, decentralized DQN agents frequently matched or outperformed more complicated algorithms, indicating that resilience may be a more important factor than the sophistication of algorithms. Reproducibility and benchmarking, hence, are equally crucial to the design of the algorithm.

Four studies could be compared to draw a number of insights. The difference in the performance of the Q-learning and DQN during the same conditions is reflected by [18], [34] have established that even cycle-level activities have DQN, it may be efficient too. To increase the boundary [26] suggest that scalable and coordinated outcomes can be obtained through continuous control by the application of actor-critic techniques. According to [15], it is better to discuss robustness and reproducibility rather than the performance. In other words, this literature suggests that DQN is a powerful extension of Q-learning, but the state-of-the-art is moving away towards more expressive models and harder test sets.

Although these are good outcomes, there are numerous challenges that have not been addressed. Scalability remains an issue: Q-learning cannot deal with large state spaces, and DQN can cause large networks to be unstable. Actor-critic techniques are more scalable, yet need to be hyperparameter-tuned. Another problem is robustness, which is also illustrated by [15] who indicated that a lot of algorithms do not perform under realistic sensing conditions. The design of rewards is also an issue, as the poorly designed rewards may give rise to unwanted behaviors. The simulation to the real world deployment transferability is yet to be completely comprehended and the interpretation is also a barrier. The majority of DRA policies are black boxes, and thus, they are hard to trust and implement by traffic engineers. However, [26] suggest a promising case in which RL rediscovered greenwave tactics.

To sum up, the reinforcement learning has made tremendous contributions to the improvement of the traffic signal control systems compared to the traditional ones. Q-learning provides a basic yet narrow baseline, and DQN constantly provides higher

adaptability and performance. Actor-critic techniques like DDPG are also more scalable and provide emergent coordination. Tools like RESCO are used in benchmarking of algorithms to make sure they are tested in real world situations. To build a thesis based on [34], it is most suitable to recreate Q-learning and DQN models as baselines and proceed with the research by Cycle-level DQN models, continuous actor-critic models, and robust benchmark environments. It enables the thorough exploration of the reinforcement learning in traffic signal control and the discussion of the factors of scaling, robustness, and interpretability that are considered essential.

2.3 Summary Table

Table 2.1: Summary Table

Ref	**Study Objective**	**Methodology**	**Key Findings/outcome**	**Challenges/Limitations**
[1]	Examined routing protocols leveraging human mobility.	Analysis of mobility patterns, social metrics	More reliable message delivery using social metrics.	Reliance on social metrics can reduce efficiency in less-connected scenarios
[17]	Modeled IoT traffic and performed measurement experiments.	IoT device simulations with traffic metrics.	Demonstrated scalable traffic data collection	Scalability issues as device numbers increase.
[8]	Predicted IoT traffic patterns using neural networks.	Neural network modeling for traffic prediction	Improved forecasting accuracy for proactive routing.	Data quality and real-time adaptability remain challenges.
[4]	Explored decentralized SIoT frameworks for traffic management.	Experimental simulations of SIoT networks.	Reduced congestion through local decision-making.	Ensuring consistent data accuracy and security.
[33]	IoT-enabled traffic control for intelligent transportation systems (ITS).	Real-time data collection combined with adaptive traffic signal algorithms. Tested on dense urban road networks.	Achieved reduced travel times and improved overall traffic throughput by dynamically optimizing traffic signals based on congestion data.	Challenges include handling high-density urban scenarios with unpredictable traffic conditions and ensuring low-latency communication across IoT devices.
[21]	Driver behavior recognition, focusing on distraction detection.	Attention-based deep neural network (ADNet) with channel and spatial attention blocks.	Achieved 98.48% accuracy in behavior recognition, outperforming existing models.	Deep learning models require substantial data for training.
[20]	To develop an intelligent traffic congestion control system for vehicular networks to enhance traffic flow and reduce congestion.	IoT-based vehicle, Machine learning Algorithms (SVM, ANN) FITCCS-VN.	Their proposed model performed better than other algorithms, with an accuracy of 95%.	System scalability for larger, more complex urban areas remains unexplored.
[12]	A machine learning-based system for accurate traffic flow prediction, addressing the complexity and scale of real-world traffic data.	Used supervised learning algorithms, such as DT, RF, and SVM. Collected data with GPS, cameras, and sensors.	91% accuracy Random Forest performed better than the other models.	Lack of practical implementation and testing in real-world traffic environments.
[6]	Improving traffic flow at the intersections.	Data has been collected from Victorian Govt. Batch learning (GBRT, RF, and XGBoost), and online learning (FIMT-DD) has been applied.	XGBoost performed the best. While FIMT-DD proved its effectiveness with real-time data.	As it was tested only for a specific area, which raises concern for whether it can perform on a larger scale or in another area.
[3]	By Using GBM to improve time travel prediction	Collected data from INRIX, vehicles and sensors. GBM was compared with RF and ARIMA for prediction.	GBM outperformed both RF and ARIMA models.	Parameter optimization (learning rate, no. of trees) was complex. Also, it increases computational cost with a big dataset.

Ref	Study Objective	Methodology	Key Findings/outcome	Challenges/Limitations
[13]	Forecast traffic flow using deep learning for more accurate predictions.	LSTM model compared to ARIMA and BPNN models, using OpenITS traffic data.	LSTM outperformed other models in prediction accuracy.	Challenges in dealing with the stochastic nature and length of traffic flow data.
[5]	Predict real-time road traffic states to improve travel time and alleviate congestion.	Combined ARIMA model with Kalman filter for prediction.	The approach demonstrated high prediction accuracy for road traffic states.	Limited by the accuracy of historical data and real-time prediction constraints.
[16]	Enhance short-term traffic flow prediction accuracy using an LSTM-BILSTM approach.	Time series analysis followed by the integration of LSTM and bidirectional LSTM networks.	The method outperforms traditional models in prediction accuracy and stability.	Model complexity and dependence on preprocessing for optimal performance.
[28]	Enhance short-term traffic flow prediction using PSO and Bi-LSTM.	PSO for parameter optimization combined with Bi-LSTM for prediction.	The PSO-Bi-LSTM model outperformed other models (LSTM, ARIMA) in prediction accuracy.	Model complexity and need for careful parameter tuning.
[32]	Predict urban traffic congestion using deep learning.	Combined GRU, CNN, and LSTM models with RGB-coded images for traffic prediction.	Achieved high accuracy in predicting congestion, demonstrating a significant improvement over traditional models.	Model complexity and the need for extensive training data for high accuracy.
[27]	Improve traffic flow prediction using Google Maps data without paid API.	Developed ADHN with ConvLSTM and attention mechanism for spatial-temporal traffic prediction.	ADHN outperformed other models in prediction accuracy.	Reliant on Google Maps data, which might limit generalizability.
[34]	Compare Deep Q-Network (DQN) and Q-learning on-ramp traffic signal control.	planned signal control systems with Q-learning (tabular) and DQN (neural network approximation). Mimicked with SUMO with speed and delay of vehicles as well as occupancy of the lanes as the performance metrics.	Compare Q-learning and Deep DQN saw an improvement in average vehicle speed by the tune of 10% and in lane occupancy by the tune of 30% compared to Q-learning. Both RL approaches were better than no-control baselines. Q-Network (DQN) of on-ramp traffic signal control.	Single ramp case; cannot be used in large-scale urban networks, or between a large number of intersections.
[18]	Research on DQN on isolated and coordinated traffic intersections.	Developed state-defined DQN agents based on queue lengths; tried using microsimulation of single and corridor-level intersections.	DQN minimized the average delay, and enhanced throughput over a fixed-time control; had been demonstrated to work in multi-intersection corridors.	Action space used signal plans calculated a priori; too small to incorporate all the flexibility of adaptive control.

Ref	Study Objective	Methodology	Key Findings/outcome	Challenges/Limitations
[26]	Investigate actor-critic algorithms (ex: DDPG) to control traffic lights.	Deep reinforcement learning applied to grid and arterial networks using an applied deep learning with a continuous action space; experimental emergent behaviors.	In large networks, actor-critic methods were better at coordinating like greenwaves and scaled than DQN.	More hyperparameter sensitive; unstable training when dealing with complicated traffic problems.
[15]	Offer standardized benchmarkin g toolkit (RESCO) of the RL-based traffic signal control.	Developed SUMO-based simulation of realistic city environments; the implementations of Q-learning, DQN, and other algorithms were provided.	Proved that the performance of algorithms is not linear under real world conditions (e.g. sensing noise, partial observability)Said the need of reproducibility and robustness.	There are numerous examples of RL techniques that have performed poorly in real-world application; demonstrates discrepancy between controlled experiments and practical applications.
[35]	Test congestion reduction plans at Gulshan-2 intersection (Dhaka).	Gathered actual road traffic information (volumes, turning, delays, queues).Simulated and analyzed congestion in varying situations.	Dataset contains specifics of traffic features on a very busy intersection; it is important to test the RL-based control frameworks on it.	Information on only one intersection; might not be applicable to other city-wide networks unless further measures are taken.

2.4 Research Gap

There has been a great amount of research in the traffic domain as well as the signal management domain. However, there seems to be some scope for a new study. The existing popular models did not address abnormal patterns such as potholes, irregular pedestrian crossings, or sudden disruptions. Moreover, the models were primarily designed for developed countries, but for underdeveloped countries and their traffic patterns, there has not been much of a study. In Bangladesh, we see reckless lane changing, jaywalking, and bad road conditions that affect the discipline of the traffic management. There has not been much study keeping in mind these obstacles and abnormalities. Specifically, modern adaptive traffic signaling uses machine learning and deep learning models. Pure RL-based controllers may respond slowly due to the exploratory behavior and unstable and non-stationary value updates. This motivates the study of hybrid control strategies to improve reliability in real-world situations.

Chapter 3

Requirements, Impacts and Constraints

This chapter will put the thesis objectives into practical constraints and system requirements into practice. It defines what the simulation platform has to do, how it has to act during experiments, and what it takes to recreate the results. Since the work is driven by the congestion realities in Dhaka, social, environmental, ethical, and economic implications are also touched in the chapter, and it ends with the well-developed project plan and risk management plan. The general goal is to make the proposed controller "CRQL" be fairly compared to strong baselines in both normal and disturbed conditions, and to make the claims to be scientifically plausible, and the implementation to be reproducible.

3.1 Final Specifications and Requirements

3.1.1 Functional Requirements

The system has to recreate a representative signalized urban intersection of Dhaka with the Eclipse SUMO traffic simulator. The simulated network should have realistic representation of lane geometry, signal phase pattern, detector locations, and traffic demand patterns in a way that can lead to a natural occurrence of recurrent congestion and queue formation over the simulation horizon. It is not like copying a certain intersection but is targeted at capturing the major characteristics of operation that one may notice in Dhaka, such as the skewed approach demand and congestion sensitivity to the signal timing.

The control of the traffic lights has to be done online via the TraCI interface. The controller in each control step must be capable of viewing meaningful traffic conditions e.g. queue length, vehicle occupancy or speed proxy, and dynamically make signal phase choices. The switching of phases should not violate fundamental traffic engineering limits such as minimum green time and valid phase change to ensure realistic or even unsafe signal behavior.

The test platform should have the capability of supporting numerous controllers in a single framework such that all controllers are tested on the same platform. These are a fixed-time (FT) controller as a baseline, tabular Q-learning (QL), deep

Q-learning (DQN) and the suggested congestion-aware reinforcement learning controller (CRQL). The architecture should enable controllers to be swapped without any adjustment to the logic of the simulation so that there is fairness and consistency in the assessment.

The system should be able to inject disturbances on a controlled basis to determine how robust the system is in the process of simulation. Two types of anomalies are taken into consideration, potholes that are modeled as localized speed reduction regions that temporarily decrease road capacity, and pedestrian jaywalking that is modeled as intermittent disruption that limits vehicle movement and injects shockwave. These anomalies are programmable in terms of location, severity and duration and are employed in testing the controller behaviour during non ideal and disturbed traffic conditions.

In every instance of running a simulation, the system is required to calculate and document the normal traffic performance parameters. These are average vehicle delay, throughput (vehicles passing at the intersection), queue length at approach and aggregate levels as well as cumulative reward of the learning based controllers. Uniform decision-making of all metrics should be made between controllers and in reproducible form. The data recorded should be able to be easily analyzed and shown in comparative plots and summary statistics without any human intervention.

3.1.2 Non-Functional Requirements

This thesis is based on the principle of **reproducibility**. Statically analogous results should be obtained with tests based on the same settings and random seeds SUMO and Python/Numpy seeds. The system should then record seed selection, configuration parameters and repeated number of runs to be used in reporting.

The system should also be **computationally efficient**. The response time to controller decisions made in individual simulation steps should be short enough to enable real-time or better performance on a typical desktop computer. A latency of under 50 milliseconds per decision step is embraced, and measured and reported latencies are taken in DQN and CRQL.

This is another important non-functional requirement known as **robustness**. The controllers should be stable with fluctuation in traffic demand and injected anomalies, and not to have extreme growth of queues, oscillating switching, or long-lasting starvation of certain approaches. Performance is thus not measured in terms of averages only, but also using **stability and fairness** measures.

Lastly, the system is needed to be **portable and serviceable**. It should be implemented on Windows and Linux platforms and be implemented in a modular architecture that has well-defined boundaries between simulation interaction, controller logic, metric computation and visualization. This framework enables testing, extension and reuse of research in the future.

3.1.3 Resource and Software Requirements

A software stack, which is accessible and reproducible, is used in this thesis.

Simulation environment: Eclipse SUMO is taken as the traffic simulator. The version should be captured in such a way that it can be reproduced, the network artifacts show that the network components were created with SUMO 1.23.1. SUMO-GUI is also applicable in visual validation and debugging whereas command-line execution of SUMO is applicable in repeatable batch experiments.

Programming environment: The Python (version 3.8+) is applied to coordinate the controllers and the experiment.NumPy as the library of numeric operations, Matplotlib as the library of plotting, and TraCI as the library of interacting with SUMO are considered core libraries. In deep reinforcement, TensorFlow 2.x should be utilized. A GPU is not compulsory and only enhances the speed of training; the experimental design would ensure that the DQN architecture is kept lightweight such that the work can also be done with only a CPU environment.

OS: Both windows and Linux are supported as target environments, which is in line with a standard academic laboratory environment.

Hardware: The standard desktop CPU is capable of running SUMO and the controllers under this latency target of the thesis. The validity of the results does not require a dedicated GPU, but it can be used to shorten the time taken in training DQN.

3.2 Societal Impact

The congestion in Dhaka has some quantifiable human outcomes: delays in travel, uncertainty of arrival, increased stress, fuel wastage, and decreased productivity. Adaptive traffic signal control is beneficial to society since the intersections are leverage points: when one problem intersection is improved, the flow at interconnected corridors can be reduced, and spillovers can be decreased. With an average delay decreased and spillover of queues avoided by a controller, commuters benefit by having shorter and predictable travel times, the series of schedules can be held more reliably by the public transportation and emergency movement is less limited by gridlock congestion.

Nonetheless, the social worth of automation is conditioned by the way it will share the waiting-time equally and by the safety of its behavior in mixed traffic conditions. The policy of pure throughput optimization can also favorably treat the most challenging strategies and delay insignificant movements many times over. In heterogeneous cities, such as Dhaka, route choice and lane discipline can be heterogeneous, and these biases can be converted into actual inconvenience, as well as encourage unsafe behavior. Because of this reason, the thesis presents the system as a simulation-based decision-support and research system and not a deployment-ready controller. Practical implementation would involve stakeholder discussion,

on-field tests, safety inspections and cautious implementation with the prevailing traffic management behaviors.

3.3 Environmental Impact

Traffic congestion enhances acceleration due to idling and stop and go acceleration and the amount of time spent in low speed that all contribute to an increase in the fuel consumption and emission. When an adaptive controller is able to minimize time in queues and maximizes throughput, it is able to minimize unnecessary idling and to minimize emissions indirectly. The environmental benefit, in this thesis is discussed as an implication of operational enhancement as opposed to a directly measured outcome due to the need to conduct a comprehensive emission study that would entail explicit emission modeling and extra calibration procedures.

Simulation per se has a relatively low footprint, but deep reinforcement learning training may be overtrained to be compute intensive. This thesis manages the cost of compute by restricting the size of the network, reducing the maximum training time, and constrained execution length and repetitive controlled training. This makes the computation costs affordable but yet lets meaningful comparisons be made.

3.4 Ethical Issues

There are fairness, safety, transparency, and accountability which are influenced in automated signal control and which are more acute in dense urban settings.

Fairness is also applicable since intersection control is by nature providing a limited resource, i. e., green time. When one strategy is constantly preferred, the other minor strategies may suffer from extreme waiting time (starvation), which results in frustration, which may lead to unnecessary crossings or red-light running.Congestion aware overrides of CRQL are designed to eliminate this risk, by limiting the growth of queues, and by preventing long-term abandonment of any strategy, and thus its behavior is predictable compared to the purely exploitative RL rule.

Safety is important because aggressive phase switching results in unrealistic or unsafe patterns particularly when there are disturbances of jaywalking. Safety constraints (minimum green times, limited action sets and stability checks) are therefore regarded as being fundamental in the thesis. The anomaly tests are added to make sure that the performance is not achieved at expense of unstable behavior when subjected to shock conditions.

Black-box deep RL controllers have a problem with **transparency**.CRQL (a conceptualization) is better stated: the RL constituent deals with adaptivity, and the congestion responsive layer is a safety-and-fairness guardrail. This is easier to decode by operators as compared to an all-neural policy.

The issue of **accountability** is not completely applicable to the simulation researches, yet the thesis itself contains the statements that restrain the claims on the simulation results. Implementing the system in the real world would need some

governance: who is the one to maintain the system, who is the one to look at it, how do failures get treated, and what are the liability provisions.

3.5 Standards

This is not a product-certification endeavor, but a work that is based on firmly established principles of traffic engineering so that outcomes are not in vain. Minimum green requirements are imposed on the signal control, valid switching logic is always defined and calculated and experimental settings are recorded to permit replication. It is focused on the methodological rigor and reproducibility and not connected with the compliance to a certain standard of deployment.

3.6 Project Management Plan

There are several steps in this thesis and each step leads to a clear and verifiable outcome before proceeding on to the next one. The project starts with analysis and validation of SUMO network, route, and detector locations, and the resulting intersection should exhibit believable congestion behavior. A fixed-time controller is then installed to give a baseline and ensure that there is correct behavior in metric computation and logging.

When the baseline is balanced, tabular Q-learning will be used where the state is represented as discrete, and learning stability will be assessed. The introduction of the deep Q-learning (DQN) agent is provided following that with thoughtful design of state features, action space, and reward formulation. Simultaneously, the metric logging and visualization pipeline is unified in such a way that all of the controllers may be compared on the same reporting basis.

Anomaly modules of potholes and jaywalking are incorporated and tested with the controllers in place. Controlled degradation experiment is then carried out by testing controllers in terms of pressure and anomalies severity in order to test the degradation of the controllers under stress. CRQL design and tuning only start when the system is able to reliably reproduce baseline and RL results. CRQL is then tested under the nominal and anomaly conditions but primarily average behavior and worst-case behavior is also considered as well as fairness markers.

Lastly, findings are collated in terms of reporting the thesis, providing figures, tables and the project timetable. Figure 3.1: Gantt chart for the project plan and milestones (insert here).erokom kichu ekta hobe tawhid

3.7 Risk Management

Identified risks in this thesis are common to RL-based traffic control studies, and all risks are mitigated in a specific manner that scientific integrity is not violated.

During RL training, oscillating improvements or erratic improvements can be observed or ringing. This is alleviated by limiting the action space, having minimum green times, and constant definition of rewards and repetitions by seed. The CRQL

congestion-sensitive override logic can remove the risks of fairness, especially the starvation of low-volume strategies, and worst-approach indicators could be reported rather than simply using averages. The anomaly models might not reflect all the real world complexities and as such, the anomaly is defined with clear assumptions and the anomaly is as a controlled stress test, but not an all round representation of the Dhaka pedestrian and road-surface behavior. Lastly, it can help in reducing the risk of overclaiming by only stating the limitations clearly and limiting the conclusions to the area of simulation.

3.8 Economic Analysis

Congestion has direct economic impacts in Dhaka in terms of wasted fuel, time loss, decreased reliability of workers and freight, higher vehicle operating costs. The aggregate savings can be achieved due to the fact that traffic volume is high and even slight decrease in average delay may lead to significant savings. Thus, adaptive signal control has an economical potential, particularly when it decreases queue spillback and enhances consistency of movements at the corridor-wide level.

This thesis fails to estimate a complete deployment cost model since it is a simulation-based one. In actual implementation, economic factors would involve sensing infrastructure, compute and communication systems, installation, calibration, and maintenance, and periodic retuning or retraining, in particular, learning-based controllers. The thesis therefore considers the economic discussion as qualitative although it is acknowledged that a quantification of benefits can be done in the future by the measure of delay reductions and fuel/time valuation models with realistic assumptions of infrastructure costs.

Chapter 4

Proposed Methodology

This chapter describes the overall methodology of the study, including simulation setup, controller design, abnormal traffic modeling, and the proposed CRQL framework.

4.1 Workplan

To study the model performance of different traffic light controlling models, we replicate a real life situation in SUMO simulator. We study how popular FT, QL, RL, DQN models perform. We discover which models are more efficient. After that, we add some new variables such as lane changing driving pattern, jaywalking, potholes etc. and see how the models perform. Finally, we propose an adhoc model optimized for our road conditions and pedestrian behaviour by studying different model hyper parameter tuning and through experimentation. The workflow is illustrated in the Figure 4.1

4.1.1 Workflow

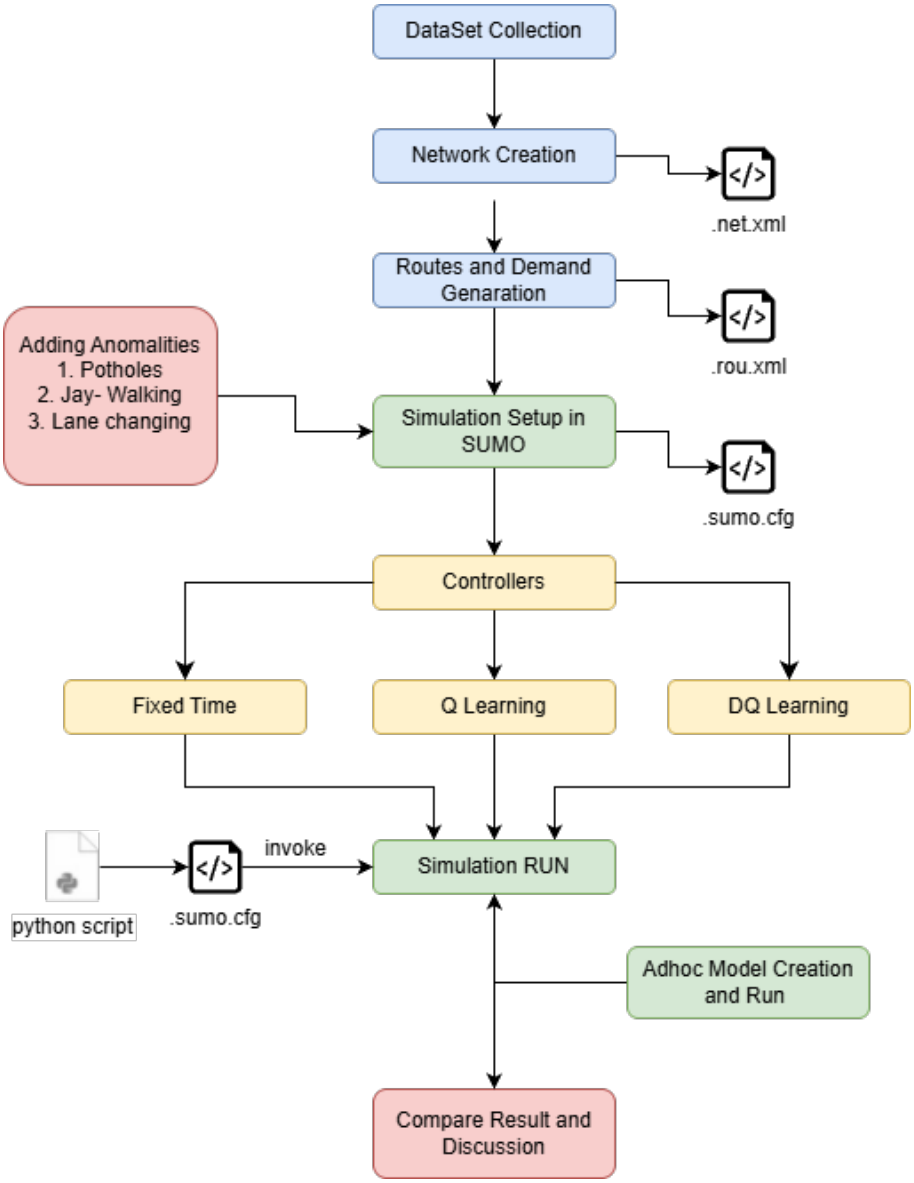


Figure 4.1: WorkFlow

4.2 Design Process or Methodology Overview

The chapter gives the full methodology employed in designing, implementing and evaluating an adaptive system of traffic control lights at the intersection of Gulshan-2 in Dhaka. The research is based on the simulation methodology: we build a realistic SUMO traffic network, model the traffic flows with the real-life observed data, and compare various signal control strategies in normal and abnormal traffic conditions. Signal controllers to be assessed are:

- **Fixed-Time (FT)** – a non-adaptive baseline having a fixed cycle;
- **Tabular Q-Learning (QL)** – a look-up-based reinforcement learning controller;
- **Deep Q-Network (DQN)** – a neural network reinforcement learning controller;
- **Congestion-Responsive Q-Learning (CRQL)** – the suggested hybrid approach that complements Q-learning with a congestion-response model.

Once implementing these controllers we introduce realistic traffic disturbances (potholes and jaywalking incidents in particular) into the simulation with TraCI and measure the performance of each controller with these disruptions. Lastly, we prove that the suggested CRQL approach is more resilient and performs well even in these abnormal scenarios than the other controllers. Figure 4.2 gives a general description of the methodology workflow- starting with gathering data, SUMO network configuration, implementation of the controller, injecting anomaly, and conducting simulation run, recording of measures, and comparing the findings.

Figure 4.2: Methodology workflow diagram

4.3 Preliminary Design and Model Specification

In this section we are specifying the simulation environment and model specification of our study. We would explain the intersection configuration in SUMO, the state space and action space of the controllers, the reward model applied when learning, the equations of each control strategy (FT, QL, DQN, and CRQL), and the modeling of traffic abnormalities that are also stressed-tested by the controllers.

4.3.1 Traffic Signal Control Environment Setup

SUMO Simulation Configuration and Network Generation

The major microscopic traffic simulation platform used in this study is SUMO (Simulation of Urban Mobility), which is integrated via the TraCI API to enable real-time communication between the simulation environment and the algorithm. Because the simulation is based on the real road network that was taken from OpenStreetMap (OSM) data, traffic patterns and geographic restrictions are guaranteed to replicate actual urban intersections.

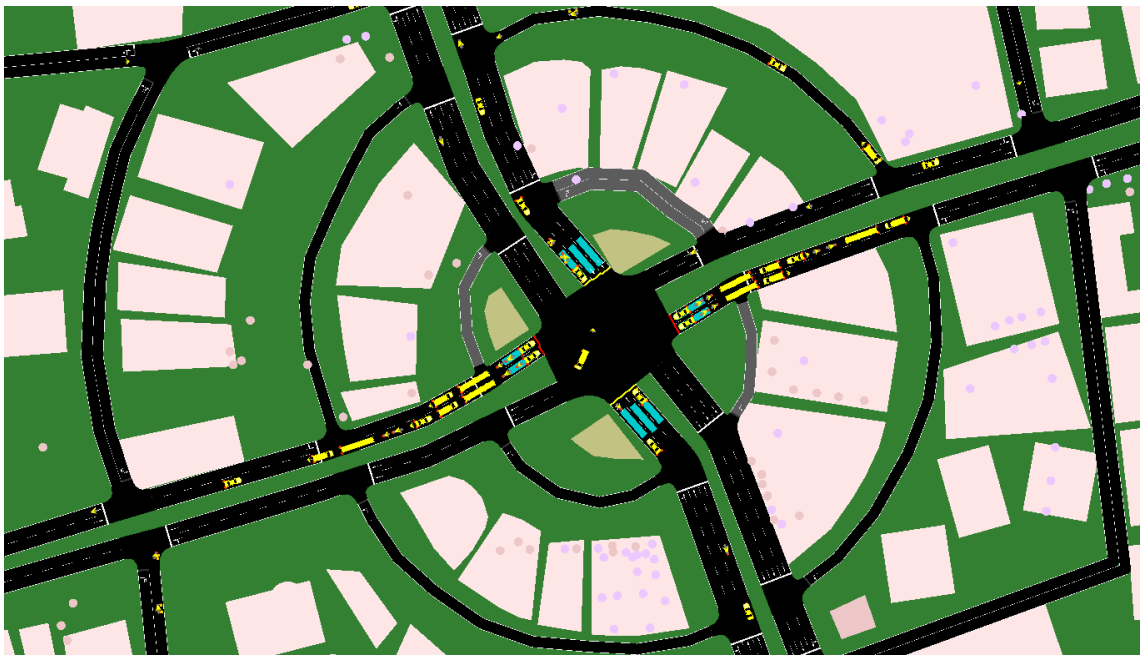


Figure 4.3: Gulshan-2 : SUMO Junction Interface

Network Generation Process: The OSMWebWizard application from SUMO, which obtains authentic geographic data directly from OpenStreetMap(OSM) servers, was used for building the road network. By providing its geographic coordinates, the Gulshan-2 roundabout intersection in Dhaka, Bangladesh, was chosen for this process, and the associated OSM data were transformed into network (.net.xml) and route (.rou.xml) files that are compatible with SUMO. To ensure that the simulation environment effectively replicates actual traffic engineering conditions by precisely preserving real-world road geometries, lane combinations, and intersection layouts, we use GIS(Geographic Information System) along with OSM. OSM does not always fully replicate the the scenarios accurately. We scaled, added or deleted lanes, modified objects and infrastructures as per the real structure of the road as per shown in Figure 4.3.

A time step of 0.1 seconds is used in the simulation to accurately record traffic movements. Also, a 100 ms processing delay is incorporated to allow for practical computational limitations. It provides a realistic and visually precise depiction of the Gulshan-2 intersection area by using a real-world view that displays real road

textures, building outlines, and geographical details using OpenStreetMap (OSM) data.

Junction Infrastructure and Traffic Light Configuration

The OSM import procedure generated the Gulshan-2 roundabout intersection, maintaining its genuine four-approach design with realistic turning patterns and lane configurations. Then, building a junction ensures that the simulated model closely resembles real-world situations by maintaining the real-world intersection's approach angles, road widths, and geometric limitations.

Junction Infrastructure Setup

SUMO automatically detects intersection nodes and creates traffic light systems as needed during the network conversion process. SUMO's junction processing algorithms created the special traffic light identification `"cluster_1575362037_5485881525_98767208_9997687393"` for the Gulshan-2 crossroads. This identification relates to a clustered junction, which is a single, signal-controlled intersection created by aggregating numerous OpenStreetMap(OSM) nodes.

All the traffic movements through the roundabout are controlled by the traffic light system, which employs a four-phase sequential control technique. Each phase has built-in clearance times to prevent conflicts between movements and is designed to safely handle particular traffic flows. The system also imposes a minimum green time of 10 seconds to improve safety by avoiding abrupt phase shifts that could result in hazardous intersection situations.

Traffic Detection System Implementation

We carefully positioned SUMO's E2 (areal) detectors on each of the roundabout's four approaches in order to create a thorough traffic detection system Figure 4.4. In order to guarantee full coverage while maintaining realistic placement, the detectors were also manually introduced to the SUMO network in accordance with regular traffic engineering methods.

With additional detectors positioned where traffic flow is higher, the detectors are arranged to give coverage proportionate to each approach's capacity. Here, by employing SUMO's `getJamLengthVehicle()` function which detects vehicles moving slower than 0.1 m/s rather than simply their presence each E2 detector counts the number of vehicles experiencing congestion and monitors specific lane segments. Consequently, decisions regarding traffic control in this structure are based on actual congestion patterns rather than typical fluctuations in traffic volume. In Table 4.1 additional sumo configuration is mentioned.

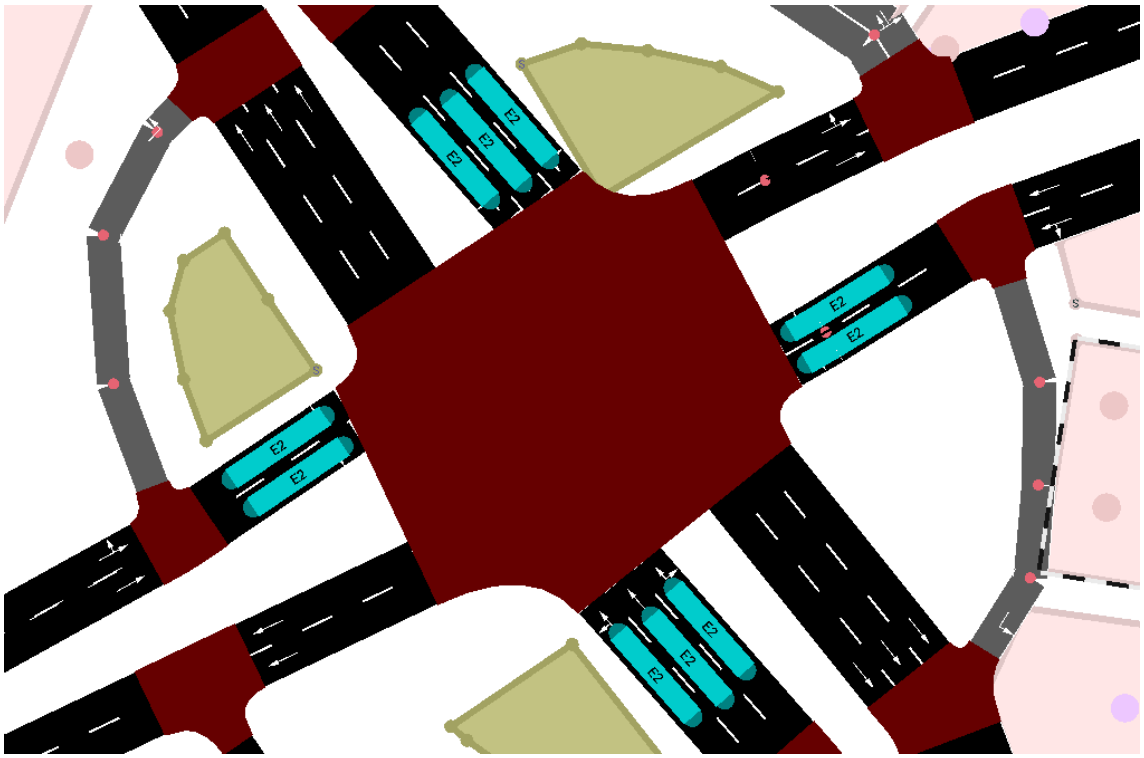


Figure 4.4: Gulshan-2: Netedit Map View

Table 4.1: SUMO Configuration and Setup

Category	Parameter	Value	Description
SUMO Execution	Command	`sumo-gui`	Launches SUMO with graphical user interface
	Configuration File	osm.sumocfg	Main SUMO configuration file
	Step Length	0.10 seconds	Simulation time resolution (0.1s per step)
	Delay	100 ms	Processing delay for visualization
	Lateral Resolution	0	Disabled sublane model
Visualization	GUI Schema	"real world"	Visual representation style
	View	"View #0"	Default camera view
Traffic Light System	Traffic Light ID	"cluster_1575362037_5485881525_98767208_9997687393"	Unique identifier for Gulshan-2 intersection
	Number of Phases	4	Sequential phase control (0→1→2→3→0)
	Minimum Green Time	100 steps (10 seconds)	Safety constraint for phase changes
Traffic Detection	Total Detectors	10 E2 detectors	Areal detectors for traffic monitoring
	Detector Type	E2 (Areal)	Jam-based vehicle counting
	Detection Method	`getJamLengthVehicle()`	Measures congested vehicles
Detector Distribution	"ka_in" approach	[e2_0, e2_1]	2 detectors
	"ma_in" approach	[e2_5, e2_6]	2 detectors
	"gs_in" approach	[e2_2, e2_3, e2_4]	3 detectors
	"gn_in" approach	[e2_7, e2_8, e2_9]	3 detectors
Simulation Parameters	Total Steps	10,000	Complete simulation duration
	Real Time Duration	1,000 seconds	Equivalent real-world time
	Data Recording Frequency	Every 10 steps	Performance monitoring interval
Network Source	Map Origin	OpenStreetMap (OSM)	Real-world Gulshan-2 intersection geometry
	Location	Dhaka, Bangladesh	Authentic urban intersection
	Network Generation	OSMWebWizard	SUMO tool for OSM data conversion
API Interface	Connection Method	TraCI (Python)	Real-time simulation control
	Initialization	`traci.start(Sumo_config)`	Establishes SUMO-Python connection
	Termination	`traci.close()`	Closes simulation connection
Action Control	Action Space	[0, 1]	Binary control decisions
	Action 0	Maintain current phase	No phase change
	Action 1	Advance to next phase	Sequential phase progression
	Safety Override	Minimum green time enforcement	Prevents unsafe rapid switching

4.3.2 Notation and Common Definitions

In order to formalize the problem, we introduce some notations and the way the traffic state has to be represented:

Time is indexed by simulation steps $t = 0, 1, 2, \ldots, T-1$, with the overall number of steps being $T = 10{,}000$ (a sim time of 1000 seconds or about 16.7 minutes, with 0.1s steps).

Each step $= \Delta t = 0.1$ seconds in real-time.

(A) Queue Aggregation: $q_d(t)$, jam- length (vehicles number) of detector d at time t. We calculate summations of the numbers of detectors of each approach to define aggregated queue lengths of each approach. As an example:

$Q_{KA}(t) = q_{e2\_0}(t)+q_{e2\_1}(t)$ in the case of the two KA detectors. $Q_{MA}(t) = q_{e2\_5}(t)+ q_{e2\_6}(t)$ in the MA approach. $Q_{GS}(t) = q_{e2\_2}(t) + q_{e2\_3}(t) + q_{e2\_4}(t)$ using the GS approach. $Q_{GN}(t) = q_{e2_7}(t) + q_{e2_8}(t) + q_{e2_9}(t)$when using the GN approach.

The cumulative length of the queue Equation (4.1) at the intersection at time t is thus the cumulative addition of all the approach queues:

$$Q_{\text{total}}(t) = Q_{KA}(t) + Q_{MA}(t) + Q_{GS}(t) + Q_{GN}(t) \tag{4.1}$$

(B) Phase: The traffic signal phase at time t is denoted by p_t. There are 4 discrete phases, which are $p_t \in \{0, 1, 2, 3\}$.. The different phases are associated with a particular set of green lights at the intersection (e.g., phase 0 may provide green to KA approach, etc., as is fixed in the configuration).

(C) State Space: We assume two types of state representation to various controllers: - In the case of the QL and CRQL controllers (using tabular Q-learning) an aggregated state representation is employed Equation (4.2):

$$s_t = \big(Q_{KA}(t),\ Q_{MA}(t),\ Q_{GS}(t),\ Q_{GN}(t),\ p_t\big) \tag{4.2}$$

This is a 5-tuple, which is the queue lengths per approach and the phase. - In case of the DQN controller (with a neural network), the state of a higher-dimensional full detector Equation (4.3) is employed:

$$s_{\text{DQN}}(t) = \big(q_{e2\_0}(t),\ q_{e2\_1}(t),\ \ldots,\ q_{e2\_9}(t),\ p_t\big) \tag{4.3}$$

but a 11-dimensional vector (10 detector counts and the current phase) it is. This more detailed information helps the neural network to be more precise regarding the distribution of vehicles in a particular lane provided by this richer state.

(D)Action Space: Action space of all the adaptive controllers is binary $A = \{0, 1\}$ at each decision step: – $a_t = 0$ **involves nothing changes to the current phase** (i.e., do not request a phase change). - $a_t = 1$ **involves changing to the next stage** (i.e., request a phase change).

A phase change is however limited to the minimum green constraint above. This is the rule of phase transition: $a_t = 0$: $p_{t+1} = p_t$ (no change of the phase). - In case

$a_t = 1$ and the minimum green time is met: $p_{t+1} = p_t + 1 \ mod \ 4$ (**jump to the next phase of the cycle**). - When$a_t = 1$and the minimum green time is not yet met: $p_{t+1} = p_t$(**the request to switch is not yet taken into account and the phase remains constant**).

The least amount of green constraint is an indicator that we define:

AllowSwitch(t) = {True, if $t - t_{\text{last\_switch}} \geq 100$ steps ($\geq 10s$); False, otherwise,

where t $t_{\text{last\_switch}}$ is the time of the simulation where the last phase change occurred. Thereby, a=1 action will only be implemented when AllowSwitch(t) is True.

(E) Reward Function: Time-varying reward is a function that is used to motivate the reduction of congestion. We have a negative reward [34], Equaion (4.4) that is a negative queue length reward:

$$r_t = -Q_{\text{total}}(t) \tag{4.4}$$

This implies that the controller will be more rewarded on small total queues (less negative) and an extremely negative reward when the queue is high. The controllers are trying to minimize the total time spent in intersection queues, by which they are implicitly trying to maximize cumulative reward.

4.3.3 Fixed-Time Control Algorithm

The Fixed-Time controller is a foundation of non-adaptive controller. It follows a predetermined rotation of phases of known green time duration of each phases which repeats in perpetual cycles irrespective of traffic conditions. This plan makes no use of any state or reward data as above; it is provided just as a point of reference.

Assume that the cycle length is C seconds and is made up of 4 phase green time g_0, g_1, g_2, g_3 (in seconds) of phases 0, 1, 2, 3 respectively. We transform them into step units in the simulation (step length $\Delta = 0.1$ s) $G_k = g_k/\Delta$ (number of steps of green of phase k), $k \in \{0, 1, 2, 3\}$. $C_{\text{steps}} = G_0 + G_1 + G_2 + G_3$ is the total length of the cycle in steps.

We have cycle counter $u(t) = t \bmod C_{\text{steps}}$ which is the distance around the cycle that the current time step has gone. This counter subsequently defines the phase:

$$p(t) = 0 \quad \text{if } 0 \leq u(t) < G_0 \tag{4.5}$$
$$p(t) = 1 \quad \text{if } G_0 \leq u(t) < G_0 + G_1 \tag{4.6}$$
$$p(t) = 2 \quad \text{if } G_0 + G_1 \leq u(t) < G_0 + G_1 + G_2 \tag{4.7}$$
$$p(t) = 3 \quad \text{if } G_0 + G_1 + G_2 \leq u(t) < C_{\text{steps}} \tag{4.8}$$

Equation 4.5 illustrates Phase 0. Equation 4.6 illustrates Phase 1. Equation 4.7 illustrates Phase 2. Equation 4.8 illustrates Phase 3.

The FT controller essentially continues through a fixed series of green times of approach in each approach. It does not learn or adapt according to rewards (however,

to be just in this respect, we also record its reward through the same formula so the results of analysis may be compared). An example of fixed-time phase cycle is presented in Figure 4.5 that has identified the green intervals of each phase in the cycle.

In Algorithm 1, the traffic signal is governed by a fixed-time schedule and is not based on the conditions of the traffic. The initial step is the start of SUMO and the connection to TraCI, in order to control and monitor the intersection during simulation (Line 1). Then, with each step of the simulation from $t = 1$ to T, the current traffic state s_t is monitored using the detectors and the current signal phase (Lines 2–3). The controller then continues to execute the identical fixed-time program (green time and yellow time do not change), regardless of variations in congestion (Line 4). Next, abnormal events such as potholes and jaywalking (when scheduled) are introduced to test the robustness of the controller (Line 5). The simulation advances by one step and performance measures including queue length, delay, and throughput are recorded (Lines 6–7). Finally, upon reaching the horizon T, the simulation terminates and the results are stored for further analysis (Lines 8–9).

Algorithm 1 Fixed-Time (FT) Traffic Signal Control

Input: Simulation horizon T, traffic signal ID $tlsID$, detectors D
Output: Traffic performance metrics

```
1  Start SUMO and establish TraCI connection
2  for t = 1 to T do
3      Observe current traffic state s_t using detectors D
4      Maintain predefined fixed-time signal program
5      Inject pothole and jaywalking anomalies (if scheduled)
6      Execute one simulation step
7      Compute reward r_t = −Q_t
8      Log queue length, delay, and throughput
9  Terminate simulation
10 Save performance results
```

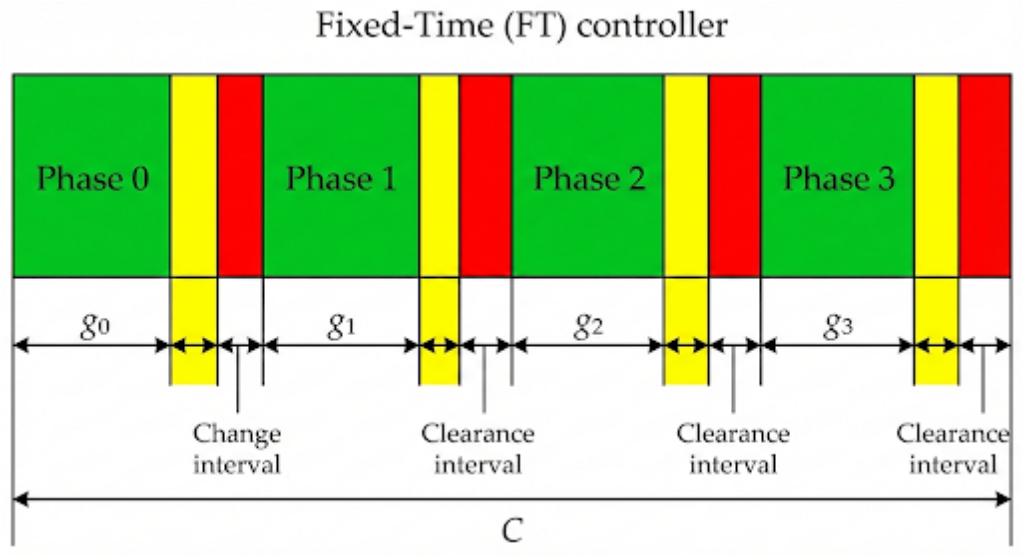


Figure 4.5: Fixed-Time (FT) controller phase timing diagram over one cycle C, showing durations g0,g1,g2,g3 for the four phases.

4.3.4 Tabular Q-Learning (QL) Controller

The Tabular Q-Learning controller is a reinforcement learning agent which optimizes an ideal signaling policy over time by the Q-learning algorithm. It has a table of $Q(s, a)$, which is a table of the state-action pairs and it is updated upon experience.

Policy (ε-greedy): At the decision step, a controller decides on an action based on an ε-greedy policy. - With probability ε, probabilistically sample an action at random between 0 and 1 (exploration). Choose an action with maximum Q-value in the present (exploitation) with the probability $1 - \varepsilon$ (exploitation).

In formula form:

$a(t) = \{\text{random}(0, 1),\ \text{with probability } \varepsilon;\ \arg\max_a Q(s(t), a),\ \text{with probability } 1 - \varepsilon$.

The common range of 0.1 (i.e., 10% of time agent explores by choosing a random action) is seen as typical. In other implementations, ε can also fade away over time to encourage exploitation during learning, although in our implementation ε is held relatively constant (around 0.1) to ensure that it does not reduce exploitation occasionally.

Q-Value Update Rule: After making an action the controller makes a scan of the immediate reward and the next state and the Q-table entry of the state-action pair $(s(t), a(t))$ is updated. According to [34] Q-learning update is Equation 4.9:

$$Q\big(s(t), a(t)\big) \leftarrow Q\big(s(t), a(t)\big) + \alpha \left[r(t) + \gamma \max_{a'} Q\big(s(t+1), a'\big) - Q\big(s(t), a(t)\big)\right] \tag{4.9}$$

where α is the learning rate, γ is the discount on rewards in the future. We set the learning rate $\alpha = 0.1$ and discount factor $\gamma = 0.9$ in our implementation. These

values imply that the controller adaptively changes the Q-values (small α) and takes into account future rewards with a rather heavy weight ($\gamma = 0.9$).

Safety Constraint: The QL controller does not violate the minimum green constraint of the traffic signal. This implies that although the Q-table is indicating that there is a phase switch (action 1), the phase switch will only occur when AllowSwitch(t) is True (i.e., the minimum green time has elapsed). When the agent decides $a(t) = 1$ too soon, it is as though the action is ignored (it does not change phase at that step), though the experience is still learnt.

Figure 4.6 shows a conceptual flowchart of the Q-learning process. each time the agent sees the state, it selects an action (which it may randomly select through ε-greedy) and applies it (with constraints) before seeing the reward and updating its Q-table before proceeding to the next step.

Algorithm 2 describes the Tabular Q-Learning (QL) based traffic signal controller. In this approach, a Q-learning agent is trained to learn the appropriate signal action based on the observed traffic state. Initially, a Q-table $Q(s, a)$ is initialized (typically with zeros) for all state–action pairs (Line 1). SUMO and TraCI are then launched to allow step-by-step interaction between the learning agent and the simulator (Line 2). For each time step from $t = 1$ to T, the agent observes the current traffic state s_t (e.g., queue-based representation) (Line 4). An action a_t is selected using an ϵ-greedy policy, allowing a balance between exploration and exploitation (Line 5). If the selected action requests a phase change and the minimum green constraint is satisfied, the traffic signal phase is updated; otherwise, it remains unchanged (Lines 6–7). The simulator advances by one step, the next state s_{t+1} is observed, and a reward r_t is computed (typically based on negative delay or queue length) (Lines 8–10). Finally, the Q-table is updated using the Bellman update rule, and the learned policy along with performance metrics is stored (Lines 11–13).

Algorithm 2 Tabular Q-Learning (QL) Traffic Signal Control

Input: T, learning rate α, discount factor γ, exploration rate ϵ
Output: Learned Q-table and performance metrics

1 Initialize Q-table $Q(s, a) = 0$
2 Start SUMO and establish TraCI connection
3 **for** $t = 1$ *to* T **do**
4 Observe current state s_t
5 Select action a_t using ϵ-greedy policy
6 **if** a_t *requests phase change* ***and*** *minimum green time is satisfied* **then**
7 Switch traffic signal phase
8 Inject anomalies and advance simulation
9 Observe next state s_{t+1}
10 Compute reward $r_t = -Q_t$
11 Update Q-table using Bellman equation:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right)$$

12 Save learned Q-table and results

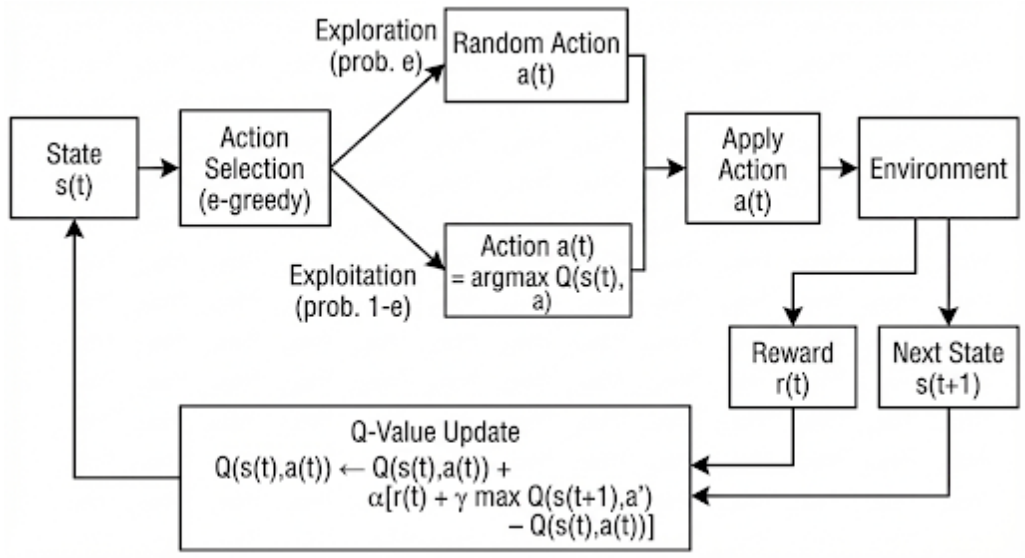


Figure 4.6: Q-Learning controller loop diagram, showing the cycle of state → action (-greedy) → reward → Q-value update.

4.3.5 Deep Q-Network (DQN) Controller

Deep Q-Network controller is a sophisticated RL agent, which employs a neural network function approximator of the Q-values rather than a tabular look-up. This enables it to generalize seen states to unseen states thus enabling it to operate on a greater state space (our state being the complete 11 dimensional state).

The DQN calculates the Q-value of state-action pairs using a neural network $Q_{\text{hat}}(s, a; \theta)$ having parameters to estimate the Q-value of state-action pairs. The network usually accepts the state as an input and gives Q-values of each action (0 and 1). The main aspects of DQN formulation are:

- **The Experience Replay:** The DQN is trained to store the experience (s, a, r, s') in a replay memory and utilize mini-batches of the experience to update itself. This sequencing sample correlates and stabilises training.
- **Target Network:** It is a common practice to use a different target network $Q_{\text{hat}}(s, a; \theta_{\text{target}})$ to compute the target values in order to obtain stability. The parameters θ_{target} network are only updated periodically to that of the main network θ.

Bellman Target: To compute the Q-value of every experience in the loss, it is computed in the following Equation (4.10):

$$y = r(t) + \gamma \max_{a'} Q_{\text{hat}}\big(s(t+1), a'; \theta_{\text{target}}\big) \tag{4.10}$$

In a simplified DQN (when no separate target network is used), $\theta_{\text{target}} = \theta$ (with the same network being used to calculate the target).

Loss Function: The DQN is trained with the mean squared loss with the Q-value prediction of the action taken and the target y. The loss Equation 4.11 at each step of the action $a(t)$ is actually taken from [34]:

$$L(\theta) = \left(y - Q_{\text{hat}}(s(t), a(t); \theta)\right)^2 \tag{4.11}$$

Gradient Update: These parameters of the network are optimized on the basis of stochastic gradient descent (or any other optimizer, such as Adam optimizer) on the loss as follows Equation (4.12):

$$\theta \leftarrow \theta - \eta \nabla_\theta L(\theta) \tag{4.12}$$

where η is the optimizer of the neural network.

In our implementation we take the entire state $s_{DQN}(t)$ as in Section 4.3.2 and we get two Q-values (that of action 0 and that of action 1). The architecture of our Q_{hat} is a standard multi-layer perceptron, and this has 11 input layers, one or more hidden layers (we do an empirical selection of the architecture), and 2 output layers. The DQN also obeys the same minimum green constraint on actions (the network can only recommend switching the environment, although it will not switch unless it is permitted to do so). The DQN architecture and training pipeline, which deals with the interaction with the SUMO environment, experience replay and target network update mechanism, are shown in Figure 4.7.

Algorithm 3 presents the Deep Q-Learning (DQL/DQN) based traffic signal controller, where a neural network replaces the tabular Q-table to handle larger or continuous state spaces. Initially, a deep Q-network $Q_\theta(s, a)$ is initialized with random parameters θ (Line 1). SUMO and TraCI are then started to enable interaction between the network-based agent and the simulated intersection (Line 2). At each simulation step from $t = 1$ to T (Line 3), the agent constructs a numerical state vector s_t using traffic obser- vations such as signal phase, lane queues, and waiting times (Line 4). An action a_t is selected using an ϵ-greedy strategy to balance exploration and exploitation (Line 5). If the selected action triggers a phase change and the minimum green time constraint is satisfied, the signal phase is updated (Lines 6–7).

Algorithm 3 Deep Q-Learning (DQL) Traffic Signal Control

Input: T, neural network Q_θ, α, γ, ϵ

Output: Trained network parameters θ and metrics

Initialize neural network parameters θ

Start SUMO and establish TraCI connection

for $t = 1$ *to* T **do**

  Construct state vector s_t

  Select action a_t using ϵ-greedy policy

  if a_t *requests phase change* ***and*** *minimum green time is satisfied* **then**

    Update traffic signal phase

  Inject anomalies and execute simulation step

  Observe next state s_{t+1}

  Compute reward $r_t = -Q_t$

  Compute target value:

$$y_t = r_t + \gamma \max_a Q_\theta(s_{t+1}, a)$$

  Update network parameters θ by minimizing:

$$(y_t - Q_\theta(s_t, a_t))^2$$

  Log performance metrics

Save trained model and results

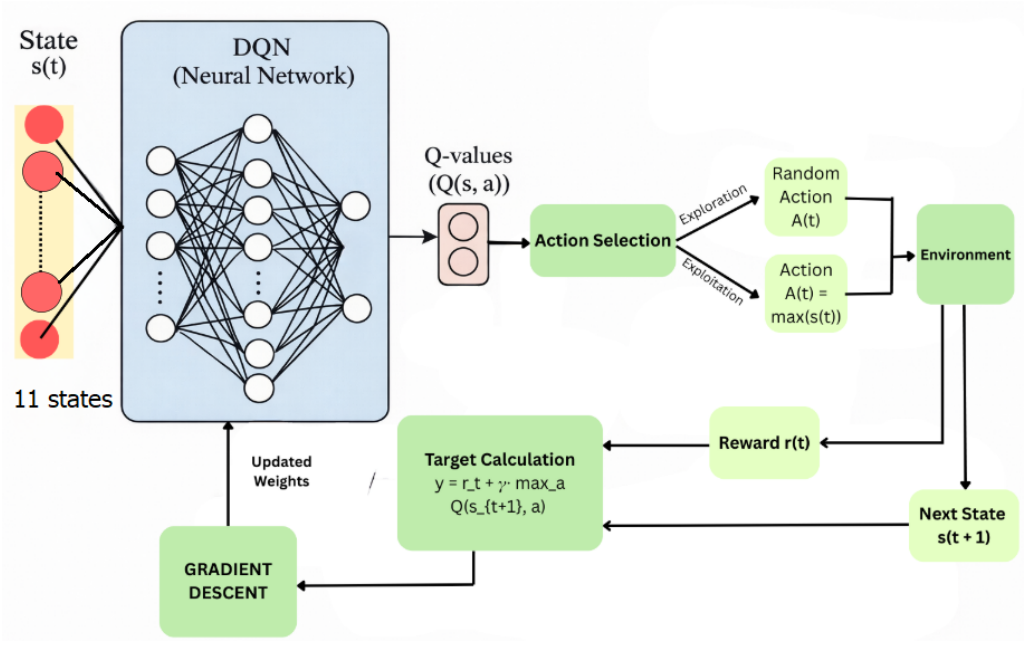


Figure 4.7: Deep Q-Network (DQN) architecture and training pipeline, showing how state inputs are processed by the neural network to output Q-values, and how training uses experience replay and periodic target network updates.

4.3.6 Abnormality Modeling in SUMO (Potholes and Jaywalking)

In order to measure the robustness of controllers, we apply two categories of anomalies to the simulation traffic: **potholes** and **jaywalking**. These irregularities are supposed to replicate real life anomalies that are likely to occur in Dhaka traffic that can lead to non-recurring congestion. These happenings in SUMO are modeled and manipulated through TraCI when running. The anomalies transiently affect traffic flow, which means that the controllers were not specifically trained to handle, and it tests the flexibility of each control strategy.

Pothole Modeling

Purpose: To imitate the destruction of roads (potholes) that compel vehicles to slow down on a specific part of the lanes. This produces artificial bottlenecks and queues in the form of shockwaves in front of the pothole.

Modeling Style: A pothole in the simulation is simulated by indicating a specific portion of a lane as a slowdown zone. Once a car enters this section, a system forcibly slows the car to create the illusion of going over a pothole.

- Let v denote a vehicle. Where $lane(v, t)$ is the lane that vehicle v is in at time t, and $pos(v, t)$ is the longitudinal position of v on that lane (meters along the lane).
- We also assign identifiers on the lanes we label as **pothole lanes** and on each of these lanes set aside a range of position (a start position and end position) which forms the **zone of a pothole**. As an illustration, the pothole zone may be identified as a 10 meters stretch on an approach lane.
- We set a low speed limit $v_{\text{pothole}} = 2.0$ m/s (about 7.2 km/h) to which automobiles are required to reduce their speed when passing through the pothole region. This speed is very low compared to normal speed limits and it resembles a driver who has rolled over a pothole cautiously.

We have an indicator function of being in a pothole zone of any vehicle v:

$$I_{\text{pothole}}(v,t) = \begin{cases} 1, & \text{if lane}(v,t) \text{ is marked with a pothole} \\ & \text{and pos}(v,t) \text{ lies within the pothole segment,} \\ 0, & \text{otherwise.} \end{cases}$$

The TraCI control logic verifies such a condition at every step of the simulation process on each vehicle on the corresponding lanes. In the case of $I_{\text{pothole}}(v, t) = 1$ we use setSpeed(v, v_{pothole}) by means of TraCI, where the vehicle is directed to 2.0 m/s.

The vehicle is brought back to its normal speed once it has left the pothole area (i.e. no longer on condition) by setting setSpeed(v, -1) (which instructs SUMO to continue its default speed control of that vehicle).

Visualization: To make the simulation visualization clear, cars that are currently slowed down by a pothole are displayed in a certain color (e.g. becoming **red** on

hitting the pothole). When a vehicle merely passes the pothole and accelerates away (after having been released on a speed limit), it is made **green** briefly to show recovery. In the default color (e.g., **yellow**), vehicles are seen in the normal state (no anomaly regarding it). Figure 4.8 is a shot of the simulation that shows a noticeable pothole section and a vehicle on the pothole area indicated by red color.

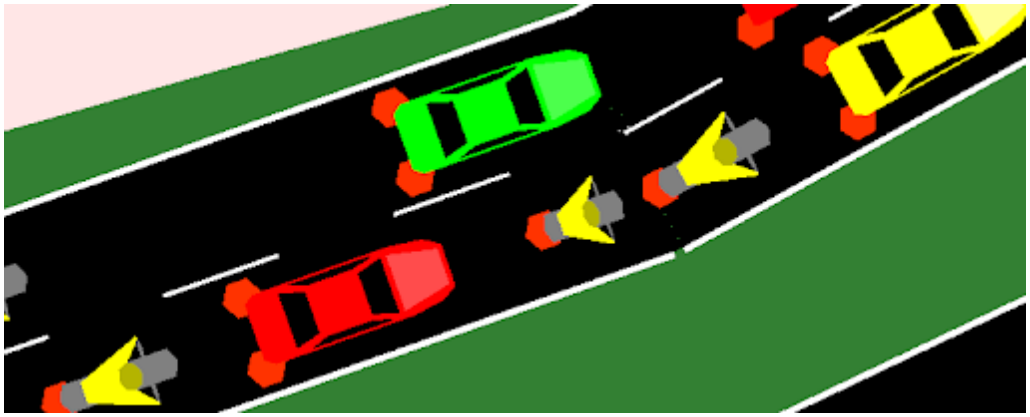


Figure 4.8: SUMO simulation screenshot showing the pothole zone on a lane, with a vehicle slowed down and highlighted in red while crossing the pothole, and a recovering vehicle in green.

Jaywalking Modeling

Purpose: Model the random events of pedestrian crossing (jaywalking) that make vehicles temporarily slow down or yield to reflect the common interruptive activities experienced in the Dhaka traffic where pedestrians suddenly cross the road without warning.

Modeling Style: Jaywalking activities are simulated as sluggishness in stochastic fashion on some lanes (usually the lanes on major roads where people may be seen running over). This is done by randomly choosing such times when an invisible pedestrian causes cars to slow down.

- We simulate small probability p_j of a jaywalking event at time step j on a particular lane, which, in the case that the lane is open, is open. The $p_j = 0.002$ per step (0.2 percentage per step, which is an average of one event per 500 steps, or one event per 50 seconds on a single lane, but it is random).
- When a jaywalking incident has occurred on a lane we put a slow speed on a brief period on the lane vehicles. As the duration of slowdown effect, we use $\tau = 30$ simulation steps, which corresponds to 3 seconds.
- Suppose that v is a vehicle on a jaywalk prone road. We have **jaywalk timer** in our cars (activated to 0). Each of these steps, where such vehicle has a jaywalk timer of 0 (i.e. is not currently slowed by some earlier event), and its random draw is less than p_j results in a new jaywalking event to occur to that vehicle: we set $jaywalktimer(v) = \tau$ (i.e. this vehicle is slowed down in the next 3 s of the simulation).
- A vehicle with $jaywalktimer(v) > 0$ will have its speed reduced to $v_{\text{jaywalk}} = 2.0$ m/s (the same slowdown speed as in the pothole case, assuming that a

pedestrian triggers the same slowdown to $\sim$ 7 km/h): Each time we take a step we reduce the jaywalk timer of the vehicle by 1. When the timer reaches 0 we put the vehicle back on regular speed (set to -1).

This process implies that randomly, there will be a slowing down of certain vehicles to 3 seconds as though it is giving way to a pedestrian crossing the road. This is a stochastic disturbance in traffic flow as the events are independent of each vehicle/lane.

Visualization: The cars that are slowed down because of a jaywalking incident can be indicated in a special color (such as **blue**) in the simulation display. This enables us to see the difference between them and cars that are slowed down by potholes or regular traffic. In Figure 4.9, the simulation screenshot depicts that an incident of jaywalking has happened and an affected vehicle has been denoted in blue and it is moving slowly throughout the entire incident.

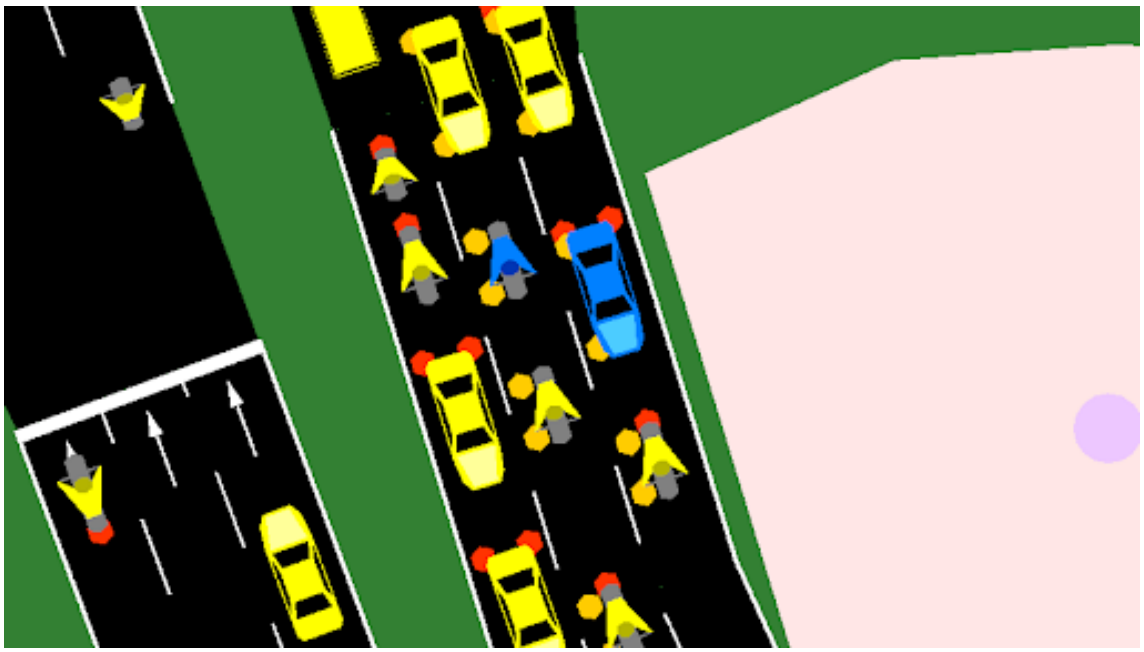


Figure 4.9: Simulation screenshot illustrating a jaywalking-induced slowdown. A vehicle on the main road is colored blue, indicating that a pedestrian crossing event temporarily slows it.

Impact of Anomalies on Controller Performance

The creation of potholes and jaywalking brings about non-recurring congestion thereby interrupting the usual traffic patterns. Such upheavals have a number of impacts on traffic measures -
Shockwaves increase the rate of queues: Sudden slowdowns generate shockwaves and, therefore, queues can form quickly behind the influence zone.
Increased average delays: These interruptions increase the waiting time of vehicles and increase the delay.
A shift in the variance of state transitions: the state of traffic (queues, delays) becomes more unpredictable and changeable, as the randomness introduces variations in it.

Old fashioned controllers are victims of such conditions. The Fixed-Time controller is the most susceptible as it has no opportunity to adjust to any demand changes or disruption. It will still run its programmed cycle potentially giving green time in a non-optimal manner as a queue due to an anomaly continue to increase. Controllers based on RL, such as QL and DQN which learned a policy in normal conditions, can even degrade due to the fact that the environment with anomalies no longer corresponds to the training scenario (the policy may not understand how to act in the presence of, e.g. a sudden stop in a single lane, caused by a pedestrian). Put another way, the existence of anomalies is a breach of the stationarity assumptions of the learned policy - the controller has been put in a new situation that it has not experienced previously, and its learned Q-values may no longer be relevant to it.

This scenario contributes to the necessity of a **more effective control strategy**, and this is the reason why we suggest CRQL. The Congestion-Responsive Q-Learning approach incorporates a specially crafted override to identify instances where congestion is becoming abnormal (as would occur in the event of such an incident) and temporally changes the control behavior in order to reduce the congestion. The second section describes the CRQL methodology that tries to keep the performance unchanged even when the environment is stochastic or adversarial because of anomalies.

4.3.7 Proposed CRQL (Congestion-Responsive Q-Learning) – Full Implementation

The presented Congestion-Responsive Q-Learning (CRQL) controller is an intermediate, which is a hybridization of the tabular Q-Learning controller (Section 4.2.4) that is complemented by a dynamic congestion detection and response mechanism. Basically, CRQL is a normal Q-learning agent in regular situations, but it continuously uses the traffic conditions to identify instances of abnormal accumulation of congestion. In the case of a serious congestion trigger being detected, CRQL temporarily replaces the learned policy with a **rule-based** approach which causes all traffic to proceed directly through the intersection (no turns allowed) temporarily in order to clear the queue. On the congestion removal, it clears the override and returns to normal operation of q-learning. In the override, CRQL continues to learn (updating its Q-table with the experiences), however, the actions that it executes are limited by the override logic. The sub-sections below describe the elements of CRQL, its parameters, mathematical formulation of its triggers of congestion and how it works stepwise. The flow chart of CRQL can be observed in the Figure 4.10.

CRQL Core Parameters and Settings

CRQL contains numerous parameters as compared to the conventional Q-Learning controller, but it also introduces new parameters allowing to observe the congestion and control the override behavior. This is because some of the key parameters that we used in our implementation (value assumed) are listed below:

Learning parameters:

- Learning rate $\alpha = 0.1$ (same as QL).

- Discount factor $\gamma = 0.9$ (same as QL).
- Exploration rate $\varepsilon = 0.1$ (where it has a probability of decay but in most cases it is 0.1).
- minimum greenhold MIN_GREEN_STEPS = 100 steps (10-second just as above).
- Steps of simulation TOTAL_STEPS = 10000 / episode (as indicated, a simulated time of 16.7 minutes).

Monitoring of congestion environment:

- moving average window size of Queue QUEUE_WINDOW = 200 steps (i.e. we are looking back approximately the last 20 seconds of data when we are calculating a queue based trigger).
- 200 steps of calculating delays is the Delay Moving window(DELAY_WINDOW).
- Base queue BASE_QUEUE_TRIGGER = 10 vehicles. (Minimum queue length to make an event abnormal - we require at least 10 vehicles waiting on the light traffic as the base).
- Base delay BASE_DELAY_TRIGGER = 5.0 seconds. (Minimum average threshold delay - even in case of low variation, there may be a delay of above 5s which would cause response).
- Multiplier of standard deviation of trigger TRIGGER_STD_MULT = 1.0. The weight on standard deviation on dynamic threshold is: (Weight on standard deviation dynamic threshold; 1.0 means trigger mean + 1 std dev).
- threshold of queue growth ratio QUEUE_GROWTH_RATIO = 1.05 (i.e. 5 percent increase in queue in one step is a significant increase).
- CLEAR_QUEUE_RATIO = 0.85. (Congestion must be 85 percent of trigger threshold, and then it is clear enough to switch of override)
- Minimum override hold time RESTRICTION_HOLD_STEPS = 50 steps (once it is enabled, the straight only mode stays on at least 5 seconds).
- Time spent rerouting during override REROUTE_INTERVAL = 10 steps (vehicles rerouted straight in 1-second mode when they are under override).

Network configuration:

- Network file: osm.net.xml.gz (SUMO network file of the area, that contains road network files as well as the definition of road network traffic lights of Gulshan-2).
- Traffic light controller ID:
  TLSID = Gulshan2_trafficlight_1568815259876720899768739 93. This is the name of the object of the network file of the Gulshan-2 intersection which is the traffic light. CRQL relies with the right traffic light with TraCI using this ID.

These parameters are obtained based on domain knowledge as well as certain empirical tuning. The moving windows sizes (200 steps) control the response rate of the

controller to the changes (20 seconds provides a reasonable responsiveness and noise reduction). The trigger thresholds (base values and multipliers) were determined on the basis of the normal queue lengths and under normal conditions - any surpassing of mean+1 equation in a queue or delay is considered to be abnormal congestion. The override hold and clear ratios ensure that when activated it does not switch on and off again wildly; the system remains in the override state a short and attenuated minimum time and only switches off on condition that the situation has significantly changed.

CRQL Mathematical Components

CRQL has its congestion detection logic that is built on few calculated metrics depending on the latest traffic. Here we put in writing the algorithm of working out the dynamic thresholds.

(A) Dynamic Queue Threshold: CRQL has a moving perspective of most recent aggregate queue lengths is Equation (4.13):

$$W_Q(t) = \{Q_{\text{total}}(t-199),\, Q_{\text{total}}(t-198),\, \ldots,\, Q_{\text{total}}(t)\} \tag{4.13}$$

the sum of the queue values at the last 200 steps (in the case $t < 199$, all the available would be used, till t). $\mu_Q(t)$ is the mean of such a window and $\sigma_Q(t)$ is the standard deviation.

Dynamic-based queue trigger threshold $Q_{\text{trigger}}(t)$ is given as Equation (4.14):

$$\begin{aligned} Q_{\text{trigger}}(t) = \max\Big(&\text{BASE\_QUEUE\_TRIGGER},\\ &\mu_Q(t) + \text{TRIGGER\_STD\_MULT}\,\sigma_Q(t)\Big) \end{aligned} \tag{4.14}$$

In our case, BASE_QUEUE_TRIGGER = 10 and TRIGGER_STD_MULT = 1.0, so effectively Equation (4.15):

$$Q_{\text{trigger}}(t) = \max(10,\ \mu_Q(t) + \sigma_Q(t)) \tag{4.15}$$

This means that the minimum level of the queue is 10 vehicles though in the event where the most recent mean and a standard deviation are above 10, then this figure is used. It simply states that in case of the abnormally large queue current compared to the recent normal queue (over the mean of the last 20 s) it is the sign of an abnormal congestion accumulation.

(B) Queue Growth Ratio: We evaluate the process of the queue development per step as well. The **growth ratio** is given as Equation (4.16) :

$$\text{growth\_ratio}(t) = \frac{Q_{\text{total}}(t)}{\max(Q_{\text{total}}(t-1),\, 1)} \tag{4.16}$$

(The present total queue divided by the total queue in the last step; we take the maximum of the two to ensure that we do not divide by a zero value). A ratio 1.0

would indicate no growth, 1.05 would mean growth of 5 percent in one step, etc. We set a condition that in Equation (4.17) if:

$$\text{growth\_ratio}(t) \geq 1.05 \tag{4.17}$$

then queue is growing at very rapid rate (a growth of 5 percent in 0.1 seconds, indicates a steep rate of growth, which is likely to be a shockwave brought about by an anomaly). This is one of the trigger criteria of override (this is used alongside the threshold above).

(C) Dynamic Delay Threshold: Just as queue, CRQL is a delay measure that is monitored in the form of its average value. V_t denotes the vehicles in the network at time t and $wait_{v,t}$ is the waiting time (in seconds) of vehicle v at time t as provided by SUMO (typically this is the time that the vehicle has been slackened or stalled). We compute Equation (4.18):

$$W_{\text{total}}(t) = \sum_{v \in V(t)} \text{wait}(v, t) \tag{4.18}$$

Time c. total waiting time of all vehicles at time t. **Mean of the delays per vehicles** Equation (4.19):

$$\text{avg\_delay}(t) = \begin{cases} \dfrac{W_{\text{total}}(t)}{|V(t)|}, & |V(t)| > 0, \\ 0, & |V(t)| = 0 \end{cases} \tag{4.19}$$

and where there might be none of the vehicles there we shall say that the mean delay is 0 at the time.

The following Equation (4.20) is our window of average delays:

$$W_D(t) = \big\{\text{avg\_delay}(t - 199), \ldots, \text{avg\_delay}(t)\big\} \tag{4.20}$$

The window $W_D(t)$ is of length 200. Let $\mu_D(t)$ be the mean and $\sigma_D(t)$ be the standard deviation of the data in $W_D(t)$. The **dynamic delay trigger threshold** is thus given by Equation (4.21):

$$\begin{aligned} D_{\text{trigger}}(t) = \max \Big(&\text{BASE\_DELAY\_TRIGGER}, \\ &\mu_D(t) + \text{TRIGGER\_STD\_MULT}\, \sigma_D(t)\Big) \end{aligned} \tag{4.21}$$

With our values, `BASE_DELAY_TRIGGER`=5.0 s and `TRIGGER_STD_MULT`=1.0, this simplifies to Equation (4.22):

$$D_{\text{trigger}}(t) = \max(5.0,\ \mu_D(t) + \sigma_D(t)) \tag{4.22}$$

The very nature of the system presupposes that the average delays would be more likely to average to a certain point; when the delay is already above the recent average by a standard deviation (even by more than 5s absolute in the instance that the normal delays were very low), it is a pointer of unusual congestion.

CRQL Override Trigger Logic

CRQL decides on the time of activation of the congestion override (straight-only mode) by the above measures. We present two fixed conditions, which are checked at each time step: - **Queue condition:** In Equation (4.23) $Cond_{queue}(t)$ is the state where the total queue exceeds the dynamic threshold and the queue is growing fast:

$$\text{Cond}_{\text{queue}}(t) : \big(Q_{\text{total}}(t) \geq Q_{\text{trigger}}(t)\big) \wedge \big(\text{growth\_ratio}(t) \geq 1.05\big) \tag{4.23}$$

That is, it is not only the high queue compared to the recent history, but also the queue is spiking up. - **Delay condition** $Cond_{delay}(t)$ is true when Equation (4.24):

$$\text{avg\_delay}(t) \geq D_{\text{trigger}}(t) \tag{4.24}$$

This means that the mean vehicle hold-up is higher than the normal values in the recent history (or more than the base level).

For this, an override (straight-only mode) flag denoted by a flag of type bool named straight_only_active is included, and is set to either straight only mode or straight only mode off. At first, straight only active = False. The enablement of the override rule is: - in the event of straight_only_active being False and $Cond_{queue}(t)$ or $Cond_{delay}(t)$ being true, then the override is to be activated. When Active: straight_only_active = True. - To specify a restriction_release_step, as $t + RESTRICTION\_HOLD\_STEPS$ (to enact the minimum hold time; e.g. when issued at time t, will not be surrendered until it is time $t + 50$). - Assuming the activation_threshold is the current queue trigger = $Q_{trigger}(t)$ value (a value that will be utilized to define when the queue has cleared accordingly in order to turn off).

In summary, the override is enabled on the occurrence of a severe abnormal congestion (such as the build up of a queue or as a result of excessive delays), and not on the same occurrence - it does not respond unless one of the prior overrides has been cleared before the current one is activated.

CRQL Override Release Logic

After the straight-only override has been turned on, CRQL keeps checking whether it should be switched off. The override must at least continue the fixed hold period and beyond that until congestion is reduced. We shall set threshold values to determine when congestion is clear:

$Q_{clear} = CLEAR\_QUEUE\_RATIO \times activation\_threshold$. This is a queue length alarm (85 percent of the amount which caused the override). When the current total queue $Q_{total}(t)$ reaches or less than Q_{clear}, it denotes that the queues have reduced considerably to the peak. -

$D_{clear} = 0.85 \times D_{trigger}(t)$. It is a delay threshold (we take 85 percent of the existing dynamic delay trigger, or corresponding concept using delay at activation in any case) that shows that mean delays are less.

The **release condition** to drop the override is: - In case of straight only active = True, and the present time t = or exceeds the restriction_release_step (i.e., the minimum hold time has elapsed), and $Q_{total}(t) \leq Q_{clear}$ and $avg\_delay(t) \leq D_{clear}$, then we drop the override by setting straight_only_active = False.

This reasoning offers hysteresis: the override does not respond to the first signal of an improvement, but only to the fact that the conditions are not only improved, but have become normal (queues to 85 percent of trigger level, delays to the same) before responding. It avoids oscillation in which the system may switch off the override only to realize that there is suddenly congestion getting back to its previous level.

Straight-Only Rerouting Mechanism

The fundamental operation of the congestion override is to create a temporary and temporary blockage over the intersection to permit only straight-through traffic. The reasoning is that when there is congestion, turning vehicles tend to cause more conflict as well as discharge flow (e.g., left turns waiting to find gaps, etc.), whereas when all of the approaches run straight, the intersection will effectively act as a cross-road with two directions of flow allowed to co-exist and thus maximize throughput to clear queues in the shortest possible time. In order to apply this in SUMO with the help of TraCI, CRQL can be used to do dynamic rerouting of vehicles:

Pre-computation: Once the CRQL has been launched, the network topology (read in based on the SUMO net file) is read to determine straight-through links at the intersection. There is an incoming approach edge informing each it what outgoing edge is straight Equation (4.25) yields a mapping:

$$\text{straight\_map}[edge_{\text{in}}] = edge_{\text{out}}(\text{straight}) \tag{4.25}$$

for each approach. We also make a list of all approach edges (incoming edges) that are controlled by the traffic light (they are the edges where the vehicles will be taken into account in case of rerouting).

Reroute on override: When straight_only_active = True, the rerouting mechanism is engaged at a fixed interval (every REROUTE_INTERVAL steps which is every 10 steps or 1 second in our case): - At each such an interval, on each approach edge in the list, the set of vehicles on that approach is retrieved (with the TraCI possibility of accessing vehicles by approach edge). - In case of every vehicle on that edge which is not yet rerouted (so it is not rerouted repeatedly). - Store its initial planned route or destination (which can then be reinstated later). - Determine that straight edge of such an approach with straight_map. - changeTarget with TraCI so that the next edge of the vehicle is the straight-going one at that intersection (that is, to make the vehicle cross the intersection and not loop around it). - Monitor this vehicle by making a mark in a data structure, indicating that it was rerouted as a result of override, to have the vehicle remember its original route.

This operation actually blocks turning operations. automobiles that would have turned to the left or right are now directed straight. Already straight vehicles are not affected (except, that is, by being recorded as having been processed).

Restore on release: After restoration release is where when the override is re-enabled (straight_only_active goes False), CRQL executes a restore operation on all the vehicles that were stored as rerouted: - For each such vehicle that remains in the simulation (it may have left the network already), restore its original target/destination with TraCI (in the case that the vehicle is still being rerouted it effectively puts it on its intended path beyond the intersection). - Removal vehicle off the detour list.

This makes sure that cars take the intended routes after the congestion is reduced and the intervention does not create permanent rerouting off destinations. The override is a local temporary one - it is cleared and normal routing is restored.

It is noted that in the override, the signal continues to run in the normal phase cycle or in the Q-learning decisions, however, as all the vehicles are held straight the only movements effectively receiving non-zero flows are the straight movements. This strategy simulates a phase or an **all-pedestrian phase** of the traffic management in which you temporarily give backlog clearance in one direction.

As shown in Figure 4.8, the CRQL control architecture diagram below demonstrates the integration of the Q-Learning agent with the congestion monitoring and straight-only rerouting override. The figure brings out the regular RL loop and the simultaneous congestion detection which has the capability to override the actions of the agent by forcing straight only traffic.)

Ongoing Q-Learning During Override

It should be pointed out that the Q-learning component in CRQL is not disabled at any time, even at times of the straight-only override. The algorithm continues to add state, transition, reward of action to the Q-table at every step. The distinction is simply that in the case of override on, the action taken on the environment may no longer be the pure argmax of the Q-table (due to the possibility of being overridden to cause straight movement). However, we are continuing to use the agent as performing whatever action was actually taken and adjust the Q-values.

The reward remains to be $r(t) = -Q_{total}(t)$ at each step. Therefore, in the event that the override has the effect of minimizing queues, the agent will be more rewarded (with less negative reward), which in theoretical terms may play to the propensity of the agent to relieve congestion. In contrast, under the case that the queue is large due to the availability of override moments (e.g., at the time of triggering, the queue is large), the agent is accompanied by very negative rewards, and the states are undesired.

Overall, the update rule of Q-learning will not change it will be same as Equation (4.9):

with $\alpha = 0.1, \gamma = 0.9$ given. CRQL just has to bias what is done in the environment in a situation of severe congestion to alleviate the issue in a short period instead of

using the learned policy only that may not provide an adequate action in that new state.

Complete CRQL Operational Flow

Summing up all, CRQL functioning in each time-step of the simulation is given in the following step-by-step manner:

1. **State Sensing:** Replace the existing values of the detectors by the new values, to calculate the approach queues $Q_{KA}(t), Q_{MA}(t), Q_{GS}(t), Q_{GN}(t)$ and sum them for $Q_{total}(t)$. And the state representation st (aggregated or full state based on the need to update Q-table) is to be recorded as well.

2. **Update Congestion Windows:** Insert the new $Q_{total}(t)$ on the queue window W_Q and calculate the new $Q_{trigger}(t)$ and $growth\_ratio(t)$. Similarly compute the current average delay $avg\_delay(t)$ and subtract it to the delay window W_D and compute $D_{trigger}(t)$.

3. **Check Override Trigger:** In case the override is not on, check $Cond_{queue}(t)$ and $Cond_{delay}(t)$. On satisfying either of the two, go into straight-only mode according to the trigger logic (set straight_only_active = True, record restriction_release_step and activation_threshold). Test the release conditions, in case override has already been activated, in case $t \geq restriction\_release\_step$ and $Q_{total}(t) \leq Q_{clear}$ and $avg\_delay(t) \leq D_{clear}$, switch off the override (straight_only_active = False) and restore the original paths of the rerouted vehicles.

4. **Action Selection (ε-greedy):** Select a policy whereby the policy of Q-table (or Q-network) action $a(t) \in \{0, 1\}$ using the ε-greedy policy. It is this that causes the learning agent to possess the desired action.

5. **Action with Constraints:** checks the minimal green constraint. In case $a(t) = 1$ and the current phase is not yet finished MIN_GREEN_STEPS then over-rule this and set $a(t) = 0$ (no phase change). Otherwise, apply $a(t)$ the action of the environment. This will either maintain the same phase or it will be transferred to the next phase (assuming that it is allowed).

6. **Override Rerouting (if active):** if straight_only_active = true at this step implies using the straight-only policy. We, at a stage of the algorithm where we are rerouting, notice an approaching traffic and redirect the route of the traffic to go through the intersection, which is in Section 4.3.7.

7. **Anomaly Enforcement:** Compute the impacts of anomalies of this step. This means, with respect to any vehicle on a pothole lane, whether in a pothole zone and modify its pace; with respect to any vehicle on jaywalk-prone lanes there might be a necessity to instigate/continue a jaywalking slowdown as per 4.3.6. It is performed using TraCi commands (setSpeed) right before the pushing forward of the simulation.

8. **Simulation Step:** a step is increased after 0.1s in sumo simulation. This gives an update to the vehicles basing on the existing state of traffic lights, the speed that is imposed and its simulation time is set to $t + 1$.

9. **Reward Observation and Learning:** Notice the reward $r(t) = -Q_{total}(t)$ of the previous step taken. Measures of the new state $s(t+1)$ (phase and detectors after the step). Initialise the Q-value capability with transition $(s(t), a(t), r(t), s(t+1))$. The Q-table update will be the following in the case of tabular QL/CRQL. In DQN, store experience and then train a neural network after some time period).

10. **Logging:** Record useful measures to be analysed e.g. the overall amount of queue $Q_{total}(t)$, average delay $avg\_delay(t)$, phase change or phase action taken, enabled override etc. In our experiments to keep track of the performance over time, we log the summary metrics after every 10 steps (1 second) to prevent information overload.

It is through this loop that the CRQL controller can adapt to the traffic conditions so as to learn optimum signal timing decisions besides avoiding the state of extreme congestion by guarding against it through its override mechanism.

The Algorithm 4, describes the proposed Congestion-Aware Reinforced Q-Learning (CRQL) controller, which extends reinforcement learning by explicitly incorporating congestion awareness and congestion-responsive actions in addition to signal control.

Initially, the controller initializes the Q-table (or RL policy variables) along with congestion monitoring parameters such as the window size W and reroute interval K (Lines 1–2). The road network is preprocessed to extract straight-only connections, which are later used for congestion-based rerouting decisions (Line 2). SUMO and TraCI are then initiated to enable real-time observation and control during simulation (Line 3). For each time step from $t = 1$ to T, the controller observes the current traffic state s_t and key congestion indicators including queue length Q_t and delay D_t (Line 5). Congestion statistics are updated using a rolling window of size W to determine whether congestion is increasing or clearing (Line 6). When the congestion exceeds a predefined threshold, straight-only rerouting is activated to reduce turning conflicts and stabilize traffic flow (Lines 7–8), when congestion clears, the rerouting restriction is deactivated (Lines 9–10). The RL agent then selects an action a_t using an ϵ-greedy Q-policy, and the traffic signal phase is updated only if the minimum green constraint is satisfied (Lines 11–13). Additionally, when $t \bmod K = 0$, rerouting decisions are applied to vehicles according to the straight-only rule (Lines 14–15). The simulation advances by one step, the next state s_{t+1} is observed, and a reward r_t is computed based on congestion level (Lines 16–17). Finally, the RL policy is updated using the Bellman equation, performance metrics are recorded, and the learned policy along with congestion statistics is stored for reporting (Lines 18–20).

Algorithm 4 Congestion-Aware Reinforced Q-Learning (CRQL)

Input: T, Q-table $Q(s, a)$, congestion window W, reroute interval K
Output: Optimized policy and congestion statistics

Initialize Q-table and congestion windows
Extract straight-only connections from network
Start SUMO and establish TraCI connection
for $t = 1$ *to* T **do**
    Observe current state s_t, queue length Q_t, and delay D_t
    Update congestion statistics over window W
    if *congestion threshold is exceeded* **then**
        Activate straight-only rerouting
    else if *congestion clears* **then**
        Deactivate rerouting
    Select action a_t using ϵ-greedy Q-policy
    if a_t *requests phase change* ***and*** *minimum green time is satisfied* **then**
        Update traffic signal phase
    if $t \bmod K = 0$ **then**
        Apply straight-only vehicle rerouting
    Execute simulation step and observe s_{t+1}
    Compute reward $r_t = -Q_t$
    Update Q-table using Bellman equation
    Log traffic performance metrics
Save learned policy and congestion statistics

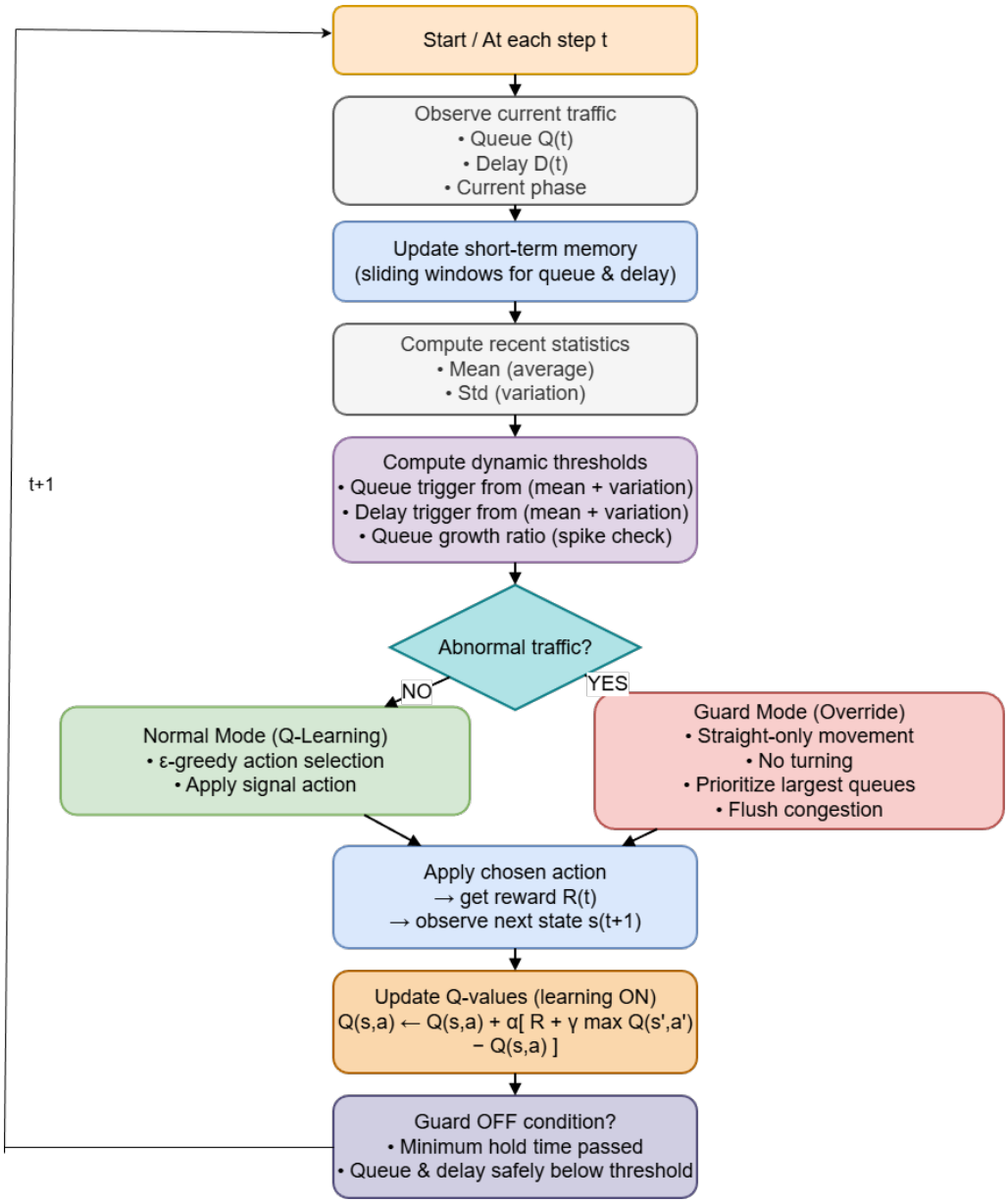


Figure 4.10: CRQL controller flowchart with state observation, congestion detection, -greedy choice of action, congestion-aware rerouting, and Q-value update as part of the SUMO-TraCI simulation process

4.4 Data Collection and Preparation

4.4.1 Dataset Selection and Description :

The measurements of the data used in this analysis were made at the Gulshan-2 intersection in Dhaka, Bangladesh[35] one of the busiest four-legged intersections in the city. Kamal Ataturk Avenue, Madani Avenue, Gulshan North Avenue and Gulshan South Avenue form the joint with each approach having three through lanes with added slip roads on left turning lanes.

4.4.2 Data Collection Method:

Traffic data were recorded on three consecutive working days (December 5 -7, 2023) at the evening peak (4:30 PM -5:30 PM) by using the manual video survey. The enumerators recorded the volume of traffic, direction and vehicle make-up at every approach. The raw figures were then added together to form per hour volumes of demand and turning percentages to be utilized in simulation.

4.4.3 Traffic Volumes and Composition:

The maximum hourly traffic volumes were:

Table 4.2: Base Traffic Flow Data

Approach	**Total Vehicles/hour**	**Major Directions**	**Relative Flow (%)**
Kamal Ataturk Avenue (KA)	2073	To Madani, Gulshan N, Gulshan S, U-turn	68, 16, 12, 4
Gulshan North Avenue (GN)	1927	To Madani, Kamal, Gulshan S, U-turn	6, 38, 52, 4
Gulshan South Avenue (GS)	1589	To Madani, Kamal, Gulshan N, U-turn	35, 6, 58, 1
Madani Avenue (MA)	2668	To Gulshan S, Kamal, Gulshan N, U-turn	11, 64, 21, 4

The types of vehicle were classified as private cars, buses, motorcycles, CNG auto-rickshaws, trucks, and bicycles which represents the heterogeneous traffic characteristic of Dhaka. As an example, the highest percentage of private cars and motorcycles was seen in Madani Avenue, and buses had a relatively higher percentage in Gulshan South Avenue. To prepare the simulation input for SUMO, the extracted field data were structured as base flow information and vehicle composition, as summarized in Table 4.2 and Table 4.3.

Table 4.3: Vehicle Composition and Behavior Parameters Adapted for SUMO

Vehicle Type	**Percentage (%)**	**Max Speed (km/h)**	**Accel (m/s^2)**	**Decel (m/s^2)**	**Representation in SUMO**
Car	35	50	2.6	4.5	`<vType id="car">`
Auto-rickshaw	25	40	2.0	4.0	`<vType id="auto_rickshaw">`
Motorbike	20	60	3.0	5.0	`<vType id="motorbike">`
Bus	10	40	1.8	3.5	`<vType id="bus">`
Pickup/Truck	8	45	2.2	4.0	`<vType id="pickup">`
Bicycle	2	20	1.5	2.5	`<vType id="bicycle">`

4.4.4 Turning Movements:

The finer turning ratios are also documented in the data. For example:

Approximately 68 of the vehicles were heading towards Madani Avenue out of Kamal Ataturk Avenue and 16 of the vehicles were heading towards Gulshan North Avenue. Out of Madani Avenue, 11 percent and 64 percent were flowing Gulshan South Avenue and Kamal Ataturk Avenue respectively. Along the Gulshan North Avenue, nearly 52 percent of the transportation were heading to the Gulshan south Avenue. In Gulshan South Avenue, 58 percent of the traffic direction was to Gulshan North Avenue, and 6 percent to Kamal Ataturk Avenue. These ratios ensure the simulation model is realistic in depiction of directional imbalances of real traffic flow.

4.4.5 Performance Indicators:

The calibration and validation of the baseline VISSIM model was performed based on the recorded data. The validation of the GEH statistics was carried out, and the values were all less than 5, which confirmed that the findings of the simulations were consistent with the traffic flows that were observed. The base-lines of primary performance were:

- Longest queue: 417 meters (Madani Avenue)
- Average delay: 82.7 seconds/vehicle (Gulshan North Avenue)
- Mean time of travel 275 seconds across the intersection approaches.
- Level of service (LOS): E or worse on three out of the four approaches.

These cues confirmed that heavy congestion is an issue at the intersection of Gulshan-2, hence making it a nice case study to apply reinforcement learning to control traffic light.

Model Specification:

In this paper, the adaptive traffic signal control model was developed that combines the reinforcement learning (RL) algorithms with the actual intersection data. The methodology is a combination of the methodological framework by [34] that contrasted Q-learning and Deep Q-Networks (DQN) with traffic signal control with traffic demand and intersection characteristics that were obtained in a case study of the Gulshan-2 intersection in Dhaka [35]. The blend of the algorithmic input of RL with field data on traffic, makes the model technically sound and contextual to the congested South Asian urban traffic conditions.

The obtained field data points were provided into a microscopic simulation in SUMO (Simulation of Urban MObility) and the geometry of the intersection, the arrangement of lanes, and the fixed-time signal configuration at the Gulshan-2 intersection were recreated. The GEH statistic was used to calibrate the model to make sure that the simulated volumes resembled those observed, and all the GEH values were less than 5. According to [35] in Equation (4.26),

$$GEH = \sqrt{\frac{2(M-C)^2}{M+C}} \tag{4.26}$$

where M is the simulated flow and C is the observed flow. An average GEH value of less than 5 was received at all the approaches and this established statistical consistency between the estimated traffic model and the measured traffic conditions. This simulated validating environment was the one where the reinforcement learning agents were trained and tested.

4.5 Implementation of Selected Design

In this section, we explain how the suggested design and controllers were brought into practice, the tools and the setup used to conduct the simulations, the particulars of each controller implementation, and the conditions of the experiment in which they were tested. We also describe the manner in which performance metrics were recorded to be analyzed later.

4.5.1 Tools and Simulation Setup

The experiments all were carried out on top of SUMO (Simulation of Urban MObility) as the traffic microsimulation platform. The value of the simulation step length was 0.1 s, i.e. the traffic state is updated at a rate of 10 times/second, providing the control logic with fine time resolution. We used TraCI (Traffic Control Interface) to interface with SUMO through python to have our outside controller scripts read detector data and change the phase of the traffic light in real time at each step.
The situation (road network and traffic demand) of the Sections 4.2 and 4.3 was installed in the network and route files of SUMO. Our custom controllers were controlling the traffic light at Gulshan-2 intersection using TraCI. To be consistent and fair, all the controllers were tested on the same network and traffic demand. It implies that the random seeds, vehicle spawn times and routes were fixed between controller runs (with the exceptions of the random factors involved in the decision making of each controller or in the learning process).
This simulation environment was executed on a regular PC, and the Python controllers exploited typical libraries (and in our case using the sumolib and traci Python modules that come with SUMO to do so; in the case of the DQN, we used PyTorch to perform neural network calculations). The 1000 seconds of traffic (and the peak period) was simulated in each of the experimental runs, and repeated over and over again where the statistical confidence was required.

4.5.2 Controller Implementation Details

All the controllers (FT, QL, DQN, CRQL) were coded in modules of Python and attached to SUMO through TraCI. Each logic was implemented in accordance with the requirements provided in Section 4.2: **Fixed-Time (FT):** This involves setting a fixed cyclic schedule by programming. The script simply captures time of the simulation and sets the stage depending on the offset of the cycle ut (in Section 4.2.3). We had the cycle splits of known Dhaka signal timing (i.e. assume each phase receives a fixed number of seconds of observation at the field). FT is a time-driven switch which does not require sensor input.

Q-Learning (QL): Q-Learning is implemented based on a Python dictionary that stores Q-values of each state (with the aggregated state representation). In every step, the script calculates the current state and selects an action based on -greedy and then implements it (when not prohibited by min-green) and updates the Q-table when receiving the reward. Aggregation provides a relatively small state space but we have added simple techniques to prevent a state space explosion (such as unobserved states are set to default Q values). Exploration rate and learning rate a were set (0.1) and we arbitrarily permitted to decrease modestly throughout the course of the simulation to give an equal balance between exploration/exploitation.

Deep Q-Network (DQN): Deep Q-Network was implemented as a neural network (we have built a simple feed-forward network topped by two hidden layers of, e.g., 64 neurons, ReLU activations, but we can optimize the structure). The state was the entire 11-dimension vector. An experience replay buffer of a specific size (e.g. 10,000 last transitions) was used and mini-batches were sampled to train the network after a few simulation seconds. The action selection of DQN was also -greedy. We have started with =1.0 and annealed gradually to 0.1 during the first few hundred seconds to promote exploration followed by exploitation. A target network was implemented, which was updated after every 100 training steps. The training was performed as the simulation ran (the simulation would halt a fixed number of steps to have a batch update). This was using the standard DQN algorithm with the adjustments to our simulation scale.

Congestion-Responsive Q-Learning (CRQL): It is the version of the QL controller that is implemented as an extension. The update logic and the same Q-table structure were utilized. Besides that, we introduced the congestion monitoring (holding the windows of the queue and delay) mechanism and the only straight override. This involved traversing the network in order to acquire the straight-edge mapping, and code functions to activate and disactivate the override as indicated. The CRQL controller main loop thus had additional steps to test conditions and make rerouting calls. We also were concerned with timing (in that the reroute instructions should occur after the phase selection but before the simulation step is advanced) to ensure that the vehicles are rerouted in time to influence their movement across the intersection in that step.

Each and every controller has been put through a scenario wherein they are expected to act (e.g. QL and DQN grab queues, the override triggers of CRQL react as we desire when we aggressively push a jam). We made sure that the TraCI commands (such as setting phase, setting speeds, re-directing traffic) were operating without simulation errors.

4.5.3 Experimental Conditions

To experiment on the controllers, we would have two main experimental conditions:

- **Normal Traffic Conditions:** There were no anomalies that were introduced in this case. Traffic is based on the demand data and all the variability is due

to the stochastic nature of vehicle arrivals and the decisions of the controllers. This serves as a benchmark on which each controller is tested to work in a normal environment. We use such measures as average delay, queue lengths, throughput of FT, QL, DQN, and CRQL during the period of the simulation. This case challenges the efficiency of the adaptive controllers in performance optimization to that of the fixed-time plan in the expected environment.

- **Abnormal Traffic Conditions:** Here, we allowed the congestion events of potholes and jaywalking (Section 4.2.6) to behave in a non-recurrent fashion. These anomalies were made consistent across runs to ensure fairness e.g. the pothole could be located on the particular lane of one approach and the random seed of jaywalking events fixed so that each controller gets the same sequence of disturbances. Under these conditions we then ran the simulation with FT, QL, DQN and CRQL. This situation tests the strength of every controller when the traffic is not what it is. It is reasonable to expect that performance of FT declines considerably (due to its inability to respond) and that QL and DQN also perform poorly due to facing states that have never been encountered during training. CRQL, since it has an appeal feature, is believed to cope with such events more effectively by breaking up congestion quickly.

In both cases, the experiment was repeated several times (particularly when using the learning-based controllers, which are rather randomized). But since the simulation is slightly deterministic with identical random seeds (except due to learning randomness) we mostly focused on ensuring that each controller had reached a performance equilibrium or stabilized.

Chapter 5

Result Analysis

This chapter analyzes and compares the performance of all implemented controllers under normal and abnormal traffic conditions using relevant evaluation metrics.

5.1 Performance Evaluation

In this section, we describe systematic review model that will be employed to determine the effectiveness of the traffic signal control measures adopted. Prior to making a controller-wise or scenario-wise comparison, there is a need to explicitly indicate what is being compared, how the comparison is done, and the reason why the chosen measures are sufficient to evaluate adaptive traffic signal control during normal and abnormal traffic flow. The performance analysis of this thesis aims at attaining two main purposes:
i)Baseline efficiency assessment at normal traffic condition and no artificial disturbances are added.
ii)Robustness test in traffic under abnormal conditions, i.e., the traffic flow is interrupted by potholes and jaywalking incidents.
There are four traffic signal controllers that are considered throughout this chapter:

- Fixed-Time (FT)
- Tabular Q-Learning (QL)
- Deep Q-Network (DQN)
- Congestion-Responsive Q-Learning (CRQL) (Proposed).

The testing of all controllers is done by the same intersection geometry, traffic patterns demand, and duration of simulation and frequency of data logging as outlined in Chapter 4. This experimental design allows a fair comparison and the difference in performance which is observed is only likely to be due to the behavior of the controllers and not due to other factors.

5.1.1 Evaluation Methodology and Simulation Horizon

The length of time spent in each single run of the simulation is 10,000 discrete steps of time, and the time of each single step is 0.10 seconds in the real world, which makes it take 1,000 seconds in the simulation (just over 16.7 minutes). This is the minimum length of simulation needed in order to identify the accumulation of traffic in an empty network, temporary congestion, steady-state performance, and the performance degradation or recovery in the presence of disturbances. The 10000 steps size is a trade-off between calculation speed and the requirement to monitor learning behavior, especially to QL and DQN controllers, without the calculation wasting time needlessly. Each 10 simulation steps (1 second) performance measures are taken, which offers sufficient time resolution and minimizes noise in the measured values.

5.1.2 Performance Metrics

The performance of a traffic signal controller cannot be evaluated on a single parameter since there are several dimensions in which the performance of a traffic signal can be measured. This study employs four complementary measures to get a comprehensive picture showing each of the 4 aspects of traffic flow and effectiveness of their control.

Total Queue Length

The total queue length is used to assess the congestion of the intersection at a given point in time. Assuming that the number of vehicles seen in the queue by detector i at time t is denoted as qi(t) and the number of all detectors is denoted as n, then the length of the queue equation 5.1 is:

$$Q_{\text{total}}(t) = \sum_{i=1}^{n} q_i(t) \tag{5.1}$$

This indicator gives a direct measure of the number of vehicles that are compelled to stop or travel at extremely slow velocities. The existence of longer queues means that the signal distribution is not efficient and cars are held up without any need.

Queue length is sensitive to abrupt changes in the traffic (e.g. potholes or jay walking) and can therefore be used to gauge the robustness of a controller when it is subjected to unexpected disturbances.

Average Vehicle Delay

Average vehicle delay is a measure of the waiting time of the drivers. It is formulated in 5.2,

$N(t)$: number of vehicles currently in the network,
$d_j(t)$: accumulated waiting time of vehicle j.

$$\text{AvgDelay}(t) = \begin{cases} \frac{1}{N(t)} \sum_{j=1}^{N(t)} d_j(t), & \text{if } N(t) > 0 \end{cases} \tag{5.2}$$

Whereas the length of a queue is used to measure the congestion of the system at the system level, average delay is the congestion and delay as perceived by the driver, what duration a vehicle must wait on account of the traffic lights.
The two controllers may cause comparable total queue, yet extremely different delays. The user perspective may not be high-performing even with a controller that maintains short queues but allows a number of vehicles to unduly wait. Average delay is thus a very intuitive measure which directly corresponds to driver satisfaction and actual traffic experience.

Throughput

Throughput is used to measure the number of vehicles that make their trips and leave the network successfully. Assuming the total number of vehicles to have traversed the intersection at time t as in 5.3 :

$$\text{Throughput}(t) = \text{Number of vehicles that have exited the network up to time } t \tag{5.3}$$

then this measurement is a direct measurement of the service ability of the intersection controlled by a given controller.
Reduction of delay is inappropriate when vehicles are not effectively cleared off the intersection. A controller may maintain short queues, still may restrict total flow, when it favors certain movements at the expense of others. Throughput makes sure that the reduction of congestion is not obtained at the expense of the decrease in the traffic flow and captures the capacity of the system to serve the vehicles effectively in the long run.

Cumulative Reward

In case of learning-based controllers (QL, DQN, CRQL), a reward signal is used to direct the decisions made by the controller during training or simulation. The reward of this study is a negative congestion measure, proportional to the total queue length (and optionally delay):

$$r(t) = -\,Q_{\text{total}}(t)$$

(or a weighted form as defined previously mentioned in equation 4.4).

$$R_{\text{cum}}(t) = \sum_{k=0}^{t} r(k) \tag{5.4}$$

Cumulative reward in equtaion 5.4 is used to give a running record of the way a controller is able to sustain low congestion on a long-term basis. As opposed to single-point measures, it reflects the consistency of the performance across time:

sharp declines signify instability, poor learning, or the inability to process disruptions. This helps it especially in comparing the learning-based controllers as it can point to both the slow gains as well as the major failures in a single curve.

5.1.3 Evaluation Scenarios

In order to fully test the performance of the traffic signal controllers, we test them in two different scenarios of traffic conditions which are best and worst.

Normal Traffic Scenario

Traffic in this case is moving in ideal conditions, potholes do not exist, jaywalking incidents do not exist, and traffic obeys conventional SUMO dynamics. This situation gives us a baseline estimation, as are able to monitor the way each of the controllers copes with queues, delays and throughput in case of constant conditions. It brings to focus the ability of adaptive controllers to learn and the effectiveness of fixed strategies in a predictable environment.
This scenario evaluates baseline efficiency and learning capability.

Abnormal Traffic Scenario

This situation adds realistic city-level disruptions to controller robustness:

- **Potholes:** The vehicles using the affected lanes have to reduce their speed, resulting in localized bottle hotels and queue up that may affect the entire network.
- **Jaywalking:** Stochastic, random pedestrian crossings cause a temporary blockage in traffic, and the delays are unpredictable.

Simulation based on TraCI is used to inject these disturbances dynamically, so that every controller is given an equal challenge. It is in this case that we can assess the efficiency in non-stationary cases by controllers, their capacity to sustain traffic and how fast they can recuperate following interferences.

5.1.4 Evaluation Philosophy

Rather than concentrating on the end numerical values, this thesis concentrates on the behavior of controllers in the long-run. The important factors that are taken into account are:

- **Temporal dynamics:** Tracking the behavior of metrics not only at the conclusion of the simulation, but also step-by-step.
- **Stability vs. instability:** This deals with identifying the controllers that can be steady in performance as opposed to those that demonstrate abrupt failures during disturbances.
- **Recovery ability:** This measures the speed at which a controller would be able to resume efficient traffic flow following an interruption.

- **Trade-offs between efficiency and robustness:** It is important to understand mechanisms through which controllers trade reducing delays when conditions are normal with ensuring performance in abnormal conditions.

This view is especially significant in the Dhaka traffic where inconveniences such as potholes and jaywalking are ordinary and not extraordinary occurrences. A controller which works well in ideal conditions would not be practical. With the analysis of temporal behavior, stability, and recovery, we are able to decide which controllers are robust, adaptive, and fit in the real-life urban traffic.

5.2 Analysis of Design Solutions

The analysis in this section compares the results of the observed performance in relation to the design features of each of the traffic signal controllers. The analysis is a response to the view of simply stating which controller is the best. It instead focuses on presenting arguments as to why a given controller acts as it does in both normal and abnormal traffic scenarios.
The analysis is organized in the following way:

i) Normal traffic performance (baseline efficiency)

ii) Abnormal traffic conditions (robustness and failure-mode performance)

iii) FT, QL, DQN, CRQL controller-wise interpretation

5.2.1 Normal Traffic Condition Analysis

The performance of the three controllers Fixed-Time (FT), Q-Learning (QL), and Deep Q-Learning (DQN) was evaluated at the Gulshan-2 intersection using identical traffic demand scenarios. Metrics considered include **cumulative reward** (negative proxy for delay) and **queue length** (number of halted vehicles).

Fixed-Time Control (Baseline)

Figure 5.1 shows Fixed-Time's cumulated reward as a function of time. The steep, downwards slope (reaching about 102,000) shows how rapidly delay accumulated under fixed scheduling. Figure 5.2 plots their corresponding queuelengths, usually between 15–20 vehicles. The oscillatory behavior shows how poor fixed cycles are for service under stochastic demand.

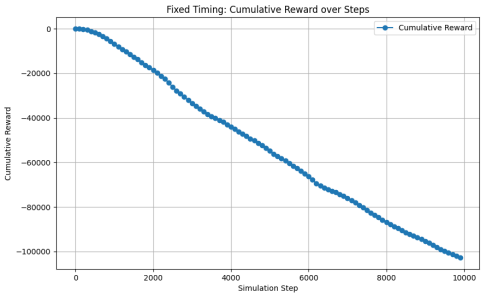


Figure 5.1: Fixed-Time Control: Cumulative Reward over Steps

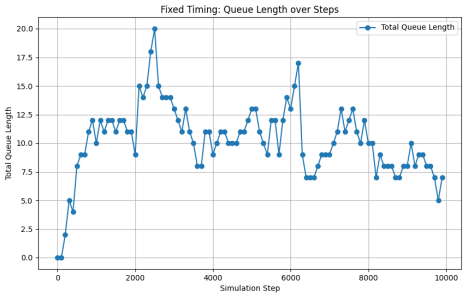


Figure 5.2: Fixed-Time Control: Queue Length over Steps

Q-Learning

In figure 5.3 shows that the Q-Learning the cumulative reward decreased more slowly compared to fixed-time about -70000. This demonstrated the less total delay. On figure 5.4 it shows the queue length stabilized in the range of 8–12 vehicles, with fewer extreme peaks compared to FT. This validates tabular Q-Learning as better adaptable to real-time fluctuations.

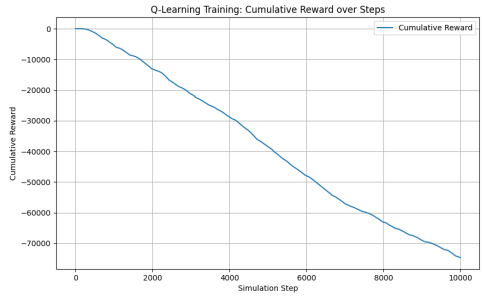


Figure 5.3: Q-Learning: Cumulative Reward over Steps

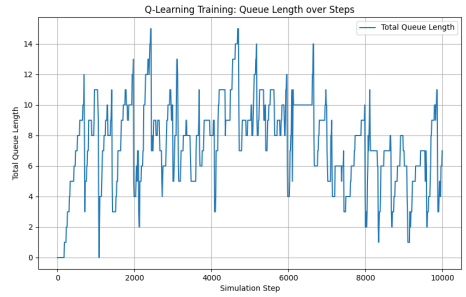


Figure 5.4: Q-Learning: Queue Length over Steps

Deep Q-Learning (DQN)

Figure 5.5 shows the cumulative reward for DQN. Like Q-Learning, the slope is less steep than FT (reaching around –70,000), but with smoother progression. The queue length (Figure 5.6) remained in the 7–12 vehicle range but exhibited fewer spikes than QL, indicating more stable control. Although the improvement margin between QL and DQN is modest, the deep model demonstrates added stability, which would be enhanced with advanced replay and target mechanisms.

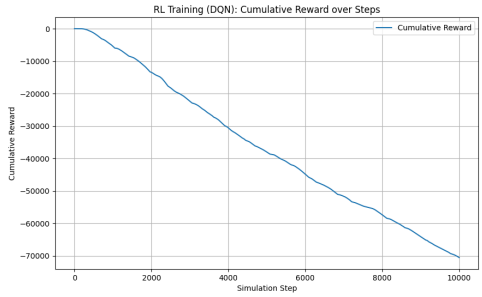


Figure 5.5: DQN: Cumulative Reward over Steps

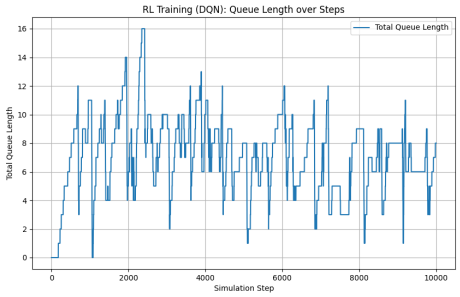
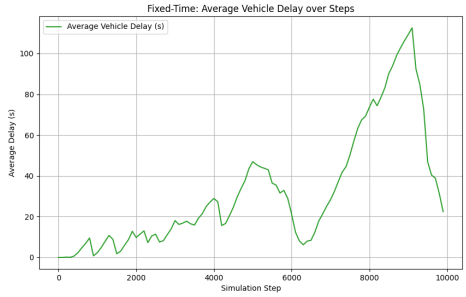


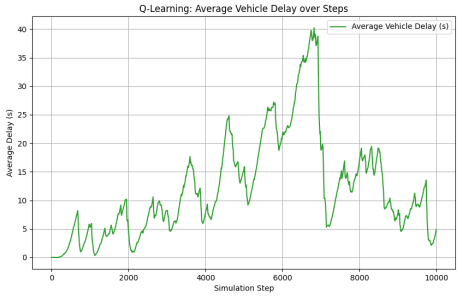
Figure 5.6: DQN: Queue Length over Steps

Comparison Matrices

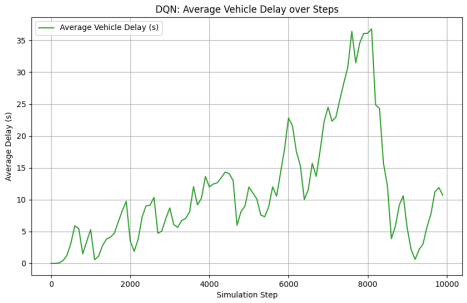
Average Delay Comparison The average vehicle delay means how long on average a vehicle remains stopped/stand still or moves slowly because of traffic signals. It is one of the most direct indicators of traffic congestion. The average delays are observed in the figure 5.7.



(a) Fixed Time



(b) Q-Learning



(c) Deep Q-Learning

Figure 5.7: Average Delay Comparison

The Fixed-Time controller has the highest delay of over 100 sec, because it rigidly follows pre-defined cycles regardless of real-time changes of traffic.

Both Q-Learning and DQN adapt their phase timing reactively, greatly minimizing mean delay by adjusting to current real-time information of queues. Of those, Q-Learning fares somewhat better, its delay values being smaller and more stable. This is due to its easier, table-based method's faster responsiveness in smaller neigh-

borhoods such as Gulshan-2, whereas DQN, even though more sophisticated, needs more training before it generalizes efficiently. Reinforcement learning controllers decrease vehicle delay by approximately 60–70% compared to the fixed-time system, indicating their potential for dynamic traffic control.

Fixed-Time: Highest delay, non-adaptive.

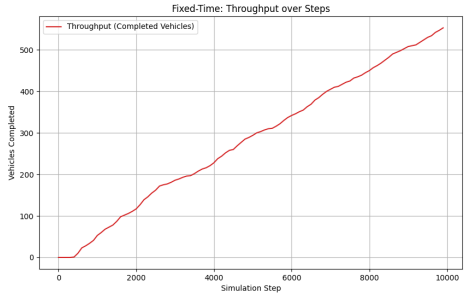
Q-Learning: Lower delay, faster convergence to efficient patterns.

DQN: Improved over fixed-time but slightly less stable than Q-Learning due to model complexity and smaller state-space interaction.

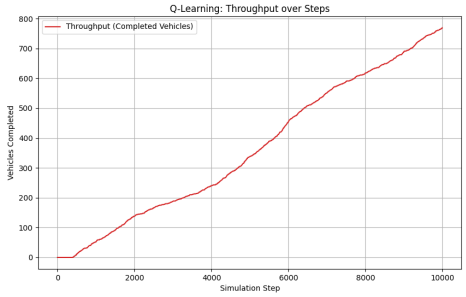
Overall, the Q-Learning controller demonstrates the best balance between adaptability and stability for this intersection, achieving the lowest average delay among the three controllers.

Throughput Comparison

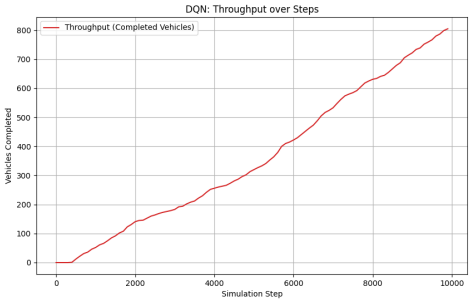
Throughput represents how many vehicles successfully pass through the intersection during the simulation. From the graphs, both Q-Learning and DQN controllers achieved significantly higher throughput compared to the Fixed-Time controller.



(a) Fixed Time



(b) Q-Learning



(c) Deep Q-Learning

Figure 5.8: Throughput Comparison

In the Fixed-time, throughput increased but at slower rate, at the end around 550 vehicles completed in simulation. As Fixed-time has pre-defined cycle and timing, so it doesn't adapt the Changing traffic density.

In contrast, the Q-Learning model showed faster growth, completing around 750 vehicles as it adapts phase changes on real-time, reducing the idle time at low-traffic approaches. The DQN model as it learned to approximate complex state-and-

reward associations, leading to smooth switching of phases and fewer dead intervals. This is why it has the highest throughput with 800 vehicles. Overall, both the RL models processed 35-45% more vehicles than FT, indicating their ability to adaptively respond to dynamic traffic demand and optimize intersection efficiency.

Comparative Analysis

The Fixed-Time has shown the lowest performance. As it operates on fixed time signal cycle, which is why it cannot adapt traffic changes on a real-time basis. However, this is why it causes unnecessary waiting times on low-traffic approaches and extends traffic jams on the busier ones. As a result, the fixed-time gives highest average delay Figure- 5.7 (a) (above 100s) and the lowest throughput Figure-5.8 (a) (550 vehicles). Most of the time FT gives green time while the road is either with few vehicles or no vehicle, this causes queue lengths to reach up to 20 vehicles and a steep cumulative penalty curve.

In contrast, the Q-Learning (QL) showed notable improvement over Fixed-Time. As it is a RL model, it learns from experience, and it dynamically allocates green time where the queues were longest. And as a result, it reduces the average queue length 8-12, reduces the average delay to around 30-40s, and improved throughput to 750 vehicles. QL uses simple tabular design, it consistently reduced unnecessary waiting and smoothed out congestion patterns. This is why it performs smoothly and slower cumulative reward decline compared to FT. The DQN controller, despite having a neural network for estimating Q-values, did not outperform QL due to the fact that here a lightweight version of the model, lacking stabilization elements of replay memory or target networks. Even though it reached about 800 vehicles of throughput and less frequent extreme spikes of queues, it did not quite overcome QL by way of average delay and stability, due to its instability during training.

In summary, both the RL model, QL and DQN adapted the given traffic environment of the Gulshan-2 intersection and performed better than Fixed-time. Both RL models responded to real-time demand, which reduces the delay, and increases the flow efficiency. Q-Learning offered a better overall compromise between adaptability and stability, while DQN showed potential for a better outcome if it were trained for longer times or executed by a more complex architecture.

Adaptivity: Both QL and DQN adapted better to changing vehicle arrivals than FT, which explains their superior performance.

Efficiency: QL provided direct improvements by reallocating green time to busy approaches, while DQN added stability in congestion control.

Limitations: The modest advantage of DQN over QL is due to the lightweight implementation; a full-featured DQN would likely show clearer superiority.

Thus, the comparative results confirm that reinforcement learning significantly improves traffic flow at Gulshan-2 compared to static signals. Q-Learning is particularly effective for smaller intersections with constrained state spaces, while DQN shows promise for larger, more complex networks where generalization is essential.

Finally, after evaluating the model, we observed there is scope for introducing abnormalities in the simulation environment of SUMO. We evaluated the model per-

formance after adding some of the mentioned abnormalities of the roads in the simulation.

5.2.2 Abnormal Traffic Condition Analysis

This subsection serves as the thesis's central idea. The traffic environment becomes non-stationary, stochastic, and capacity-limited in anomalous circumstances. While jaywalking causes sporadic, brief stops, potholes cause long-term speed reductions. The results after introducing abnormalities can be observed in the figure 5.9.

Fixed-Time (FT) Controller – Abnormal Scenario

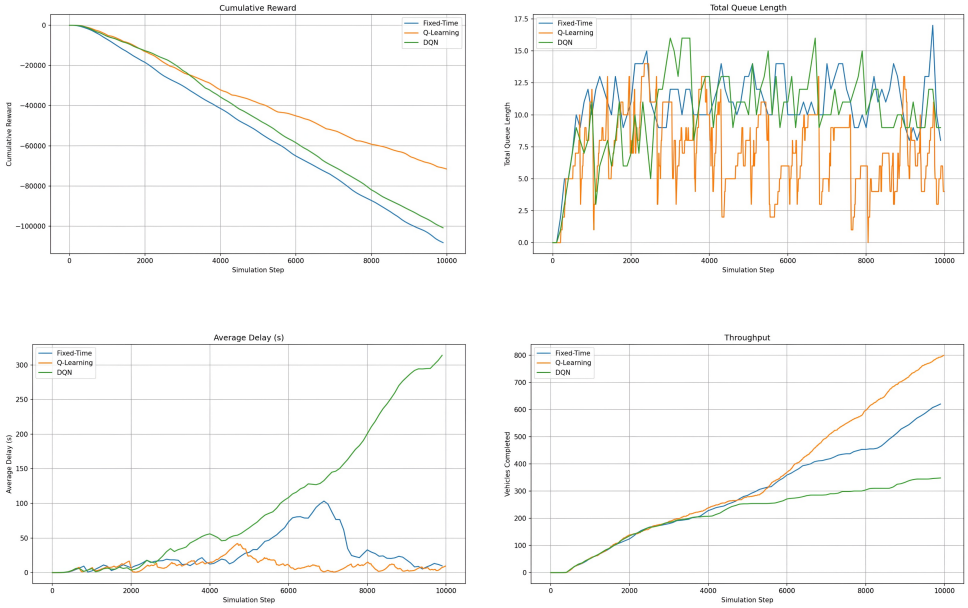


Figure 5.9: Abnormal Scenario

The Fixed-Time (FT) controller has severe difficulties when there is unusual traffic. **observation:**

- For the majority of the simulation, delays rise significantly and remain high.
- During disruptions, lines appear to become out of control and increase quickly.
- Significant decreases in throughput indicate that traffic flow is seriously impeded.

FT is totally unaware of unexpected events such as potholes or jaywalking. It strictly follows its pre-programmed green-time, and it still gives green lights to those lanes that are slackened or congested. Consequently, the green time is wasted, the queues are prolonged and the traffic jam is prolonged.
The rigidity of the FT controller is its primary flaw. It cannot adapt to abrupt

disruptions or decreased lane capacity. When traffic patterns diverge from the ideal, it performs poorly even though it functions flawlessly under ideal circumstances.s.

Queue-Learning (QL) Controller – Abnormal Scenario observation:

- Behavior observed: Delay rises before stabilizing.
- The length of the queue varies.
- Throughput is still constrained but improves over FT.

QL reacts when congestion occurs. It takes lessons out of the extended queues instead of expecting abnormal growth. This allows certain recovery but learning process is slow as compared to the dynamics of the disturbance.
Reactive learning without congestion prediction is the failure mode.

Deep Q-Network (DQN) Controller – Abnormal Scenario observation:

- Delay explodes and never recovers.
- Throughput collapses.
- Reward drops sharply.

The DQN controller is based on a neural network that has been trained to predict the value of different actions. The abnormal conditions like potholes or jaywalking will make the traffic to pass into states which the network has never trained on. Consequently, the controller does not effectively control traffic, and its decisions are of poor quality, sometimes even arbitrary.
The main drawback of DQN is its catastrophic generalization failure, which results in extreme congestion and nearly total performance loss under unforeseen disruptions.

5.3 Final Design Adjustments

In this section we document the final adjustments and refinements of our proposed hybrid model (CRQL). Firstly, we explain why the existing models' (FT, QL, DQN) controllers cannot be made robust by minor tweaking and tuning. Secondly, we discuss how the issues or performance depreciation were addressed in the CRQL model. And lastly, we provide justification for the hybrid model and why it is most suitable for our real-life simulation environment of the Gulshan-2 intersection.
To begin with, we first evaluate why the existing controller models cannot be adjusted:
Fixed Time: In our study the Fixed Time controller operates periodically, maintaining the static cycle time for green-yellow and red. However, it shows two major weaknesses: (i) large delay spikes (for both normal traffic condition and abnormal traffic conditions), (ii) Slow recovery (due to congestion caused by potholes and jaywalking) We have tried using different green splits to see whether it can reduce the issue. But the results do not improve; rather, we have observed that if we increase

the cycle time, the result decreases.

The main reason the Fixed Time controller fails to address the issue is the environment has dynamic, non-stationary entities like potholes and jaywalking. Fixed Time does not track queues or delays, nor can it respond to real-time traffic changes. Configuring for one issue will not be able to solve another disturbance. Therefore, no amount of predetermined offline tuning, such as longer green splits, shorter cycles, etc., can make the controller adaptable and robust for dynamic and unpredictable disturbances.

Q-Learning: Now, let's address why Q-Learning could provide a better solution. The Q-Learning algorithm performed better than Fixed Time in normal situations by changing the signal time based on the traffic demand and queue states. However, it's performance had declined due to: (i) increasing delay due to potholes and jaywalking; (ii) oscillating queue lengths and (iii) slow recovery from congestion. The issue did not fully recover even after tuning the learning rate, discount factor, and exploration rate. The main issue behind the problem is the model fails to address non-stationary states created by random and sudden jaywalking and the effect of potholes on an affected vehicle. Q-Learning reacts after the congestion is formed. It does not have the mechanism to early detect abnormal congestion growth, distinguish between normal fluctuations and structural capacity loss or enforce emergency control actions.

Q-Learning learns gradually and fails to process sudden abnormal disturbances that put the system into unfamiliar states.

Deep Q-Network: DQN had shown the worst performance amongst the three controllers with: (i) rapid delay explosion; (ii) throughput collapse and (iii) persistent instability. It can be argued that the performance is bad because of insufficient model training. Though it is partially true, there are other major issues. If we observe the normal condition, DQN still had failed to converge. And under a non-stationary environment in the abnormal scenario, it had failed catastrophically. It was due to neural networks extrapolating poor performance in unseen stats; it lacks explicit safety and congestion constraints. Increasing training time does not address sudden change capacity loss. Moreover, if we design a complex network, the computational cost will increase and make it more expensive for real-world deployment, without guaranteeing optimized performance.

5.3.1 Motivation and building blocks for CRQL refinement:

By observing the shortcomings of the FT, QL and DQN controllers, we try to build the hybrid CRQL model that addresses:

i) Adaptive learning mechanism similar to QL

ii) Explicit congestion awareness

iii) Rule-based intervention

Our goal was to build a Reinforcement-based model that does not fail to learn even in unpredictable, disruptive situations.
To make the changes, we have to address the congestion detection threshold, refine the straight-only override mechanism, and integrate the minimum green time constraint.

Congestion Control Threshold

To make adjustments in the Congestion Control Threshold, we had to observe how the model behaves and found it triggers frequently and leads to oscillatory behaviors. So we made changes in the threshold and made it dynamic. The dynamic threshold mechanism uses a moving window consisting of the queue length and delay statistics. Instead of using a fixed threshold, it allows adaptiveness. The growth rate is monitored to implement rules in case the queue length grows faster than the predefined ratio, preventing unwanted threshold triggering. To ensure system stability and driving experience, combined delay and queue criteria are followed. Either sustained high delay or rapid queue growth can trigger the mechanism.

Refinement of Straight-Only Override

When the rule intervention is triggered, it is an important matter how long the duration should be. We are enforcing the traffic signal rule with arrow-marked signal lights, which forces a red right arrow on the right. As we have seen, the right-turning vehicles cause a gridlock situation. So forcing straight only mitigates the situation, as left-turning vehicles can use slip lanes and right-turning vehicles can move straight and take a U-turn and choose either the slip lane or a conventional left turn at the intersection. With experimental data we have set a minimum duration (50 steps) for Straight-Only Override. We have seen that if the value is too small, the congestion does not clear and if the duration is too long, the turning vehicle delay is increased unnecessarily. The Release condition was tied to this override mechanism for queue reduction and delay normalization. Minimum hold durations were integrated as hysteresis to prevent unnecessary toggling between the turning and straight-only override.

Integration with Minimum Green Constraint

All the controllers enforce a minimum Green time for ensuring safety. CRQL was designed to respect this requirement, which has an effect on routing but not on the emergency phase switching. A sudden switch may produce road accident situations. Which violates road safety and traffic engineering principles. We integrated safety, keeping road safety in mind. The emergency rule intervention does not affect traffic rule principles nor compromise road safety.

Summary of motivation and refinement

In summary, in normal and less congested situations the model works as QL. It observes the queue length and keeps track in the moving window. If the threshold is

exceeded and triggers the straight-only override, a traffic rule-based intervention is implemented. When the condition stabilizes and goes below the threshold, the rule is waived and the model goes back to the normal learning state.

5.4 Statistical Analysis

Data Aggregation and Analysis Windows

For statistical analysis, raw simulation data were logged at regular intervals during each simulation run:
(i)Simulation step length: 0.10 s
(ii)Logging interval: every 1 second
(iii)Total simulation duration: 10,000 steps

To avoid bias from the initial warm-up phase (when the network is empty and traffic is building up), statistics were computed over the steady-state portion of the simulation. Specifically: The first 2,000 steps were treated as a transient warm-up period. Statistical summaries were computed over steps 2,001 to 10,000. We ensure steady and stable computation by considering the warm-up period and computing from 2001-10,000 steps.

Table 5.1: Controller-wise performance summary under normal traffic conditions (approximate values).

Controller	**Avg. Delay (s)**	**Avg. Queue (vehicle)**	**Throughput (vehicle)**	**Cumulative Reward**
Fixed-Time	∼35–40 (High)	∼9–10 (Moderate)	∼800 (Low–Mod.)	$\approx -1.1 \times 10^5$
Q-Learning	∼25–30 (Lower)	∼7–8 (Lower FT)	∼900 (Mod.)	$\approx -6 \times 10^4$
DQN	∼60+ (Very high)	∼12–15 (Unstable)	∼700 (Low)	$\approx -1.0 \times 10^5$

In our simulation setup, Q-Learning is the most efficient algorithm here under normal environment becuse it minimizes the delays and queue and slightly enhance the throughput and culumative reward compared to other algorithm in this environment. Fixed-Time is stable but medium in that the performance is predictable. DQN could not give an efficient output. Overall, Q-Learning shows best and efficient traffic condition under this condition(Table 5.1).

CRQL is optimal under an abnormal simulated traffic, low delays, limited queues, high throughput, and minimal negative reward. After that, Q-Learning is a middle-ground and is better than Fixed-Time that is not only exceptionally delayed but also has very long queues and negative reward. DQN is the least favorable with enormous delays, unstable queue, low throughput and failed reward. Lastly, CRQL is the best and most stable controller under abnormal conditions(see Table 5.2).

In our simulation, CRQL is the most predictable and variability is low and kept to a minimum and peak delays are controlled. Also, Q-Learning is moderately stable,

Table 5.2: Controller-wise performance under abnormal traffic conditions (approximate values).

Controller	**Avg. Delay (s)**	**Avg. Queue (vehicle)**	**Throughput (vehicle)**	**Cumulative Reward**
Fixed-Time	∼85–90 (Very high)	∼12 (High)	∼600 (Low)	$\approx -1.3 \times 10^5$ (Extremely negative)
Q-Learning	∼50–60 (High)	∼8–10 (Moderate)	∼800 (Moderate)	$\approx -1.0 \times 10^5$ (Very negative)
DQN	$\gg$200 (Extremely high)	∼15+ (Unstable)	∼350 (Very low)	Collapsed (reward diverged)
CRQL	∼15–20 (Low)	∼8 (Bounded)	∼1100 (High)	$\approx -8 \times 10^4$ (Least negative)

Table 5.3: Stability and variability analysis of different controllers under traffic fluctuations.

Controller	**Delay Std. Dev. (s)**	**Queue Std. Dev. (vehicle)**	**Peak Delay (s)**	**Stability Assessment**
Fixed-Time	∼20 (High)	∼4 (Moderate)	∼120 (Very high)	Unstable
Q-Learning	∼10 (Moderate)	∼3 (Moderate)	∼80 (High)	Moderately stable
DQN	∼50+ (Extremely high)	∼6+ (High)	∼300+ (Catastrophic)	Unstable
CRQL	∼5 (Low)	∼2 (Low)	∼40 (Controlled)	Highly stable

whereas Fixed time and DQN are unstable. Overall, CRQL is the most stable algorithm that outperforms other algorithms(see Table 5.3).

Table 5.4: Relative performance degradation under abnormal traffic conditions compared to normal operation.

Controller	**Increase in Avg. Delay**	**Decrease in Throughput**
Fixed-Time	≈150% (from ∼35 s to ∼85 s)	≈25% (from ∼800 to ∼600 vehicles)
Q-Learning	≈100% (from ∼30 s to ∼60 s)	≈11% (from ∼900 to ∼800 vehicles)
DQN	>300% (from ∼60 s to ∼250+ s)	≈50% (from ∼700 to ∼350 vehicles)
CRQL	≈0% (no increase, ∼15–20 s in both cases)	**No drop** (∼960 → ∼1100 vehicles)

CRQL exhibits here no performance degradation with abnormal traffic, where dealy is low and throughput is increased.The throughput increase can be observed in the

Table 5.4. The moderately degraded Q-Learning, the highly degraded Fixed-Time and the extremely degraded DQN was observed here. So, CRQL is the strongest when there are abnormal conditions.

Table 5.5: Relative performance improvement of CRQL compared to baseline controllers under abnormal traffic conditions.

Metric	**CRQL vs Fixed-Time**	**CRQL vs Q-Learning**	**CRQL vs DQN**
Avg. Delay	−78% (78% lower)	−67% (two-thirds lower)	−92% (drastically lower)
Throughput	+83% (nearly double)	+38% (significantly higher)	+214% (over 3× higher)

CRQL outperforms significantly better than all baseline controllers in the case of abnormal simulated traffic. It minimized average delay by 67 - 92% and maximized the throughput by 38 - 214% that of Q-learning, Fixed-Time and DQN(see Table 5.5).

5.5 Comparisons and Relationships

5.5.1 Average Delay (s)

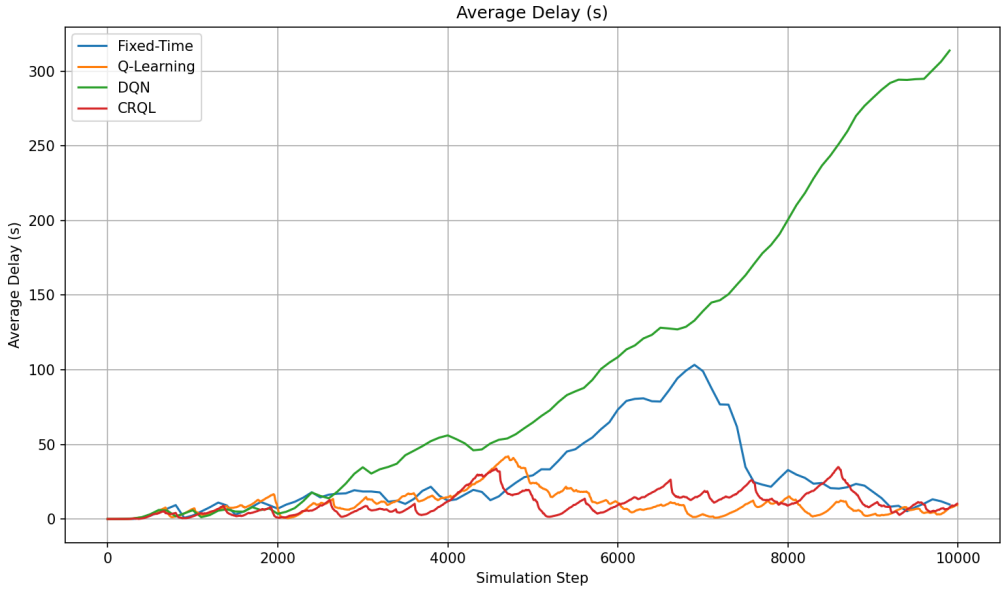


Figure 5.10: Average Delay Comparison

This figure 5.10 number shows how average vehicle delay changed throughout 10,000 simulation steps of all the four controllers. The Fixed-Time controller depicts a slow growth in delay with large variation leading to a full-horizon mean delay of about 28.24 s which indicates a lack of responsiveness to changing conditions during traffic. Comparatively, the DQN controller does not make the best performance, and the delay rises steadily and reaches over 300 s at later steps, resulting in an extremely large overall mean delay of about 103.74 s, which is instability and build-up of

congestion. Q-Learning has relatively low delay during the simulation with a full-horizon mean delay of 10.42 s, though clear oscillations can be seen, implying that it is sensitive to changes in traffic. CRQL also has low and constrained delay along the entire simulation horizon, having a slightly lower mean delay of about 10.37 s. This is equivalent to a 63.3% decrease in average delay over Fixed-Time and 90% decrease over DQN. Although the average delay of the CRQL and Q-Learning have a close value, CRQL does not escalate the delay over time, which implies a better defense against long-term operation. In general, CRQL has consistent and competitive delay guarantees as compared to learning-based baselines.

5.5.2 Throughput

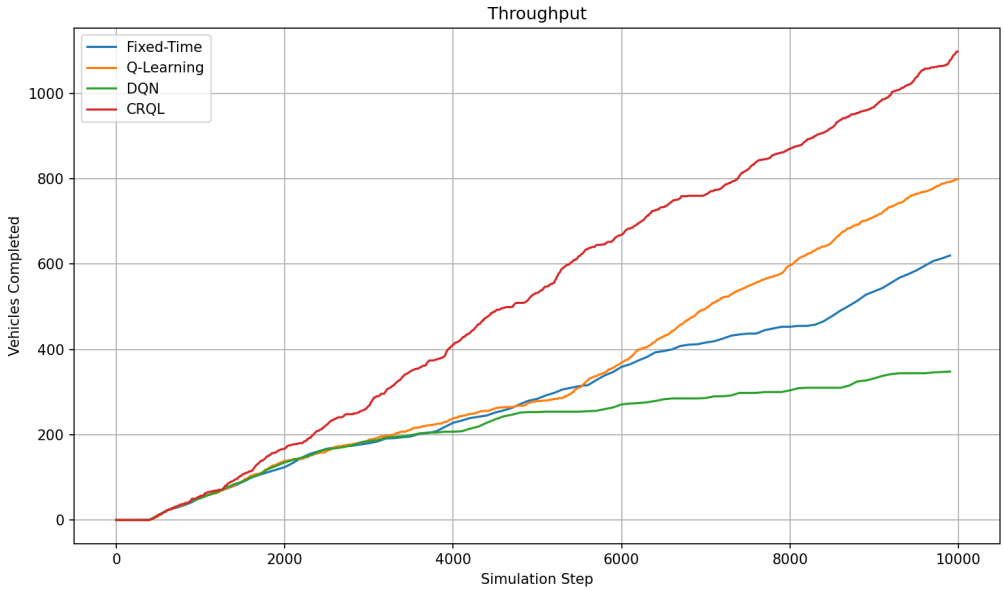


Figure 5.11: Throughput Comparison

This figure 5.11 gives the cumulative figure of vehicles which have been cleared to the network. The DQN controller has the lowest throughput, having finished the work with about 348 vehicles at the conclusion of the simulation, which shows that the discharge of the traffic is inefficient in the conditions considered. Through fixed-time controller the throughput clears about 621 vehicles, whereas under Q-Learning the throughput clears about 799 vehicles, with the help of adaptive signal timing. CRQL is the highest throughput with the vehicles amounting to about 1098 by the end of the simulation. This is a 76.9% improvement over Fixed-Time, 37.4% improvement over Q-Learning and 215.2% improvement over DQN. This is indicated by the fact that the slope of the CRQL curve was continually steeper during all the simulation, indicating long-term discharge of traffic as opposed to immediate benefits. Such findings suggest that CRQL is especially efficient in enhancing network-level traffic clearance during the entire period of the simulation.

5.5.3 Total Queue Length

This figure 5.12 demonstrates how the total length of queue changes throughout the simulation. Fixed-Time and DQN controllers have a persistent queue buildup,

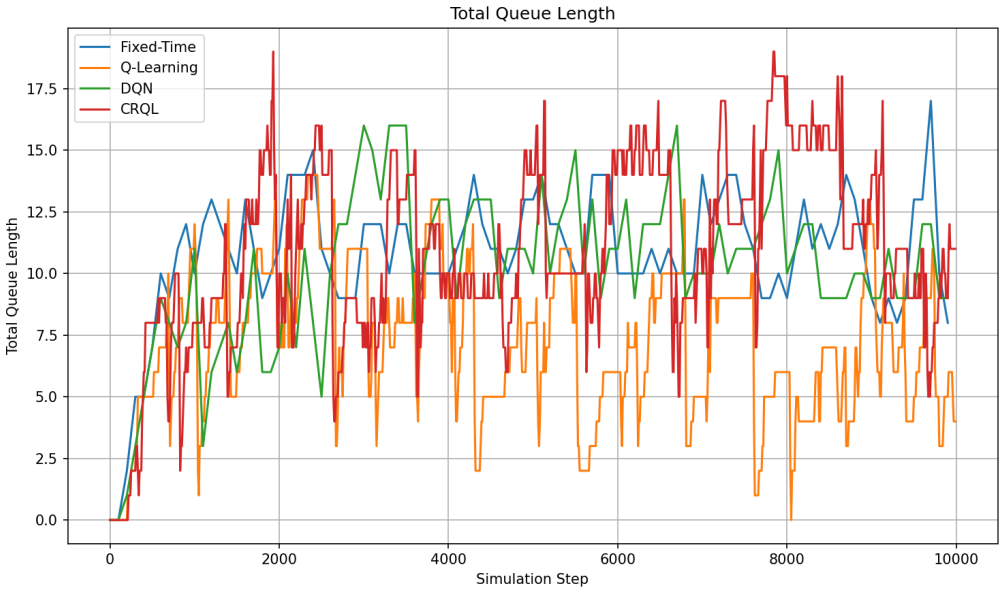


Figure 5.12: Total Queue Length Comparison

and the mean queue lengths of 9.36 vehicles and 8.84 vehicles with full-horizon, respectively, indicate their low responsiveness to real-time congestion. Q-Learning ensures that the raw queue lengths are lower in various intervals and the queue length is around 6.83 vehicles on average though at the cost of the lower throughput. CRQL keeps queue length values within a determined and limited range during the simulation and the average queue length during the simulation is about 9.37 vehicles and does not experience long spikes in the congestions. Even though the raw queue length of CRQL is similar to Fixed-Time, CRQL is simultaneously serving a significantly more substantial number of vehicles. CRQL has the lowest queue-per-served-vehicle ratio when queue length is divided by throughput (about 0.00853), which is a 43.4% improvement in comparison to Fixed-Time and 66.4% compared to DQN. This implies that CRQL provides a more balanced trade-off between queue control and system discharge in terms of traffic discharge and control.

5.5.4 Cumulative Reward

In the figure 5.13 cumulative reward value shows the total reward signal (the cumulative signal) of all the controllers with time. The DQN controller receives huge negative rewards, which can reach up to -100,709, which is in line with its unsteady behavior in control and increase in congestion. The Fixed-Time controller also gathers a high negative reward of about -108,110, which is an indication of its inability to adapt to different traffic demand. Q-Learning has a relatively higher cumulative reward of about -71,575. CRQL gains a cumulative reward of about -108,425 and at the same time serves a significantly higher amount of traffic. On throughput normalization of cumulative reward, CRQL has a reward efficiency of about -98.75 per served vehicle, a 43.3% improvement over Fixed-Time and a 65.9% improvement over DQN. All those observations suggest that the trend of the CRQL reward represents increased system utilization and not a reduction in control quality, which is why the cumulative reward must be interpreted together with the traffic throughput.

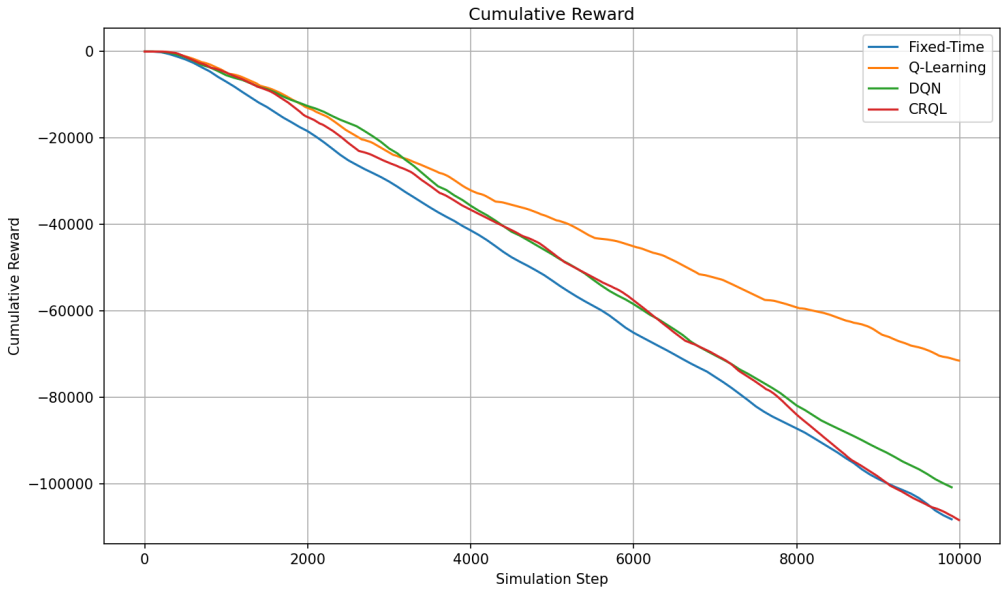


Figure 5.13: Cumulative Reward Comparison

Combining the numbers,, it can be seen that CRQL offers high and stable performance throughout the simulation horizon. Compared to Fixed-Time and DQN, CRQL attains significant gains in the average delay, and the delay under CRQL is similar to that of Q-Learning. Moreover, CRQL has the highest throughput constantly proving its efficiency in the maximization of traffic discharge. Despite some of the isolated metrics (Raw queue length and cumulative reward) showing a favorable result with Q-Learning, the CRQL shows a more balanced and scalable result in the metrics of delay stability throughput and normalized efficiency taken together. These findings indicate that CRQL is an effective traffic signal control approach to maintain traffic in the city within the considered conditions.

5.6 Discussions

In our experimental study, we have observed that amongst the 3 existing controller models, Q-Learning performs better than FT and DQN in normal situations. But when we introduce abnormalities that represent real-world scenarios, we have seen that the CRQL outperforms the base models. It is evident that CRQL has some structural differences from the other controller models, and it reduces gridlock through rule-based intervention. On the other hand, FT is not adaptive; QL and DQN face convergence issues as they learn from past learning and cannot process non-stationary entities. CRQL addresses those issues. In a normal scenario, it learns and for abnormal scenarios and excessive gridlock and congestion, it uses rule-based intervention.

A significant contribution of this thesis is that it evaluates the popular traffic controllers in abnormal road conditions, which are common in developing countries. In real cities similar to Dhaka, we see that the roads are uneven or have potholes and pedestrian behavior is unpredictable; these issues cause disruption in the flow of the traffic.

Controllers that perform well only under idealized conditions provide limited real-world value. The abnormal scenario experiments reveal that (i) FT deteriorates rapidly because it cannot respond, (ii) QL adapts too slowly, (iii) DQN becomes unstable due to non-stationarity, and (iv) CRQL maintains stable operation. Confirming the fact that we need robustness of the controller models rather than peak performance in ideal situations in developing countries.

One insight from this study is that a pure reinforcement learning-based model is not fully stable in a dynamic and disruptive environment. QL and DQN fail to filter the noises in our case: non-stationary jaywalking and vehicles affected by potholes. QL learns slowly, and the Q-Table gets affected due to the non-stationary entities. Congestion is not reversible once severe gridlock is formed. Exploration during heavy congestion degrades system performance. As DQN is neural network based, it shows poor performance for unseen disruption due to generalization. Whereas CRQL protects the learning by introducing threshold and rule-based intervention in critical situations. This hybrid approach protects learning from catastrophic failure.

Despite being a complex model, DQN performs worse than QL and CRQL. DQN assumes the environment is stationary. It requires huge data and training. And still it is not guaranteed that it will perform better, as non-stationary environments disrupt convergence. On the contrary, CRQL uses a moving window to keep the learning state meaningful and concise. It responds to gridlock situations and enforces Straight Only Override to ensure the gridlock situation is mitigated and the flow of traffic is increased. This is important, as in real life we have to keep in mind the randomness, computational capacity, and infrastructural capacity.

From a practical implication standpoint, in urban areas at major intersections with traffic congestion: (i) controllers should not just be learning agents; (ii) controllers should have congestion detection mechanisms; (iii) some critical situations should be encoded rather than the controllers learning them by themselves and (iv) models should be tested under disruptive and noisy environments, not just under steady and stable environments. Our proposed Hybrid controller handles these criteria and creates further scope for future study and development.

Chapter 6

Conclusion

This chapter summarizes the key findings of the thesis, discusses the contributions of the proposed approach, and outlines potential directions for future research.

6.1 Summary of Findings

The study examined the issue of very high congestion at signalized urban intersection in Gulshan-2, Dhaka , which is one of the most congested and busiest intersections in the city. Current traffic signal control in Dhaka is based mainly on a fixed time control program, which by definition cannot respond to changing traffic demand, inconsistent driver behavior and sudden interference to normal operations like road surface damage or pedestrian interference. These constraints inspired the discussion of the adaptive and learning-related traffic signal direction solutions that could react to the current traffic circumstances.

In order to research the problem within a controlled, but realistic environment, a complex simulation environment was created in the SUMO traffic simulator. The road network was created based on OpenStreetMap and the demand in traffic was fine-tuned based on peak hour traffic trends to model real-world congestion behavior. In this setting, four types of traffic signal control methods were introduced and tested: a traditional Fixed-Time (FT) controller, a tabular Q-Learning (QL) controller, a Deep Q-Network (DQN) controller and a newly proposed hybrid method, called Congestion-Responsive Q-Learning (CRQL).

As can be seen, in the normal traffic conditions, learning-based controllers obviously have better performance in comparison to the traditional fixed-time approach. Q-Learning as well as DQN can adjust signal timings according to perceived traffic conditions resulting in reduced average delays, queue length and increased throughput than fixed time control. This justifies the fact that signal control through reinforcement learning is useful in relatively stable and stationary traffic environments whereby the pattern of traffic does not shift drastically.

Nonetheless, as the environment of the simulation is expanded to abnormal traffic conditions the drawbacks of the traditional methods of reinforcement learning become apparent. Slowdowns caused by potholes and slowdowns caused by jaywalking present non-stationarity and sudden disruption to the system of traffic. In

this case, Q-Learning and DQN fall with performance considerably. Their acquired policies, which are optimized to average conditions, find it difficult to respond fast and safely to unexpected congestion build-ups, thus leading to more delays, longer queues and unreliable signal behaviour. The worst result of the fixed-time control is in such situations when it has no mechanism to react to the changes in the demand or anomalies.

The offered CRQL solution, on the contrary, exhibits a significantly better performance and stability when subjected to normal or abnormal traffic. CRQL can identify critical congestion and implement corrective measures to manage excessive congestion and retain the growth of the queue and extended starvation of individual approaches by combining the methods of reinforcement learning with congestion-sensitive interventions based on the rules. Such a hybrid design enables CRQL to support the flexibility of learning-based control and, at the same time, is stable and fair in the situation, when the traffic conditions are not reflected in the nominal patterns.

On the whole, the results suggest that pure forms of reinforcement learning can work in idealized or stable settings, but are incapable of operating alone in complex, irregular and disturbance-prone urban traffic settings like in Dhaka. The fact that CRQL performs better in all the considered scenarios supports the idea that hybrid traffic signal control policies, the ones that combine learning and congestion-sensitive logic, are more appropriate to work in the real-life scenario of an urban traffic system, which is characterised by uncertainty and anomalies.

6.2 Contributions to the Field

We have mainly conducted an exploratory study and then pointed out some issues of the existing solution of traffic congestion intersection controllers and proposed a possible solution to the bottleneck issue. Our contribution consists of-
i) Realistic Intersection Modeling
ii) Integrating Traffic Anomalies in the environment
iii) Comparative evaluation of existing traffic controllers
iv) Proposal of CRQL Hybrid controller

Realistic Intersection Modeling: In our study, we have modelled the intersection of Gulshan-2, Dhaka, in the SUMO simulator. We have used the OSM web wizard and found some inconsistencies. We had to manually use GIS(Geographic Information Systems) in order to verify the road measurements and increase the accuracy of the infrastructure. We used the dataset provided by Zeesan et al. of peak hours and created traffic demand. We installed an IoT-based inbuilt road detection system in the lanes, which takes real-time data of the traffic, including vehicle count, queue length etc. and integrates the information in real time in the traffic controllers. We have visited the Gulshan-2 intersection and interviewed on-duty traffic sergeants to know about the existing fixed-time controller and learnt the green time varies and they have multiple combinations for different entries/exits. Moreover, passersby jaywalk and the fixed time shows inefficiencies in regard to less congested lanes or even free lanes; it completes the full green time even if there is

not even a single vehicle. Some slip lanes of the inner circles are blocked by car parking and street vendors. In our modeling of the environment, we tried to address these issues and incorporate these real-world scenarios. Therefore, our environment setup resembles a real-life scenario.

Integrating Traffic Anomalies in the Environment: We have observed the two major problems of the Gulshan-2 intersection: jaywalking and potholes due to road wear and construction work. We have integrated these abnormalities into our environment. Other studies resembled only ideal road conditions. But we have tried to add realistic abnormalities and found these disturbances introduced non-stationarity, queue shockwaves, and unpredictable delays, allowing a more realistic stress test of traffic signal controllers.

Comparative evaluation of existing traffic controllers: We have studied the most used existing Fixed Time controller, Q-Learning-based controller and Deep Q-Network-based controller. We have evaluated the controllers in ideal, non-anomalistic environments. We compared the results and evaluated the findings. Later on we have introduced anomalies in the simulation environment. We have seen how the models perform. We saw some bottlenecks and pointed out that for random jaywalking and vehicles affecting potholes, the Q-Learning-based controller failed to handle the real-time Q-table and DQN shows catastrophic failure in regard to converging for too many non-stationary entities. Even the Fixed Time controller outperformed DQN. This type of stress testing helped us to find out the shortcomings of the existing system.

Proposal of CRQL Hybrid Controller: By analyzing the drawbacks and bottlenecks of the better-performing QL controller, we have proposed a hybrid controller (CRQL), which is responsive to non-stationary entities. In our study we have seen imposing rule-based intervention provide better congestion control. CRQL integrates tabular Q-Learning with dynamic congestion detection, adaptive thresholds, and a straight-only rerouting override mechanism. While still learning from the environment, this controller provides a stable and effective response even in extreme situations.

6.3 Recommendations for Future Work

Multiple intersections and combined traffic networks: In our studies, we could not study the Gulshan-2 intersection due to a lack of data. If we had multiple connecting intersections' datasets, we could have built a larger connecting traffic network. In the future, we hope to conduct field surveys, collaborate with traffic experts and civil engineers, and create datasets of multiple intersections.

Assessing Emergency Situations and Lanes: In Bangladesh, there are no dedicated lanes for emergency vehicles such as ambulances, fire trucks, police or important personnel movement. That is why when such emergency situations arise, the overall traffic system fails to provide priority to the emergency vehicle(s). In

our study, we could not evaluate such emergency vehicle or VIP protocol movements.

Diverse Anomalies: In our studies we could only add potholes and jaywalking. But in other intersections of Dhaka city and throughout Bangladesh, there are other anomalies such as risky overtaking, reckless lane changing, illegal parking, road blockades, illegal stopping of cars, CNGs, autorickshaws, buses etc.

Advanced Model Integration: To keep the simulation smooth, we could not use complex algorithms and explore deeper into neural network-based models. In the future, we would like to explore more advanced models, including Double DQN and dueling DQN, to see if these stabilize model performance better than DQN.

Real-World Deployment: Ultimately, the goal is deploying the best-performing AI-based controller in the real world and mitigating traffic jams. With help and collaboration from government and traffic engineers, our objective is to slowly deploy the Hybrid CRQL model into Gulshan-2 and then as we conduct more surveys and research in other intersections, we aim to gradually expand the deployment to those other intersections as well.

To conclude, we have studied how the baseline models (FT, QL, and DQN) perform under real-life traffic anomalies and found out the reasons behind their performance drop. We addressed those issues and proposed our CRQL controller. Though it is not fully a better model in all aspects, it is overall providing better throughput in our environment. Better throughput means more numbers of traffic flow, which is actually a way of finding a solution for the bigger picture. As more numbers of vehicles flow, more people reach their destination. Due to lack of a real dataset, we could not test our CRQL controller in other environments and intersections. In the future, we need to stress test our model with more real-world environments and improvise accordingly. By refining the controller further, it will be real-world ready for deployment in intersections of Dhaka City and maybe the scope can be broadened to other developing countries as well.

Bibliography

[1] S. Wang, M. Liu, X. Cheng, and M. Song, "Routing in pocket switched networks," *IEEE Wireless Communications*, vol. 19, no. 1, pp. 67–73, 2012.

[2] A. Stevanovic, C. Kergaye, and M. Zlatkovic, *Scoot and scats: A closer look into their operations*, Online, Accessed: 2025-10-01, 2015. [Online]. Available: https://www.researchgate.net/publication/274137098_SCOOT_and_SCATS_A_Closer_Look_into_Their_Operations

[3] Y. Zhang and A. Haghani, "A gradient boosting method to improve travel time prediction," *Transportation Research Part C: Emerging Technologies*, vol. 58, pp. 308–324, 2015.

[4] S. Mostafi, F. Khan, M. K. I. Antar, and A. Robinuzzaman, "Real time traffic analysis using decentralized siot," Ph.D. dissertation, BRAC University, 2017.

[5] D.-w. Xu, Y.-d. Wang, L.-m. Jia, Y. Qin, and H.-h. Dong, "Real-time road traffic state prediction based on arima and kalman filter," *Frontiers of Information Technology & Electronic Engineering*, vol. 18, pp. 287–302, 2017.

[6] W. Alajali, W. Zhou, S. Wen, and Y. Wang, "Intersection traffic prediction using decision tree models," *Symmetry*, vol. 10, no. 9, p. 386, 2018.

[7] M. Nabi, "Dhaka loses 3.2 m working hours to traffic congestion daily," *Dhaka Tribune*, vol. 5, 2018.

[8] A. R. Abdellah, O. A. K. Mahmood, A. Paramonov, and A. Koucheryavy, "Iot traffic prediction using multi-step ahead prediction with neural network," in *2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, IEEE, 2019, pp. 1–4.

[9] Y. Lin, J. McPhee, and N. L. Azad, "Longitudinal dynamic versus kinematic models for car-following control using deep reinforcement learning," in *Proceedings of the IEEE Intelligent Transportation Systems Conference (ITSC)*, Accessed: 2025-10-01, Auckland, New Zealand, Oct. 2019, pp. 1504–1510.

[10] C. X. You, J. B. Lu, D. Filev, and P. Tsiotras, "Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning," *Robotics and Autonomous Systems*, vol. 114, pp. 1–18, 2019, Accessed: 2025-10-01. DOI: 10.1016/j.robot.2019.01.003 [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889018302021

[11] T. F. Express, "Dhaka gridlock costs tk 153b a day," 2020, Accessed: 2025-01-28. [Online]. Available: https://thefinancialexpress.com.bd/economy/bangladesh/dhaka-gridlock-costs-tk-153b-a-day-1608257836

[12] G. Meena, D. Sharma, and M. Mahrishi, "Traffic prediction for intelligent transportation system using machine learning," in *2020 3rd International Conference on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things (ICETCE)*, IEEE, 2020, pp. 145–148.

[13] J. Zheng and M. Huang, "Traffic flow forecast through time series analysis based on deep learning," *IEEE Access*, vol. 8, pp. 82 562–82 570, 2020.

[14] Anadolu Agency, "Bangladesh loses 40% of fuel due to poor traffic management," *Anadolu Agency*, 2021, Accessed: 2025-01-28. [Online]. Available: https://www.aa.com.tr/en/asia-pacific/bangladesh-loses-40-of-fuel-due-to-poor-traffic-management/2449934

[15] J. Ault and G. Sharon, "Reinforcement learning benchmarks for traffic signal control," in *NeurIPS 2021 Datasets and Benchmarks Track (Round 1)*, 2021. [Online]. Available: https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/file/9bf31c7ff062936a96d3c8bd1f8f2ff3-Paper-round1.pdf

[16] C. Ma, G. Dai, and J. Zhou, "Short-term traffic flow prediction for urban road sections based on time series analysis and lstm_bilstm method," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 5615–5624, 2021.

[17] H. Nguyen-An, T. Silverston, T. Yamazaki, and T. Miyoshi, "Iot traffic: Modeling and measurement experiments," *IoT*, vol. 2, no. 1, pp. 140–162, 2021.

[18] S. Park, J. Lee, S. Bae, S. Kim, and S. Choi, "Deep reinforcement learning for traffic light control in intelligent transportation systems," in *Proceedings of the 2021 IEEE International Conference on Big Data and Smart Computing (BigComp)*, IEEE, 2021, pp. 179–186. DOI: 10.1109/BigComp51126.2021.00038

[19] Y. Mu, S. F. Chen, M. Y. Ding, J. Y. Chen, R. J. Chen, and P. Luo, "Ctrl-former: Learning transferable state representation for visual control via transformer," in *Proceedings of the 39th International Conference on Machine Learning (ICML)*, Accessed: 2025-10-01, Baltimore, MD, USA, Jul. 2022. [Online]. Available: https://proceedings.mlr.press/v162/mu22a/mu22a.pdf

[20] M. Saleem, S. Abbas, T. M. Ghazal, M. A. Khan, N. Sahawneh, and M. Ahmad, "Smart cities: Fusion-based intelligent traffic congestion control system for vehicular networks using machine learning techniques," *Egyptian Informatics Journal*, vol. 23, no. 3, pp. 417–426, 2022.

[21] W. Xiao, H. Liu, Z. Ma, and W. Chen, "Attention-based deep neural network for driver behavior recognition," *Future Generation Computer Systems*, vol. 132, pp. 152–161, 2022.

[22] T. M. Zhu, M. J. L. Boada, and B. L. Boada, "Intelligent signal control module design for intersection traffic optimization," in *Proceedings of the IEEE 7th International Conference on Intelligent Transportation Engineering (ICITE)*, Accessed: 2025-10-01, Beijing, China, Nov. 2022, pp. 522–527. [Online]. Available: https://www.mdpi.com/2032-6653/15/6/246#B4-wevj-15-00246

[23] Y. Ali, M. Rafay, R. D. A. Khan, M. K. Sorn, and H. Jiang, "Traffic problems in dhaka city: Causes, effects, and solutions (case study to develop a business model)," *Open Access Library Journal*, vol. 10, pp. 1–15, 2023. DOI: 10.4236/oalib.1109994 [Online]. Available: https://www.scirp.org/journal/paperinformation?paperid=124756

[24] J. Chen, Z. Zhou, Y. Duan, and B. Yu, "Research on reinforcement-learning-based truck platooning control strategies in highway on-ramp regions," *World Electric Vehicle Journal*, vol. 14, p. 273, 2023, Accessed: 2025-10-01. DOI: 10.3390/wevj14100273 [Online]. Available: https://doi.org/10.3390/wevj14100273

[25] Kallol Mustafa, "Why exactly is dhaka the slowest city in the world?" *The Daily Star*, Oct. 2023, Accessed: 2026-01-25. [Online]. Available: https://www.thedailystar.net/opinion/views/news/why-exactly-dhaka-the-slowest-city-the-world-3436751

[26] Y. Liu, X. Chen, and J. Wang, "Deep q-network-based traffic signal control models: A comprehensive survey and performance evaluation," *Transportation Research Part C: Emerging Technologies*, vol. 145, p. 103 901, 2023. DOI: 10.1016/j.trc.2022.103901

[27] M. M. Rahman and N. Nower, "Attention based deep hybrid networks for traffic flow prediction using google maps data," in *Proceedings of the 2023 8th international conference on machine learning technologies*, 2023, pp. 74–81.

[28] P. Redhu, K. Kumar, et al., "Short-term traffic flow prediction based on optimized deep learning neural network: Pso-bi-lstm," *Physica A: Statistical Mechanics and its Applications*, vol. 625, p. 129 001, 2023.

[29] S. A. Sayed, Y. Abdel-Hamid, and H. A. Hefny, "Artificial intelligence-based traffic flow prediction: A comprehensive review," *Journal of Electrical Systems and Information Technology*, vol. 10, no. 1, p. 13, 2023.

[30] J. R. Tan, "A method to plan the path of a robot utilizing deep reinforcement learning and multi-sensory information fusion," *Applied Artificial Intelligence*, vol. 37, p. 2 224 996, 2023, Accessed: 2025-10-01. DOI: 10.1080/08839514.2023.2224996 [Online]. Available: https://doi.org/10.1080/08839514.2023.2224996

[31] The Financial Express, *Losing time and money in Dhaka's traffic gridlock*, https://today.thefinancialexpress.com.bd/editorial/losing-time-and-money-in-dhakas-traffic-gridlock-1696090104, Accessed: Jan. 28, 2025, 2023.

[32] M. Azhagiri et al., "Deep learning based predicting urban traffic congestion with rgb-coded images using gru-cnn and lstm," *Multimedia Tools and Applications*, vol. 83, no. 38, pp. 86 261–86 280, 2024.

[33] H. Dui, S. Zhang, M. Liu, X. Dong, and G. Bai, "Iot-enabled real-time traffic monitoring and control management for intelligent transportation systems," *IEEE Internet of Things Journal*, 2024.

[34] C. Ouyang, Z. Zhan, and F. Lv, "A comparative study of traffic signal control based on reinforcement learning algorithms," *World Electric Vehicle Journal*, vol. 15, no. 3, p. 137, 2024. [Online]. Available: https://www.mdpi.com/2032-6653/15/6/246

[35] M. Zisan, T. Rahman, S. Ahmed, and R. Khan, "Evaluation of an existing intersection model to minimize congestion using ptv vissim software: A case study of gulshan-2, dhaka," in *Proceedings of the International Conference on Transportation and Traffic Engineering (ICTTE)*, Case study dataset paper for Gulshan-2 intersection, IEEE, 2024.

[36] B. R. T. Authority, *Bangladesh road transport authority (brta) traffic information*, Accessed: 2025-01-28, 2025. [Online]. Available: https://brta.gov.bd/site/page/74b2a5c3-60cb-4d3c-a699-e2988fed84b2

[37] Numbeo, *Traffic rankings by country*, Accessed: 2025-01-28, 2025. [Online]. Available: https://www.numbeo.com/traffic/rankings_by_country.jsp

[38] W. H. Organization, *Who / bangladesh country data*, Accessed: 2025-01-28, 2025. [Online]. Available: https://data.who.int/countries/050

[39] Prothom Alo, "New traffic lights at 22 intersections in dhaka for tk 180 million," *Prothom Alo English*, Apr. 2025, Accessed: 2025-10-01. [Online].

Available: https://en.prothomalo.com/bangladesh/city/fjvvxp4z03#:~:text=New%20traffic%20signal%20lights%20are,traffic%20lights%20came%20from%20there

[40] Worldometers, *Bangladesh population*, Accessed: 2025-01-28, 2025. [Online]. Available: https://www.worldometers.info/world-population/bangladesh-population/