

**PUBLIC TRANSPORT ROUTE OPTIMIZATION  
WITH REINFORCEMENT LEARNING**

**TAY BEE SIM**

**UNIVERSITI TUNKU ABDUL RAHMAN**

**PUBLIC TRANSPORT ROUTE OPTIMIZATION WITH  
REINFORCEMENT LEARNING**

**TAY BEE SIM**


**A project report submitted in partial fulfilment of the  
requirements for the award of Bachelor of Engineering (Honours)  
Electronics (Computer Networking)**

**Lee Kong Chian Faculty of Engineering and Science  
Universiti Tunku Abdul Rahman**

**May 2023**

**DECLARATION**

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature :  \_\_\_\_\_

Name : Tay Bee Sim \_\_\_\_\_


ID No. : 1804019 \_\_\_\_\_

Date : 20/5/2023 \_\_\_\_\_

**APPROVAL FOR SUBMISSION**

I certify that this project report entitled “**PUBLIC TRANSPORT ROUTE OPTIMIZATION WITH REINFORCEMENT LEARNING**” was prepared by **TAY BEE SIM** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Electronics (Computer Networking) at Universiti Tunku Abdul Rahman.

Approved by,

Signature :   
\_\_\_\_\_  
Supervisor : Ir Ts Dr Tham Mau Leun  
Date : 20/5/2023

Signature : *keckhor*  
\_\_\_\_\_  
Co-Supervisor : Dr Khor Kok Chin  
Date : 20/5/2023

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2023, TAY BEE SIM. All right reserved.

## ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to all who have contributed to the successful completion of this project. Firstly, I would like to thank my research supervisor, Dr. Tham Mau Leun for his invaluable advice, guidance, and enormous patience throughout the research development. Secondly, I wish to thank my co-supervisor, Dr. Khor Kok Chin for his guidance and efforts in inviting Professor Somnuk Phon-Amnuaisuk to provide the Reinforcement Learning Workshop. I am grateful for the guidance and insights provided by Professor Somnuk Phon-Amnuaisuk, which have been instrumental in shaping the direction of my research study.

Lastly, I would also like to express my gratitude to my loving parents and friends who helped and encouraged me throughout this project.

## ABSTRACT

High transportation demand due to a large population has resulted in traffic congestion problems in cities, which can be addressed through public transport. However, unbalanced passenger demands and traffic conditions can affect the performance of buses. The stop-skipping strategy effectively distributes passenger demand while minimizing bus operating costs if the operator can adapt to changes in passenger demands and traffic conditions. Therefore, this project proposes a deep reinforcement learning-based public transport route optimization where the agent can acquire the optimal strategy by interacting with the dynamic bus environment. This project aims to maximize the passenger satisfaction levels while minimizing bus operator expenditures. Thus, the dynamic bus environment is designed based on a bus optimization scheme that comprises one express bus followed by one no-skip bus to serve stranded passengers due to skipped actions. The reward function is formulated as a function of passenger demand, in-vehicle time, bus running time and passenger waiting time, which is used to train the double deep Q-network (DDQN) agent. Simulation results show that the agent can intelligently skip stations and outperform the conventional method under different passenger distribution patterns. The DDQN approach yields the best performance in the static passenger demand scenario, followed by the scenario with dynamic passenger demands according to time, and lastly the randomly distributed passenger demand scenario. Future studies should consider the load constraints of buses and other factors, such as bus utilization rate, to improve the performance of stop-skipping services for passengers and operators. Real-life passenger data could be incorporated into the DRL model using Internet of Things technology (IoT) for route optimization.

## TABLE OF CONTENTS

<b>DECLARATION</b>		<b>i</b>
<b>APPROVAL FOR SUBMISSION</b>		<b>ii</b>
<b>ACKNOWLEDGEMENTS</b>		<b>iv</b>
<b>ABSTRACT</b>		<b>v</b>
<b>TABLE OF CONTENTS</b>		<b>vi</b>
<b>LIST OF TABLES</b>		<b>ix</b>
<b>LIST OF FIGURES</b>		<b>x</b>
<b>LIST OF SYMBOLS / ABBREVIATIONS</b>		<b>xii</b>
<b>CHAPTER</b>		
<b>1</b>	<b>INTRODUCTION</b>	<b>14</b>
1.1	General Introduction	14
1.2	Importance of the Study	17
1.3	Problem Statement	17
1.4	Aim and Objectives	18
1.5	Scope and Limitation of the Study	19
1.6	Contribution of the Study	19
1.7	Outline of the Report	20
<b>2</b>	<b>LITERATURE REVIEW</b>	<b>21</b>
2.1	Route Optimization	21
2.2	Skip-Stop Services	21
2.3	Reinforcement Learning	22
2.3.1	Policy-based and Value-based Learning Algorithms	23
2.3.2	Q-learning Algorithms	25
2.4	Deep Reinforcement Learning	26
2.4.1	Deep Learning	26
2.4.2	Deep Q-Network	28
2.4.3	Double Deep Q-Network	30



2.5	Previous Studies on Public Transport Optimization Model	30
2.5.1	Evaluation Metrics	32
2.6	Summary	33
<b>3</b>	<b>METHODOLOGY AND WORK PLAN</b>	<b>34</b>
3.1	Introduction	34
3.2	Software Setup	34
3.2.1	Google Maps Platform	34
3.2.2	Flask	35
3.2.3	TensorFlow	35
3.3	Data Collection	36
3.3.1	Bus Route	36
3.3.2	ETA Data	37
3.4	Bus Optimization Scheme	38
3.5	Deep Reinforcement Learning Model Framework	41
3.5.1	Passenger Demand Model	42
3.5.2	State Space Design	43
3.5.3	Reward Function Design	43
3.5.4	Double Deep Q-Network Agent	47
3.6	Model Training	49
3.6.1	Training Environment	49
3.6.2	Training Workflow	50
3.7	Model Evaluation	53
3.8	Model Deployment in Web Application	54
3.9	Gantt Chart	54
3.10	Summary	55
<b>4</b>	<b>RESULTS AND DISCUSSION</b>	<b>56</b>
4.1	Introduction	56
4.2	Results of Model Training	56
4.2.1	Training Result of Scenario 1	57
4.2.2	Training Result of Scenario 2	57
4.2.3	Training Result of Scenario 3	58
4.2.4	Loss of Model Training	59
4.3	Results of Model Evaluation	59

4.3.1	Evaluation Result of Scenario 1	59
4.3.2	Evaluation Result of Scenario 2	61
4.3.3	Evaluation Result of Scenario 3	62
4.4	Results of Model Deployment in Web Application	64
4.5	Summary	66
<b>5</b>	<b>CONCLUSIONS AND RECOMMENDATIONS</b>	<b>68</b>
5.1	Conclusions	68
5.2	Recommendations for future work	69
	<b>REFERENCES</b>	<b>71</b>

**LIST OF TABLES**

Table 3.1:	Location of Ten Bus Stops assigned along the Bus Route.	36
Table 3.2:	Alighting Rate for Each Station along The Route.	42
Table 3.3:	Boarding rate for each station along the route.	43
Table 3.4:	Hardware Specification of Training Environment.	50
Table 4.1:	The parameter settings in DRL model.	56
Table 4.2:	Average Evaluation Score of DDQN and Conventional Approaches in Scenario 1.	60
Table 4.3:	Skip-stop pattern of DDQN Agent in Scenario 1.	61
Table 4.4:	Average Evaluation Score of DDQN and Conventional Approaches in Scenario 2.	62
Table 4.5:	Skip-stop pattern of DDQN Agent in Scenario 2.	62
Table 4.6:	Average Evaluation Score of DDQN and Conventional Approaches in Scenario 3.	63
Table 4.7:	Skip-stop pattern of DDQN Agent in Scenario 3.	63

## LIST OF FIGURES

Figure 1.1:	Reinforcement Learning Model (Martín-Guerrero and Lamata, 2021).	16
Figure 2.1:	Deep Neural Network (IBM, 2020).	27
Figure 2.2:	Deep Q-Network Workflow (He, et al., 2021).	29
Figure 3.1:	Bus route of UTAR bus generated with Google Maps.	37
Figure 3.2:	Stored ETA Data in Excel File.	38
Figure 3.3:	Bus Optimization Scheme.	38
Figure 3.4:	DRL Model Framework.	41
Figure 3.5:	Workflow of action selection in DDQN Model.	47
Figure 3.6:	Architecture of the DDQN agent.	49
Figure 3.7:	Simulation Flow of First Bus Trip.	51
Figure 3.8:	Simulation Flow of Second Bus Trip.	52
Figure 3.9:	Simulation of Model Training in an Episode.	53
Figure 3.10:	Gantt Chart of Project Timeline for the June 2022 Semester.	54
Figure 3.11:	Gantt Chart of Project Timeline for the January 2023 Semester.	55
Figure 4.1:	Training Results over 5000 episodes for Scenario 1.	57
Figure 4.2:	Training Results over 5000 episodes for Scenario 2.	58
Figure 4.3:	Training Results over 5000 episodes for Scenario 3.	58
Figure 4.4:	Loss of Model in all Three Scenarios.	59
Figure 4.5:	Evaluation Result of Scenario 1.	60
Figure 4.6:	Evaluation Result of Scenario 2.	61
Figure 4.7:	Evaluation Result of Scenario 3.	63
Figure 4.8:	User Interface of Web Application.	64

Figure 4.9:	Query Panel of Web Application.	65
Figure 4.10:	Initial Stage of Bus Simulation in Live Map.	65
Figure 4.11:	Bus Simulation Result from S01 to S02.	66
Figure 4.12:	Bus Simulation Result from S04 to S06.	66

## LIST OF SYMBOLS / ABBREVIATIONS

$a$	action
$A_{i,j}$	arrival time of bus $i$ at station $j$
$D_{i,j}$	departure time of bus $i$ at station $j$
$G_t$	cumulative reward
$H_{i,j}$	headway of buses at station $j$
$l_i$	load of bus $i$
$L_{i,j}$	passenger left behind by bus $i$ at station $j$
$L$	loss function
$J$	policy score function
$P_{i,j}$	passenger demand at station $j$
$q_\pi$	action-value function
$Q$	Q-value
$R$	reward
$s$	state
$t_{j-1,j}$	travel duration between station $j-1$ and station $j$
$T_{r,jk}$	running time between stop $j$ and stop $k$
$T_{v,jk}$	in-vehicle time between stop $j$ and stop $k$
$\overline{T}_{w,j}$	average waiting time of passengers at stop $j$
$\overline{T}_{ws,j}$	average waiting time of passengers stranded at stop $j$
$U_{i,j}$	number of boarding passenger at station $j$
$V_{i,j}$	number of alighting passenger at station $j$
$v_\pi$	state-value function
$W_{i,j}$	number of waiting passenger at station $j$
$Y_t$	target network
$\gamma$	discount factor
$\theta$	weight
$\pi$	policy
$\tau_{i,j}$	dwel time of bus $i$ at station $j$
$\lambda_{j,k}$	arrival rate of passengers heading to stop $k$ from stop $j$

CNN	convolutional neural network
DDQN	double deep Q-network
DL	deep learning
DNN	deep neural network
DQN	deep Q-network
DRL	deep reinforcement learning
ETA	estimated time of arrival
HTML	hypertext markup language
HTTP	hypertext transfer protocol
IoT	Internet of Things
MDP	Markov Decision Process
ML	machine learning
RL	reinforcement learning
VRP	vehicle routing problem
WSGI	Web Server Gateway Interface

## CHAPTER 1

### INTRODUCTION

#### 1.1 General Introduction

Most cities face the problem of traffic congestion due to the high transportation demand from a large population. Public transportation is considered as an alternative solution to reduce transportation demand and improve the urban transportation system. Transit network planning is crucial for the efficiency of public transport, and it is divided into three levels: strategic, tactical, and operational (Lu, Han, & Zhou, 2018). In strategic planning, the transit network design considers the routes, types of vehicles, number and location of stations, and distance between stations to meet the mobility needs of the population. Tactical decision making involves optimizing the service frequencies and timetables of public transport to improve the transportation service level (Ibarra-Rojas et al., 2015). Lastly, operational strategies provide control services to minimize operating costs.

All planning for public transportation faces challenges related to traffic conditions and unbalanced distribution of passenger demand along the lines. The predetermined schedules and static routes used in conventional methods for bus and railway systems cannot accommodate the increasing transportation demand in cities. Moreover, it is impractical to continuously expand and upgrade the infrastructure and services of transit systems. Therefore, operation control strategies are alternative solutions to further improve the reliability and efficiency of service while minimizing operating costs. The bus system commonly implements these operation control strategies:

- **Short-turn:** The bus only serves part of the route with higher demand.
- **Deadheading:** Empty buses return to the line's starting point in the low-demand direction to swiftly serve the high-demand direction.
- **Holding:** Buses are delayed at specific stops for a particular duration to synchronize their arrival with other buses or passengers.
- **Stop-skipping:** Bus only stops at certain stops with robust passenger demand and skips the stops with lower passenger demand.



According to Ortega et al. (2019), the stop-skipping approach suits transit routes with unbalanced passenger demand. The optimal skip and stop stations can be planned and designed according to passenger needs. In 1947, the Chicago Metro System began offering skip-stop service to distribute passenger demand effectively at certain stations during peak hours. The Japanese railway system adopted skip-stop operations by providing alternative routes and rapid train services to serve stations with higher passenger demand. In the bus system, skip-stop operations must consider traffic, passenger demand, and operating costs to establish effective and reliable public transport routes.

An ideal public transport route optimization should intelligently select the optimal route and waypoints. Previous studies used mathematical and heuristic models to optimize skip-stop operations. Many parameters need to be collected and updated occasionally to create a stochastic environment. The emergence of artificial intelligence (AI) has promoted the use of intelligent transportation control. Reinforcement learning (RL) is one of the research trends in optimizing the transit system as it learns through trial and error with minimal human intervention.

Machine learning (ML) is a subset of AI that builds learning methods in machines by using data and algorithms. RL is a type of ML technique that trains the agent to select actions by interacting with its environment. The critical components of RL, which is derived from the Markov Decision Process (MDP) framework, are described below:

- **Environment:** The environment is the setting or context where the agent functions.
- **Agent:** The agent is responsible for making decisions within the environment through interaction. It selects the appropriate action based on its observations and experience.
- **State Space:** The state space comprises information related to the environment, representing a snapshot of the environment at certain time step.
- **Reward:** The reward reflects the quality of action taken, with positive rewards indicating good actions and negative rewards indicating bad

actions. The agent's primary objective is to increase its total reward over time.

- **Policy:** The policy is a decision-making strategy used by the agent to choose actions based on its current state. It is a function that connects states to actions, which governs the agent's behaviour in the environment.

RL uses the concept of reward-motivated behaviour to train the agent to learn from mapping actions to a given set of states through a cumulative reward or punishment. Figure 1.1 shows the workflow inside the RL model. The agent utilizes the state space to obtain information about the environment at a given time  $t$  and then selects the best action based on its policy. The policy maps the action based on the given states in the environment. The agent evaluates the effectiveness of the action in the given state and updates the policy accordingly using the reward. This process is repeated continuously as the agent learns from the states, actions, and rewards, with the ultimate goal of maximizing cumulative reward over time. The agent-environment interaction ends when the state reaches the goal state. The agent learns continuously from the immediate reward in the episodic task and thus maximizes cumulative return by getting the optimal policy. The agent's previous actions directly affect the immediate reward from the environment and influence the decision-making in the next state and all subsequent rewards. RL uses trial and error methods to explore the state-action pair without human intervention and then exploits the experience to increase the model's performance.

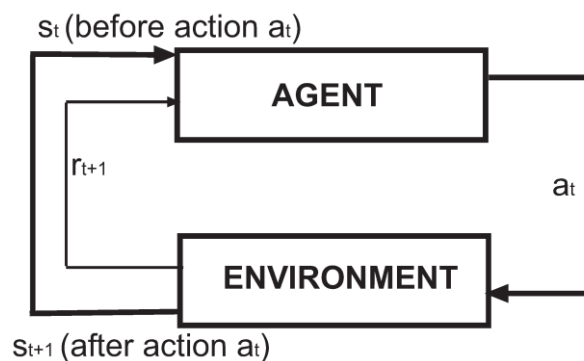


Figure 1.1: Reinforcement Learning Model (Martín-Guerrero and Lamata, 2021).

## **1.2 Importance of the Study**

Nowadays, the increasing transportation demand has raised the need for optimization in the public transportation system. The conventional method that implements static routes and predetermined schedules cannot efficiently distribute the unbalanced demand of passengers. Improper frequency settings in the bus system led to passengers being stranded at bus stations due to full loads. The travel time and waiting time of passengers are also prolonged by traffic congestion during peak hours. Previous studies have utilized mathematical models and heuristic algorithms to optimize public transportation systems. The emergence of AI has increased interest in implementing intelligent transportation systems. Thus, this project introduces skip-stop services to address the problem of unbalanced passenger demand using RL techniques.

The project employs a cutting-edge approach using RL to optimize public transport routes. By providing the estimated time of arrival (ETA) and passenger demand along bus lines, the RL agent can learn to make real-time decisions through trial and error, which improves decision-making over time. This approach provides an efficient way to generate the best route for skip-stop service based on factors such as passenger demand, traffic conditions, and travel times. Hence, the main importance of this project is to explore the possibility of RL in optimizing public transport routes. The project's results could improve the quality of the skip-stop service provided by considering passenger demands, travel times, and the total trip duration of the bus along the route.

## **1.3 Problem Statement**

The city's current public transport system has become overwhelmed due to traffic congestion. The conventional public transportation system implements a fixed route with a predetermined schedule which cannot adapt to the disturbances in the predetermined route, such as traffic jams, accidents or maintenance work. Due to the disturbance on the route, public transport cannot arrive at the destination before a given deadline, leading to delay problems. The prolonged travel duration might delay the arrival time of public transport and even might affect the schedule planning.

Besides that, the number of passengers varies at different times. In peak hours, the static route suits the high passenger demands at every station but might face overcrowding and unbalanced passenger distribution problems. It can be improved by adjusting the frequency setting of bus. However, the conventional stop approach at every station, even if the demand is low during the off-peak time, leads to resource waste and passenger dissatisfaction. A static route is not optimal in this scenario as it increases travel time without meeting utilization rates. To address this, this project proposes an optimization scheme that includes stop-skipping to solve the Vehicle Routing Problem (VRP) with RL technique. It needs to determine the optimal route from the starting points to various sets of waypoints based on the demand of passengers. VRP even becomes complicated when real-time information, such as traffic conditions is considered.

The previous researcher has proposed various algorithms to optimize the routing and scheduling problem in the public transport system. However, it involves complex computation to determine the optimal route, mainly when applied in real-life applications. Developing deep learning (DL) leads to research on implementing RL in route optimization with deep neural networks to reduce computation time. Currently, studies focusing on improving the skip-stop service in bus route optimization using the DRL technique still need to be completed.

#### **1.4 Aim and Objectives**

The primary aim of this study was to develop a deep reinforcement learning (DRL) model to optimize public transport routes. To achieve this, several objectives were identified and pursued throughout the course of the research:

- Develop a deep reinforcement learning model to optimize the stop-skipping operation of bus
- Implement the solution on Google Map,
- Compare with the conventional method, which uses predetermined routes.

### **1.5 Scope and Limitation of the Study**

This study concentrates on implementing a DRL model in public transport route optimization. In this project, the UTAR bus is used as public transport for testing purposes. Ten bus stops have been designated along the original UTAR bus route. Real-time ETA data between the stations along the bus routes were collected from the Google Maps Platform. The DRL model has been developed and trained with Double Deep Q-Network (DDQN) using data collected from Google Maps. The performance of DDQN and conventional approaches is being compared and evaluated using the reward function designed. The design of the reward function in this project considers various factors, including passenger demand, passengers waiting time, passengers in-vehicle time and total travel duration of bus. Lastly, the trained DRL model has been deployed in a web application to show the simulation results.

A limitation of this project is the lack of real-time data on passenger demand at each station. The model needs to generate the passenger demand for simulation, which may not accurately reflect real-time conditions. Therefore, different passenger demand models were generated to examine the performance of the DRL model in different scenarios. Besides that, there are some restrictions on the service provided by the Google Maps Platform. It only gives the data for present and future departure times; the past travel duration in traffic conditions cannot be obtained. The vehicle type cannot be set to bus in Distance API's driving mode, which might cause inaccurate results.

### **1.6 Contribution of the Study**

This study proposes a novel approach to optimizing public transport routes using DRL. It explores the DRL approach in optimizing the route with real-time data from the Google Maps Platform. Besides that, it also focuses on enhancing the stop-skipping operation in express buses by considering various factors like passenger demand and traffic conditions. The reward function, which considers different evaluation metrics, is designed in the DRL model. The performance of the DRL and conventional approaches is compared and evaluated using the designed evaluation metrics. The DRL model has also been deployed in a web application to simulate and display the results. The

application can simulate the route selection process based on real-time ETA data from Google Maps.

### **1.7 Outline of the Report**

The report is structured into five main chapters, which include an introduction, a literature review, a methodology and work plan, results and discussion, and a conclusion. The introduction chapter provides an overview of public transport route optimization and RL while explaining the problem statement, objectives, scope, and limitations of the study. It also highlights the importance and contribution of the project. Chapter 2 comprehensively reviews the public transport optimization model and the skip-stop service. The chapter also explores the theoretical concepts behind RL and DRL and further discusses the working principles of Q-Network, DQN, and DDQN. The literature review includes a summary of previously conducted studies on the subject and their respective evaluation metrics. Chapter 3 explains the DRL model framework and the work plan for the entire project. Chapter 4 presents the training results for three different passenger demand scenarios. The DDQN agent's performance is assessed using the reward function designed and compared to the conventional approach. Lastly, Chapter 5 concludes the study's results and provides recommendations for future improvement.

## CHAPTER 2

### LITERATURE REVIEW

#### 2.1 Route Optimization

The main objective of public transport route optimization is to improve the efficiency of routes, taking into account the benefits for both passengers and bus operators. The design, service, and operation of public transport are the three stages of transit network planning that contribute to its enhancement, as discussed in Chapter 1. One way to optimize public transport routes at the operational stage is through stop-skipping control, where the travel duration and passenger flow are the important criteria. Ibarra-Rojas et al. (2015) the stochastic nature of passenger distribution patterns and traffic flow can significantly affect the efficiency of the transportation system. However, obtaining different data, such as traffic conditions, real-time travel duration and the actual passenger flows at the station, from open-source platforms is a challenging task. Therefore, the inadequacy of data is a critical challenge in optimizing public transportation routes. In real-life scenarios, public transport route optimization also needs to balance meeting passenger demand while minimizing operating costs. However, the more constraints considered, the greater the computational requirements, as noted by Iliopoulou and Kepaptsoglou (2019).

#### 2.2 Skip-Stop Services

Gkiotsalitis and Cats (2021) defined stop-skipping service as a control measure that allows the bus to skip certain stops if it is running behind schedule. Instead of stopping at all stations along the line, the express bus only serves a subset of stations according to demand. Stop-skipping can balance the passenger demand while minimizing the travel time.

Many studies have been undertaken on optimizing public transportation routes to improve and enhance the skip-stop operations in public transportation systems. Kikuchi and Vuchic (1982) defined stopping regimes into three basic types: all-stop, on-call stopping, and demand stopping. The all-stop regime restricts the vehicle to stop at every station, while the on-call

stopping allows the vehicle to serve those stations with boarding and alighting demands. In the demand stopping regime, passengers can request the vehicle to stop at any stations along the line for boarding and alighting. This study even found that the optimal stopping regime and the number of stops change according to passenger demand. As the travel demand rises along the route, the optimal stopping regimes change from demand stopping to on-call stopping and finally the all-stop regime (Kikuchi and Vuchic, 1982). It indicates that the stop-skipping operation is more suitable for the scenario with unbalanced passenger demand.

### **2.3 Reinforcement Learning**

RL takes the concept of reward-motivated behaviour in human beings to train the machine to make decisions through reward and punishment mechanisms. Several vital elements in the reinforcement learning model originate from the Markov Decision Process (MDP). According to Sutton and Barto (2018), MDPs are a straightforward framework for learning from trial and error to achieve goals. Thus, MDP is an ideal mathematical framework representing the RL environment as it can formalize sequential decision-making in the RL model. There are five critical components in MDP: state, action, transition probability, reward, and discount rate.

MDP has a finite set of actions, rewards and states. At every discrete time step, the agent interacts with the RL environment, obtains the current state space, and selects the appropriate action. The environment responds by transitioning to the next state and providing a reward for the state-action pair. The agent takes the next current state and reward to evaluate the policy used. The policy in RL can be defined as the probability of transition from the previous state-action pair to the state. The agent can learn from the reward and update the transition probability, which is the probability distribution over the set of actions.

The agent aims to optimize the overall return. The environment returns the reward at every discrete time step. The episodic task is a multi-decision process in RL, and it ends when the environment reaches the goal state and resets all parameters. The interaction between the environment and agent can run continuously without limit. The return,  $G_t$  is the sum of all the



rewards,  $R$  calculated at the end of the time step  $T$ . In this scenario, the agent would consider the cumulative return in future compared to the immediate reward. However, this continuing task might sometimes run infinitely, leading to missing a larger total reward in the long run. The single action that gives a decent reward might be ignored in cumulative rewards. According to Lee (2019), a discount mechanism discounts the importance of future rewards at different time steps so that the cumulative reward can be valued differently depending on the scenario. The discount factor,  $\gamma$  falls between zero to one. When the  $\gamma$  approaches zero, the agent learns to choose the action to maximize the immediate rewards (Sutton and Barto, 2018). The discount factor,  $\gamma$  allows the agent to consider more the immediate reward, especially in a particular state. If the discount factor,  $\gamma$  approaches one, the agent would consider the future rewards more. The discount factor helps to achieve the balance between immediate and cumulative rewards.

### 2.3.1 Policy-based and Value-based Learning Algorithms

The policy defines the behaviour of agents in decision-making. Equation 2.1 defines the policy,  $\pi$  as the probability distribution that maps actions,  $a$ , from the given state,  $s$  in MDP (Jagtap, 2020). The agent maps the action to a given state at the time,  $t$  using the state-transition probability matrix of all possible actions. The agent discovers the optimal policy to improve the cumulative rewards over time, taking into account the discount factor which increases the importance of immediate rewards.

$$\pi(a|s) = P[A_t=a | S_t=s] \quad (2.1)$$

where

$P$  = probability distribution

$A_t$  = action at time,  $t$

$S_t$  = state at time,  $t$

In the model-free RL, there are mainly two learning methods: policy-based and value-based. Both types of learning are used to store the previous experience for obtaining the optimal policy. The policy-based learning focuses

on finding the optimal policy directly by using the state-action probabilities, while value-based learning uses the value function to find the optimal policy.

Policy-based learning defines the policy as a set of parameters  $\theta$ . The policy score function,  $J(\theta)$  calculates the expected reward under the policy. By exploring the environment and observing the expected rewards under the policy, the policy parameters  $\theta$  that maximizes the policy score function  $J(\theta)$  can be calculated by performing gradient ascent. It is constantly updated to obtain the optimal policy. In short, the policy-based methods train the policy directly without using the value function. The standard algorithms in policy-based methods are the Proximal Policy Optimization, Policy Gradient and more.

The value-based learning finds the optimal policy by implementing the state-value function. The agent trains to learn which state can get more reward and takes action to lead to it. Equation 2.2 shows that the value function returns the expected value for a given state. The Q-function is denoted by Equation 2.3, where the expected return is called the Q-value.

$$v_{\pi}(s) = E_{\pi} [R_{t+1} + \gamma v_{\pi}(S_{t+1}) | S_t = s] \quad (2.2)$$

$$q_{\pi}(s, a) = E_{\pi} [R_{t+1} + \gamma q_{\pi}(S_{t+1}, A_{t+1} | S_t = s, A_{t+1} = a)] \quad (2.3)$$

where

$v_{\pi}(s)$  = value function

$q_{\pi}(s, a)$  = Q function

$E_{\pi}$  = expected value given under policy,  $\pi$

$R$  = reward

$\gamma$  = discount factor

Q-function links a state and action to the expected total return if that action is taken in that state. The 'Q' in Q-function can be interpreted as representing the quality of the action in that state, as it reflects the expected cumulative reward. Equation 2.4 expresses the relationship between the Q-function and the value function. The agents either learn from the value function or Q-function based on the algorithms used.

$$v_{\pi}(s) = \sum_{a \in A} \pi(a|s) q_{\pi}(s, a) \quad (2.4)$$

### 2.3.2 Q-learning Algorithms

The Q-learning algorithm learns by taking different actions and updating the Q-table with maximum action values. It trains the agent using the greedy action that gives the maximum Q-value at a given state. The agent learns through exploration and exploitation in RL. In Q-learning, the Q-table is used to store all Q-value of state-action pairs. In the initial stage of training, the Q-value for all state-action pairs is zero. The agent needs to explore the environment to obtain the Q-value of the state-action pair and store it in the Q-table. As the agent keeps learning, the Q-table keeps updating each state-action pair. The agent also exploits the existing Q-value and updates the Q-table with rewards. However, a balance between exploration and exploitation is important to ensure the agent can continue learning all possibilities in state-action pairs. Thus, the Epsilon greedy strategy is introduced to balance the exploration and exploitation rate in the agent. In this strategy, the exploration rate decreases as the episodic task increases. The agent generates a random number between 0 and 1. It starts to exploit the environment when the random number is higher than the exploration rate.

The main idea of the Q-learning algorithm is to obtain the optimal Q-value for every state-value pair that gives the maximum expected return. The learning rate determines the amount by which the previous Q-value should be incorporated into the new Q-value calculation. Generally, the learning rate is constant in the training process. As the learning rate approaches 1, the new Q-value increasingly overrides the previous one, and vice versa. It is crucial to determine the appropriate learning rate when applying Q-learning to solve a problem. The calculation for the new Q-value of the state-action pair can be seen in Equation 2.5, which considers the previous Q-value, discounted Q-value and learning rate.

$$Q_{new}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [R_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \quad (2.5)$$

However, the size of the Q-table increases significantly as the number of state-action pair increases. It has led to the slow convergence speed of Q-learning. As mentioned by Ganapathy, Soh and Joe (2009), the agent faced difficulty exploring all possible state-action pairs in mobile robots due to the enormous computational burden, a great deal of time required, and the increasing size of states in the environment. As data increases and becomes more complex, the conventional Q-table cannot handle the problem. The neural network can replace the Q-table to improve Q-learning and form deep Q-learning. Thus, the authors have focused on integrating Q-learning with the artificial neural network to train the autonomous mobile robot to learn in an environment. The neural network is functional when there is a complex numeric computation which can improve the Q-learning further.

## **2.4 Deep Reinforcement Learning**

Scalability is the main challenge faced in RL because real-life applications consist of high-dimensional states that are different from the simple state in MDP. As the number of datasets increases, the training process becomes more complicated, and the amount of memory and time needed also increases significantly. The deep neural network (DNN) can help the RL model handle vast amounts of data and thus introduce the idea of deep reinforcement learning (DRL). DRL integrates RL with deep learning (DL) to implement the DNN to perform function approximation for both policy and value functions.

### **2.4.1 Deep Learning**

DL is a machine learning technique inspired by the human brain's neurons. It implements a brain-like logical structure of algorithms called the neural network to learn specific knowledge by observing a large set of labeled data. The neural network is considered deep when the number of processing layers more than two. Data feeds into the deep neural network, and the model continues to iterate until output accuracy reaches an acceptable level. It is useful in automating predictive analytics by collecting, analyzing and interpreting large amounts of data.

DNN comprises of multiple layers, including the input, hidden, and output layers. The hidden layer consists of multiple processing layers, and

each layer consists of interconnected nodes, as shown in Figure 2.1. The weight is a parameter assigned to the links between interconnected neurons in different layers that determine the influence of inputs on output. Forward propagation proceeds with the computation from the input to output layer, which chooses the output action based on the weight. The input layer ingests the data for processing, and the neurons in hidden layers build upon the previous layer to refine the data. In the hidden layer, the input is computed with the weighted sum from the previous layer. The activation function is applied to the neuron to get a new transformed input from the previous layer (Li, 2017). In backpropagation, the error derivatives in prediction are computed backward, and the weight and bias are adjusted to optimize the loss function by moving from output to input layers. With this DNN mechanism, the error in the result can be used to update the weight of the related link and thus gradually increase the accuracy of prediction.

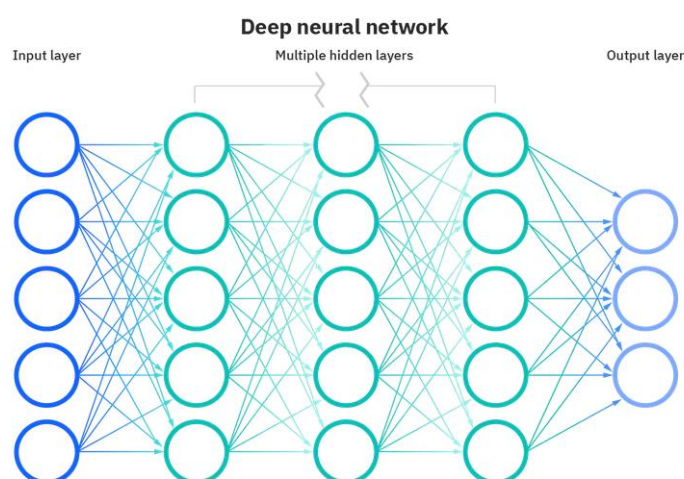


Figure 2.1: Deep Neural Network (IBM, 2020).

DNN learns from large amounts of raw data and computes the prediction automatically. Forward propagation and backpropagation of the DNN can be implemented in the RL learning algorithms to form DRL. It can be utilized as a function approximation to optimize the policy and value function, which handles the scalability problem in RL.

### 2.4.2 Deep Q-Network

DQN is a commonly employed algorithm in the field of DRL, which merges Q-learning with deep learning. The implementation of DQN to play Atari games by the Google DeepMind team is a milestone in DRL. Mnih et al. (2013) successfully combined Q-learning with convolutional neural networks (CNNs), resulting in an agent that played the Atari games better than ordinary human players. CNNs are deep neural networks that specialize in finding patterns in images, making them suitable for the Atari games where the agent learns from the video input.

Q-learning utilizes the Q-table to save the Q-values for all state-action pairs, while DQN utilizes the Q-Network to approximate the value function in Q-learning. The input layer in the Q-network represents the state, the output layer represents the action, and the hidden layer stores the weights and biases for the DQN. Without processing the data, the problem of high correlation between adjacent training data and non-stationary distributions might affect the performance of the agent in DQN. To achieve unconditional convergence like the Q-learning algorithm, DQN was implemented by combining two approaches: experience replay and the target network.

Experience replay mechanism saves the previous transition in a memory buffer and removes training data randomly to reduce correlations between data. Two neural networks are implemented in DQN: the target and Q-networks. DQN trains neural network by minimizing the loss function that penalizes the difference between Q-network and target network. The loss function compares the Q-value in the Q-network with target network and updates the weight of Q-network in the backpropagation. Equation 2.6 shows the loss function  $L(\theta)$  of DQN.

$$L(\theta) = E [(R_{t+1} + \gamma \max_a Q_{target}(s_{t+1}, a; \theta^-) - Q(s_t, a_t; \theta))^2] \quad (2.6)$$

where

$Q_{target}$  = Q-function of target network

$\theta^-$  = weight of target network

$Q$  = Q-function of Q-network

$\theta$  = weight of Q-network

DQN computes the target Q-value by taking the next state as input, whereas the evaluation network uses the current state as input. The loss function is designed to adjust the weight of the interconnected node in the Q-network by using the target Q-value as an estimated future value. To generate a new estimated value for training, the target network periodically synchronizes the parameters from the Q-network. Figure 2.2 shows the workflow of the DQN algorithm, where the Q-network is represented by the main net and the target network is represented by the target net.

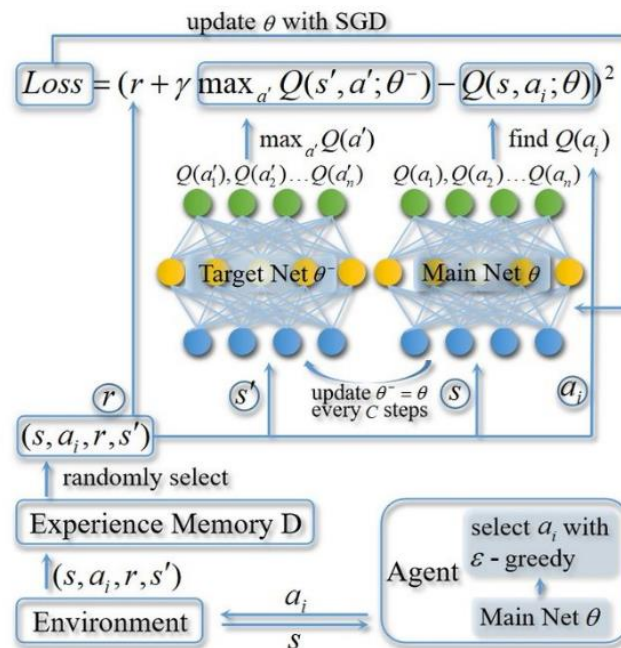


Figure 2.2: Deep Q-Network Workflow (He, et al., 2021).

Implementing neural networks provides a solution to the limitations of Q-tables in handling high-dimensional state spaces. However, it also introduces the problem of data correlation. Unlike Q-learning, which saves the value of state-action pair separately in a table, neural networks determine the best action based on the weights of the nodes in each layer. Backpropagation can modify the weights of other state-action pairs, leading to a decrease in accuracy. To address this, experience replay and target networks are used to

reduce data correlation by randomly selecting data and using an estimated Q-value to compute the loss function.

### 2.4.3 Double Deep Q-Network

There is an overestimation problem in Q-learning where the estimated Q-value is often more significant than the actual Q-value. To solve this problem, Van Hasselt developed double Q-learning in 2010 (Van Hasselt, Guez, and Silver, 2016). This idea was further developed to solve the overestimation problem in DQN by introducing double DQN. Double Q-learning uses two Q-functions to minimize the overestimation problem by updating the Q-function with the value from another Q-function (Jang et al., 2019). This algorithm reduces the bias in Q-learning by dividing the evaluation function. Like Q-learning, the DQN always chooses the best action based on the largest estimated Q-value from the target network, which can exaggerate the previous error, leading to overestimation. Van Hasselt, Guez, and Silver (2016) proposed double DQN to minimize the overestimation problem. The Q-Network and target network predict their Q-values from the current state. The target network provides the actual Q-value based on the maximum action argument in the Q-network. The Q-network chooses the action, while the target network provides the actual Q-value based on the given action. Compared to DQN, DDQN separates the action selection and evaluation in the target network. The improvement can be seen clearly by comparing Equations 2.7 and 2.8 for the target network in DQN and double DQN.

$$Y_t^{DQN} \equiv R_t + \gamma \max Q_{target}(s_{t+1}, a_{t+1}, \theta) \quad (2.7)$$

$$Y_t^{DoubleDQN} \equiv R_t + \gamma Q_{target}(s_{t+1}, \arg \max_a Q(s_{t+1}, a; \theta), \theta) \quad (2.8)$$

## 2.5 Previous Studies on Public Transport Optimization Model

Feng and Chen (2005, cited in Zhang and Jia, 2021) discovered that conventional bus route optimization models mainly employ optimization algorithms and mathematical analysis models. Li, Rousseau, and Gendreau (1991) proposed nonlinear 0-1 stochastic programming model to optimize



real-time bus schedules with skip-stop operations. The nonlinear mathematical model only sought to reduce the overall penalty cost of unsatisfied passenger demands, neglecting the operating cost of the bus. Fu, Liu, and Calamai (2003) approached the stop-skipping control problem as a non-linear programming problem that involves 0 and 1 variables. Their approach factored in the waiting times of both waiting and stranded passengers, the total time passengers spent in transit, and the total travel time for the bus. The optimization algorithm proposed accounted for the costs of both passengers and operators, however, the method employed brute-force techniques. Sun and Hickman (2005) investigated the feasibility of real-time implementation of skip-stop control and presented two objective functions, the policy alternative and basic policy. Both policies considered the waiting and transit time for passengers, but the basic policy also factored in the additional cost to passengers for disembarking at a skipped stop, as the bus does not stop according to the passengers' requests. The study concluded that the policy alternative approach is better suited for real-time use, as passengers can choose their disembarking station using an executive search method.

Heuristics algorithms are explored in several studies to optimize the skip-stop service in public transportation systems. Liu et al. (2013) utilized the Genetic Algorithm with Monte Carlo Simulation to enhance the stop-skipping services in bus system. It implemented an optimization scheme comprising of one express bus line, followed by another no-skip bus line to minimize the overall waiting time of stranded passengers. The researchers aimed to minimize the weighted sum of three objective functions, namely bus operating time, in-bus time and waiting time for the benefit of both passengers and operators. However, the study did not consider the load constraint of the bus. Chen et al. (2015) improved the optimization model of the previous study by considering capacity and random trip time. Zhang et al. (2018) also proposed a genetic algorithm approach but considered the factor of imbalanced passenger demand in executing stop-skipping control. The express bus services will only start if the demand exceeds the specified threshold. However, optimizing the skip-stop service of a bus without considering real-time traffic data is impractical because traffic conditions may change as the bus moves to the next stops.

As machine learning becomes increasingly popular, the RL technique is another option to optimize the public transport system, enabling the agent to learn through trial and error. Jiang et al. (2019) proposed the RL approach in controlling the passenger inflow and generating the skip-stop strategy in the railway system. This RL model considered the penalty value of stranded passengers distributing along the metro line to optimize the train rescheduling. The Q-learning is implemented to deal with the train rescheduling strategies of skip-stopping while considering passenger congestion along a real-life metro line. This study optimizes the skip-stop strategy in the metro line without the need to consider the traffic condition as the bus system. The DRL can also be applied to optimizing the dynamic bus schedule as studied by Ai et al. (2022). The agent is trained to decide whether the bus should depart or hold at a bus station by considering the time, crowded degree, waiting time of passengers, and carrying capacity utilization rate. However, the optimization of the bus system's route was not considered.

### **2.5.1 Evaluation Metrics**

Based on the research of Gkiotsalitis and Cats (2021), the authors summarized the literature on stop-skipping and listed typical evaluation metrics from previous studies. The assessment metrics can be broadly classified into two distinct categories: those that contemplate the advantages to passengers, such as waiting time, in-vehicle time, unsatisfied passenger demand, and extra cost for alighting when skipping stops; and those that consider the benefits to the operator, such as total bus travel time, reduction of control actions, and deviation from the planned schedule.

Based on the findings of the literature review presented in Section 2.5, it can be inferred that various parameters, such as waiting time, number of stranded passengers and in-vehicle time can be used to assess passengers' satisfaction. Similarly, the total bus travel time is the key parameter that can be employed to evaluate the cost of bus operators. In this project, these evaluation metrics will be considered to formulate the reward function of the RL model.

## **2.6 Summary**

The stop-skipping operation is a popular technique for optimizing public transport routes. This chapter introduces the concepts of RL and DRL, along with the working principles of Q-learning, DQN, and DDQN. Finally, previous studies on the subject are reviewed, and their evaluation metrics are summarized.

## CHAPTER 3

### METHODOLOGY AND WORK PLAN

#### 3.1 Introduction

The entire project can be mainly categorized into three stages for the development of the DRL-based public transport optimization model. In the first stage, the ETA data is obtained from the Google Maps Platform, and research is conducted on Google Maps API and RL. In the second stage, the project continues with the design of the RL framework. The environment and agent are built using Python, and the model building and refinement process is repeated through model training and evaluation. The reward function is tuned until the performance of the results reaches optimal levels. In the last stage, the model is deployed in a web application and integrated with Google Maps.

#### 3.2 Software Setup

##### 3.2.1 Google Maps Platform

The Google Maps Platform is the maps services provided in the Google Cloud Platform. It provides various application programming interfaces (API) to retrieve map data such as location, best travel route, and travel time from Google Maps. The developer is allowed to embed the Google Maps into the website or mobile applications by using the API key. It is a closed source tool and charged per API call. The Google Maps libraries are available through different client interfaces such as HTTP request, JavaScript API and even Python. The Maps JavaScript API provides the same functionality as the client libraries but is more specialized in building the maps on the web application.

The API key is needed to access the Google Maps Platform Service. There are three APIs are enabled for this project which are Maps JavaScript API, Distance API and Distance Matrix API. In this project, the ETA data between the stations is essential for model training and evaluation. A full set of ETA data between the bus stations can be collected by using Direction Matrix API. Maps JavaScript API and Distance API will be used to develop the web application for model deployment. The Maps JavaScript API is used to build the customized maps for bus simulation. The Distance API received the

direction requests and returned the direction and route from Google Maps. It can integrate with Maps JavaScript API to display the route in the customized map. The origin and destination of the bus stops are specified by using the longitude and latitude to avoid including the walking distance in path generation.

### **3.2.2 Flask**

Flask is a lightweight and compact web application framework written in Python. It is built on the foundation of Werkzeug, which is a Web Server Gateway Interface (WSGI) toolkit. Werkzeug serves as an intermediary between server-client web applications for handling requests and responses, while WSGI facilitates the communication protocol between web server and web application. Additionally, Flask allows for the rendering of dynamic web applications through Jinja2, a widely-used template engine in Python. By combining variables from a Python script with an HTML file, Flask can deliver dynamic content within web applications. In this project, Flask is utilized to develop a basic web application that supports standard HTTP requests. The "GET" request permits clients to retrieve data from the web server using a URL, while the "POST" request sends data to the server when the client submits a form. The Python server then obtains the required data from the client and returns the simulation response.

### **3.2.3 TensorFlow**

TensorFlow is a library developed for ML and DL applications. It supports many programming languages such as Python, JavaScript, C++ and Java. Libraries in TensorFlow can perform numerical computation to create DL models directly or indirectly and develop neural networks. It supports building and training the DL models easily using the module provided. It is one of the popular frameworks and tools for ML and DL. It is used to build the DRL model for optimizing the route selection in this project.

TensorFlow is a machine learning platform that provides a variety of useful libraries for constructing machine learning models. The libraries within TensorFlow are capable of performing numerical computations to create machine learning and deep learning models either directly or indirectly, and

can be used to develop neural networks. This platform facilitates easy building and training of deep learning models through its provided modules. It is one of the most popular frameworks and tools for machine learning and deep learning. In this project, TensorFlow will be utilized for constructing a deep reinforcement learning model that optimizes route selection of express bus.

### 3.3 Data Collection

The project began with research on the Google Maps Platform. A route with ten stations was designed based on the original route of the UTAR bus. By integrating the Google Maps API with Python, the designed route and stations can be displayed on Google Maps, and ETA data can be collected using the Distance Matrix API.

#### 3.3.1 Bus Route

There are ten bus stops along the KTM Serdang bus route, which have been selected and modified based on the original Route 5: KTM Serdang Station of UTAR bus in Sungai Long Campus. The locations of these ten bus stations are shown in Table 3.1, along with their longitude and latitude coordinates.

Table 3.1: Location of Ten Bus Stops assigned along the Bus Route.

Bus Stop	Name	Longitude	Latitude
S01	UTAR Sungai Long Bus Stop	3.039519	101.794555
S02	Komersial BT 11	3.042469	101.772179
S03	Aeon Cheras Selatan (Opp)	3.032193	101.766019
S04	Klinik Taming	3.024327	101.730146
S05	KTM Serdang	3.023479	101.716029
S06	The Mines	3.026984	101.718918
S07	KG. Baru Belakong	3.026502	101.749507
S08	Shell Sungai Long	3.041762	101.789526
S09	Sungai Long Golf & Country	3.042027	101.801568
S10	UTAR KB Block Entrance	3.039751	101.794424

The route is generated by Google Map by providing the origin and destination. The Maps JavaScript API displayed the customized map in the web application while the Distance API retrieves the route data from Google Maps. The generated bus route with ten bus stations is shown in Figure 3.1 below:

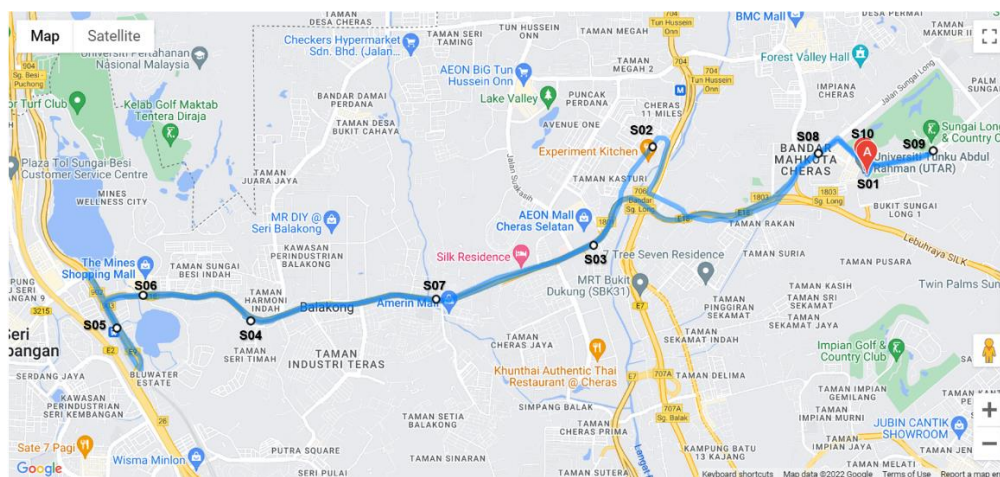


Figure 3.1: Bus route of UTAR bus generated with Google Maps.

### 3.3.2 ETA Data

ETA data refers to the estimated travel duration between stations. In the DRL model, the bus is only allowed to select between no skipping, skipping one station, or skipping two stations in order to optimize the route. For example, assuming the bus departs from station  $j$ , it can only stop at station  $j+1$ ,  $j+2$ , or  $j+3$  depending on the chosen action. The ETA data can be obtained from Google Maps Platform by using the Distance Matrix API, which requires the origin-destination pair and departure time to obtain the related ETA data. The data was collected between 7:00 AM and 6:30 PM and stored in an Excel file, as shown in Figure 3.2, with the results being stored in seconds. In summary, ETA data for 15 days was collected and stored as ten training sets and five testing sets for model training and evaluation purposes.

Origin	1		2		3		4		5		6		7		8		9							
Destination	2	3	4	3	4	5	6	5	6	7	6	5	8	7	6	9	8	9	10	9	10	10		
07:00:00	933	1168	1535	236	581	837	340	596	737	349	475	392	256	456	889	285	717	1070	479	832	826	350	329	231
07:30:00	851	1083	1423	225	575	827	364	604	736	371	494	399	246	465	894	287	714	1080	479	845	819	352	328	219
08:00:00	830	1061	1413	222	602	827	336	587	715	345	483	373	244	472	892	270	695	1051	478	834	808	334	298	231
08:30:00	850	1067	1414	227	569	836	332	611	767	367	519	388	254	474	910	274	708	1053	469	814	792	323	315	227
09:00:00	807	1001	1350	204	533	909	347	606	772	361	509	388	264	477	954	279	743	1102	493	852	832	348	356	247
09:30:00	786	977	1345	200	580	1112	360	791	987	497	693	506	286	533	1090	293	832	1219	585	972	960	330	336	237
10:00:00	825	1016	1416	195	555	1120	358	978	1226	715	961	846	322	654	1346	333	1018	1425	690	1088	1065	398	376	255
10:30:00	884	1080	1421	183	579	1054	391	943	1203	695	955	784	454	780	1493	444	1177	1559	732	1115	1067	374	342	250
11:00:00	852	1050	1372	208	535	848	320	694	871	461	665	554	352	641	1328	372	1059	1407	751	1096	1063	385	345	249
11:30:00	841	1041	1365	208	531	802	324	603	756	357	563	394	347	564	1137	310	890	1213	679	1002	964	366	353	251
12:00:00	779	981	1319	197	549	782	347	586	684	345	427	392	284	485	982	308	807	1129	592	914	914	340	321	238
12:30:00	767	952	1274	181	516	752	340	572	653	336	414	385	272	461	894	296	731	1064	525	857	828	360	336	229
13:00:00	728	898	1229	186	513	774	326	580	609	334	402	383	264	418	837	301	721	1018	478	774	759	330	301	225
13:30:00	721	893	1217	175	491	735	321	562	584	335	354	371	220	402	793	285	678	937	456	715	700	332	300	227
14:00:00	705	882	1196	176	493	728	318	552	573	327	333	364	203	385	777	277	669	935	438	705	696	384	245	216
14:30:00	693	860	1147	165	455	695	290	532	525	315	300	351	179	381	756	270	646	949	430	733	719	283	270	195
15:00:00	655	813	1102	161	445	684	284	519	503	302	274	345	150	361	728	259	627	894	412	674	673	271	245	179
15:30:00	698	857	1132	152	430	696	278	519	498	300	277	328	164	358	715	246	604	858	405	652	634	226	224	187
16:00:00	567	719	984	141	426	659	285	529	495	305	288	338	169	354	702	253	596	804	392	611	614	236	236	178
16:30:00	542	684	970	140	422	653	281	521	494	288	266	324	151	346	690	244	589	795	381	604	594	223	226	210
17:00:00	497	627	911	146	451	642	304	511	490	305	297	333	156	339	662	227	553	741	361	554	542	204	204	181
17:30:00	486	623	883	150	427	639	276	496	483	270	259	308	143	339	660	217	533	726	330	533	524	204	219	181
18:00:00	506	645	915	152	425	633	272	498	469	278	264	316	141	335	646	222	539	745	348	530	530	175	182	169
18:30:00	485	618	904	152	434	637	281	484	455	250	236	311	117	337	646	220	534	753	338	499	504	160	166	155

Figure 3.2: Stored ETA Data in Excel File.

### 3.4 Bus Optimization Scheme

There are  $N$  bus stations along the route and each station can be denoted as  $j = 1, 2, \dots, N$ . Figure 3.3 shows the overview of the bus optimization scheme. The bus stop-skipping scheme starts with express bus  $i$  and is followed by a no-skip bus  $i+1$ . The bus  $i$  allowed to skip one or two stops while the next bus  $i+1$  needs to stop at every stop to avoid passengers stranded at the bus stop for a long period. Each bus must depart from the first station and terminate at last station  $N$ .

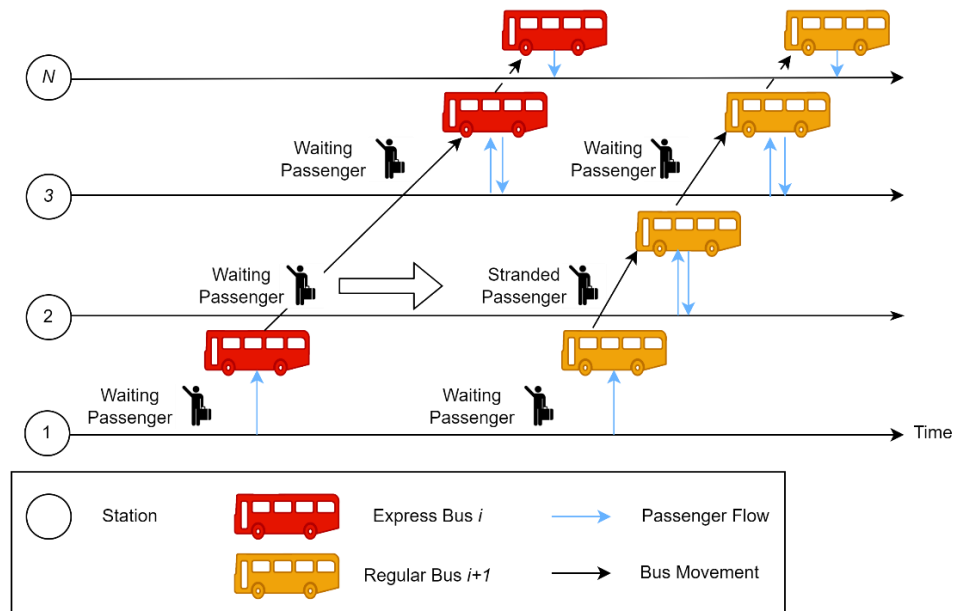


Figure 3.3: Bus Optimization Scheme.

The bus stop-skipping scheme is set to repeat every 30 minutes from 7:00 AM until 6:30 PM. The express bus departs first from the first station, while the regular bus departs 15 minutes later. Assuming the express bus has



departed from previous station  $j-1$  and reached station  $j$ , the ETA between the stations is obtained from the Distance Matrix API and denoted as  $t_{j-1, j}$ . At station  $j$ , the time of arrival  $A_{i,j}$  of bus  $i$  can be expressed as follows:

$$A_{i,j} = D_{i,j-1} + t_{j-1,j}. \quad (3.1)$$

As the bus reaches station  $j$ , passengers usually alight from the backdoor of the bus, while waiting passengers at station  $j$  board the bus. Let  $U_{i,j}$  denotes the passengers count that boarding bus  $i$  at station  $j$ , while  $V_{i,j}$  denotes the passengers count that alighting from bus  $i$ . The duration for passengers to board and alight is known as dwell time. The average time for boarding the bus is 5 seconds, whereas the average time for alighting is 3 seconds. The dwell time  $\tau_{i,j}$  is equal to the maximum time for boarding or alighting:

$$\tau_{i,j} = \max(5 \cdot U_{i,j}, 3 \cdot V_{i,j}). \quad (3.2)$$

As bus  $i$  picks up all passengers at station  $j$ , the bus starts heading to station  $k$ , where  $k$  can be station  $j+1, \dots, N$ . The departure time of bus  $i$  at station  $j$  is equal to the time of arrival plus the dwell time:

$$D_{i,j} = A_{i,j} + \tau_{i,j}. \quad (3.3)$$

The subsequent regular bus  $i+1$  will arrive at station  $j$  after a certain amount of time. The time interval between the departure of bus  $i$  and the arrival of bus  $i+1$  is known as the headway. The headway  $H_i$  for bus  $i$  and the previous bus at every station is fixed at 15 minutes, while the headway between bus  $i+1$  and bus  $i$  is expressed as:

$$H_{i+1,j} = A_{i+1,j} - D_{i,j}. \quad (3.4)$$

At station  $j$ , the total count of passengers waiting for bus  $i$  is denoted as  $W_{i,j}$ . It is generated using the arrival rate  $\lambda_j$ , where different passengers will

have different destination stations. The arrival rate  $\lambda_{j,k}$  represents how often passengers arrive at station  $j$  with the intention of heading towards station  $k$ . The count of passengers  $W_{i,jk}$  waiting for bus  $i$  at station  $j$  and intended to go to station  $k$  is formulated as:

$$W_{i,jk} = \lambda_{j,k} \cdot H_{i,j}. \quad (3.5)$$

The load of bus  $i$  denotes as  $l_i$  and is the sum of boarding passengers on the bus  $i$ . When passengers board the bus, the count of boarding passengers  $U_{i,j}$  is equal to the total number of waiting passengers  $W_{i,j}$  at station  $j$ . The total number of alighting passengers is the sum of passengers who boarded the bus at the previous station  $n$  and intended to alight at station  $j$ , as expressed by Equation 3.6. Equation 3.7 shows the how the bus load is updated with the boarding and alighting passengers at each station  $j$ .

$$V_{i,j} = \sum_{n=1}^{j-1} U_{i,nj} \quad (3.6)$$

$$l_{i,j} = l_{i,j-1} + U_{i,j} - V_{i,j} \quad (3.7)$$

where

- $V_{i,j}$  = passengers alight at station  $j$
- $U_{i,j}$  = total number of passengers board at station  $j$
- $l_{i,j-1}$  = the load of bus  $i$  at previous stopped station  $j-1$

The express bus will skip some stations along the bus route. Assume that bus  $i$  skipped station  $j$ , leaving passengers stranded at station  $j$ .  $L_{i,jk}$  is the number of passengers left behind by bus  $i$  at station  $j$  who intended to go to station  $k$ . The subsequent bus  $i+1$  will serve these passengers at station  $j$  to avoid long waiting times. The travel duration between stations can be obtained from Google Maps Platform, which considers real-time traffic conditions. The same travel time is implemented for both bus  $i$  and bus  $i+1$ .

### 3.5 Deep Reinforcement Learning Model Framework

In this project, the DRL model is utilized to optimize the route selection of an express bus in a bus optimization scheme, considering traffic conditions and passenger demands. The bus is required to follow a set time schedule and terminate service at the last station. To select an action (no skip, skip one station, or skip two stations), the agent analyzes the environment using the state space at time  $t-1$ , which includes important information such as waiting passengers and ETA between stations. If the agent selects to skip two stations, the bus performs the action by departing from the current station  $j$  and stopping at station  $j+3$  with a duration of  $t_{j,j+3}$ . The decision made by the agent is evaluated using the reward function and next state space at time  $t$ . The simulation continues until both the express bus and no-skip bus arrive at the last station. Figure 3.4 depicts the framework of the DRL model.

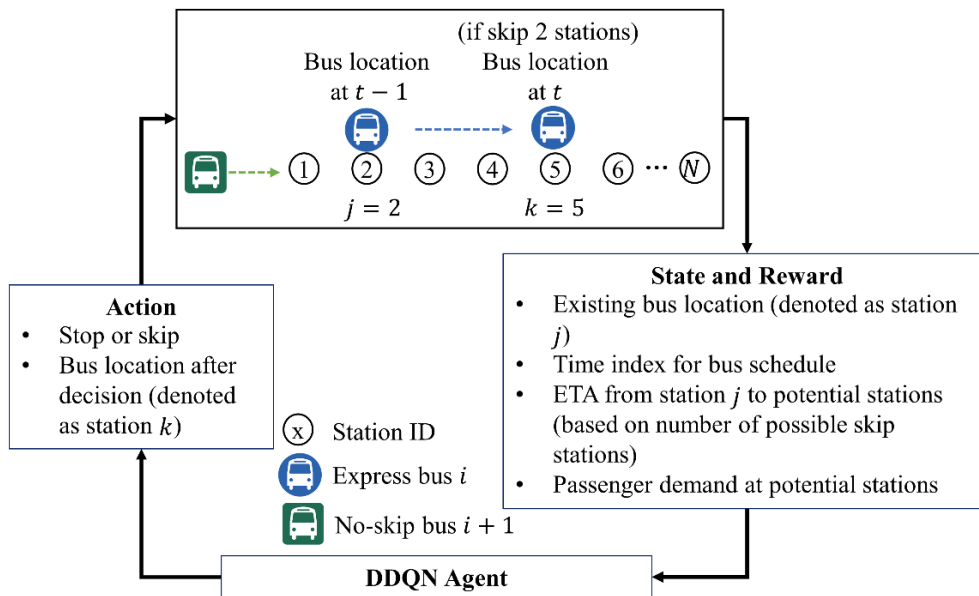


Figure 3.4: DRL Model Framework.

The dynamic bus environment in the DRL model simulates the system model design outlined in Section 3.4. The optimization scheme for the bus operates every 30 minutes, with the express bus departing from the first station and the second no-skip bus departing from the first station after 15 minutes. During the simulation, the DRL model makes optimal slot-by-slot decisions, with the decision point being when the bus is about to depart from

the current station. The agent makes decisions based on its experience and the state space received from the bus environment. Passenger demand and ETA between stations are considered by the DRL model when selecting the route, with ETA data collected from the Google Maps Platform using the Distance Matrix API and passenger demand generated using the passenger demand model.

### 3.5.1 Passenger Demand Model

The passengers flow at each station  $j$  is generated using the arrival rate  $\lambda_j$  and the interval between the buses,  $H_{i,j}$ , as described in Equation 3.5. The arrival rate at station  $j$  comprises various passengers who are anticipated to be traveling towards station  $k$ , denoted as  $\lambda_{j,k}$ . The alighting rate at each station is fixed as per Table 3.2:

Table 3.2: Alighting Rate for Each Station along The Route.

Station	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10
Alighting Rate	0	0.7	0	0.5	0	0.8	0.5	0	0	1

The arrival rate at each station is adjusted with the boarding rate and alighting rate to generate different types of passengers demands. Three different passenger demand scenarios can be generated with this model:

- Static passenger demand,
- Dynamic passenger demand varied with time,
- Random passenger demand.

The first scenario has a static boarding rate as shown in Table 3.3. In the second scenario, the boarding rate at four stations (S01, S02, S05, and S07) is randomly varied over time to be higher than the boarding rate at the other stations, which is set to be lower. In the third scenario, the arrival rate of passengers is randomly assigned without adjusting for the boarding rate. The passenger arrival rate is randomly generated within the range of 0.1 to 0.4 for all three scenarios.

Table 3.3: Boarding rate for each station along the route.

Station	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10
<b>Boarding Rate</b>	1	1	0	0	0	1	0	0	0	0

### 3.5.2 State Space Design

In the DRL model, the state space comprises environment-related data that helps the agent understand its surroundings. In this project, one of the most crucial pieces of information that must be included in the state space is the station ID, which indicates the current station where the bus has stopped. The starting point of the dynamic bus environment is denoted as station ID 0, and each station is represented by an ID number ranging from 0 to 9.

Another important piece of information that needs to be included in the state space is the scheduled departure time of the bus, particularly in a dynamic passenger demand scenario. The bus operates from 7:00 AM to 6:30 PM, and the schedule is divided into 24 time indices. For instance, a time index of 0 represents the bus's departure from the first station at 7:00 AM.

Additionally, it is necessary to include the travel duration of the bus to reach the next three stations in order for the agent to select the best action. Since the agent can only choose between not skipping any stations, skipping one station, or skipping two stations, the ETA data can help the agent predict the appropriate action while minimizing the operating cost. The ETA data is converted to minutes and stored in the state space.

Finally, the count of passengers waiting at the next three stations is also incorporated into the state space to train the agent to halt at stations with a higher demand. In summary, eight parameters are saved in the state space to aid the agent in comprehending the environment.

### 3.5.3 Reward Function Design

The RL model is designed to effectively distribute passenger demand while minimizing costs for both bus operators and passengers. The reward function plays a crucial role in shaping the agent's behaviour. An appropriately designed reward function can define the agent's goal and provide feedback on

its performance. The agent learns through reward-punishment mechanisms, with rewards given to encourage correct decisions that lead to the desired outcome, and penalties given when undesired behaviours occur.

Assume that the bus  $i$  locates at station  $j$  in the dynamic bus environment and the agent decides to skip station  $k-1$  and heading at station  $k$ . To optimize the stop-skipping service of the express bus, the reward function considered the following factors:

- Passenger demand: The reward should be given when the DRL model meets the demand of passengers. The step reward is proportional to the number of passengers alighting and boarding at each station, which can be formulated as:

$$P_{i,k} = U_{i,k} + V_{i,k}. \quad (3.8)$$

- Waiting time: Passengers prefer shorter waiting times at bus stops. The DRL model should be rewarded when reducing the passengers' average waiting time at each station. The formula for the average waiting time of passengers can be expressed as:

$$\overline{T_{w,k}} = \frac{H_{i,k}}{2} \quad (3.9)$$

- In-vehicle time: Passengers prefer shorter travel times on the bus. The reward should increase when the in-vehicle time decreases. The in-vehicle time between station  $j$  and  $k$  for passengers on bus  $i$  includes the dwell time and ETA between stations. The load of bus  $i$  is normalized with the maximum load of the bus,  $l_{max}$  to scale the in-vehicle time and avoid making it too large. The formula to calculate the in-vehicle time is:

$$T_{v,jk} = (\tau_{j,k} + t_{j,k}) \cdot \frac{l_{i,j}}{l_{max}} \quad (3.10)$$

- Bus running time: Operators prefer shorter running times between stops. The DRL model should be rewarded when reducing the

operating time of the bus. Equation 3.11 shows the formula to calculate the running time of the bus between station  $j$  and  $k$ :

$$T_{r,jk} = (\tau_{j,k} + t_{j,k}) \quad (3.11)$$

- Number of stranded passengers: The penalty should be given when the DRL model skips a station and passengers are stranded at the station  $k-1$ . The number of stranded passengers, denoted as  $L_{i,k-1}$ , is utilized calculate the penalty.
- Waiting time of stranded passengers: Passengers who are left stranded at a station due to the bus skipping can experience longer waiting times, which can negatively impact their experience and satisfaction. The DRL model should be punished for the long average waiting time of stranded passenger. The formula for the average waiting time of stranded passengers at station  $k-1$  is expressed as:

$$\overline{T_{ws,k-1}} = \frac{H_{i,k-1}}{2} + H_{i+1,k-1} \quad (3.12)$$

Since the bus optimization scheme involves one express bus and one no-skip bus, the reward function is split into two parts to capture rewards and penalties from both the first express bus trip and the second no-skip bus trip. Boarding demand represents the actual passenger flow at bus stops, while alighting demand represents the demand of passengers alighting at specific stops. When the agent decides to stop at a station with higher demand, the reward should be designed to be higher. The average waiting time and normalized in-vehicle time should be reduced to improve passenger satisfaction. These parameters should be inversely proportional to the reward given. The operating time of the bus is included to consider the bus operator's expenditure. The reward should decrease as the running time increases. In addition to passenger demand, in-vehicle time, average waiting time, and bus running time, a penalty of 0.3 is deducted from the reward to represent the constant operating cost of the bus operator. The step reward function for the first bus trip is:

$$R_{t1} = \frac{C_1 P_{i,k}}{C_2 \bar{T}_{w,k} + C_3 T_{v,jk} + C_4 T_{r,jk}} - 0.3. \quad (3.13)$$

The weighted values for each metric are represented by  $C_1$ ,  $C_2$ ,  $C_3$  and  $C_4$ . In Equation 3.13, the values of  $C_1$ ,  $C_2$  and  $C_3$  are set to 1, 2 and 1.3 respectively. The value of  $C_4$  is defined by the total number of stations stopped by bus  $i$  before arriving station  $j$  with a weight of 0.4. While the no-skip bus will pick up the stranded passenger at station  $k-1$ , the total leftover passenger and average waiting time is included for the step reward function in second bus trip with  $C_5 = 0.3$ :

$$R_{t2} = C_5 \frac{L_{i,k-1}}{T_{ws,k-1}}. \quad (3.14)$$

Equation 3.15 employs an extra reward constraint,  $R$ , which serves to either incentivize or penalize the agent's decision to reach the terminal station. If the bus successfully reaches the terminal station, the DRL model is rewarded. However, if the bus fails to reach the terminal station  $N$  within the maximum steps allowed in an episode, a penalty is imposed. Equation 3.16 shows the overall step reward function in an episode.

$$R = \begin{cases} 5 & , \text{ if bus } i \text{ reached station } N \\ -5 & , \text{ if bus } i \text{ unable to reach station } N \end{cases} \quad (3.15)$$

$$R_t = \begin{cases} R_{t1} + R_{t2} \\ R_{t1} + R_{t2} + R & , \text{ if } done = True \end{cases} \quad (3.16)$$

To summarize, the reward function is specifically designed to motivate the agent to minimize passenger waiting time, reduce in-vehicle travel duration, and decrease the overall length of the bus trip. Simultaneously, it aims to satisfy the high passenger demand for both boarding and alighting from the bus. The function also penalizes the agent for actions that result in stranded passengers, extended waiting times, or excessive operating costs.



### 3.5.4 Double Deep Q-Network Agent

The DDQN agent employs the epsilon-greedy approach to choose actions based on the state space, which balances exploration and exploitation rates. Exploration involves the agent taking random actions to gain a deeper understanding of the environment, while exploitation involves the agent selecting actions based on the Q-network's weights and biases. The trade-off between these two is crucial for the agent to understand the environment and choose the best action based on the state. Figure 3.5 depicts the agent's workflow for selecting actions using the epsilon-greedy method.

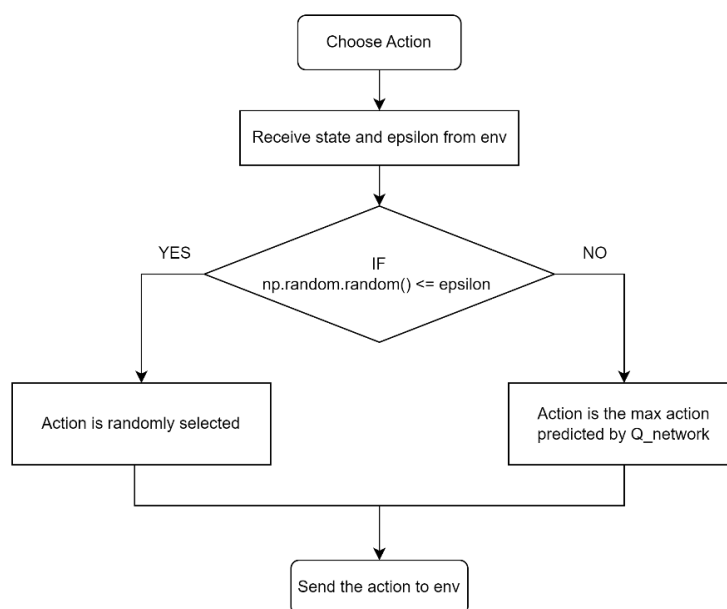


Figure 3.5: Workflow of action selection in DDQN Model.

To maintain exploration during training, the epsilon value starts to decay after the data stored in the experience replay exceeds 2000, with a decay rate of 0.999 in the DDQN agent. The agent's experiences obtained over multiple episodes are stored in the replay memory, and the state and reward obtained from the environment are used to evaluate the action selection of the agent. When the number of stored data exceeds 2000, the memory is randomly sampled to create a minibatch of experiences, which are then used to update the Q-network's weights and biases using the DDQN approach.

The sampled state, next state, action, reward, and done status data are stored separately to calculate the Q-value for both the Q-network and target network in the DDQN agent. The target network is used to obtain the actual Q-

value based on the action selection in the Q-network. The agent then calculates the difference between the Q-value predicted by the Q-network and the actual Q-value obtained from the target network. This difference is used to update the Q-network's weights through backpropagation. To stabilize the learning process, the weights of the Q-network are periodically synchronized with the target network. The Q-value is updated directly with the reward in the corresponding state when the done status is true. When the done status is false, the Q-value of the Q-network at time step  $t$  is updated using the formula given below:

$$Q(s_t, a_t) \leftarrow \begin{cases} R_t + \gamma Q_{target}(s_{t+1}, \arg \max_a Q(s_{t+1}, a; \theta), \theta) \\ R_t, \end{cases} \quad \text{if } done = True \quad (3.17)$$

The architecture of the DDQN agent can be visualized in Figure 3.6. The input layer of the DDQN agent receives the current state of the environment, which consists of eight features at each time step. The Q-network, a neural network with hidden layers, processes the state information and predicts the expected rewards for each of the three possible actions using a softmax function. However, since the Q-network's parameters are updated during training, it can lead to instability and overestimation of the expected rewards. Therefore, the target network is used to estimate the true expected rewards. The target network's parameters are kept fixed and updated periodically using the Q-network's parameters. By comparing the estimated rewards of the Q-network and target network, the DDQN agent learns to select the optimal action at different states of the environment by maximizing the reward. This approach reduces the overestimation bias and results in more stable and efficient training.

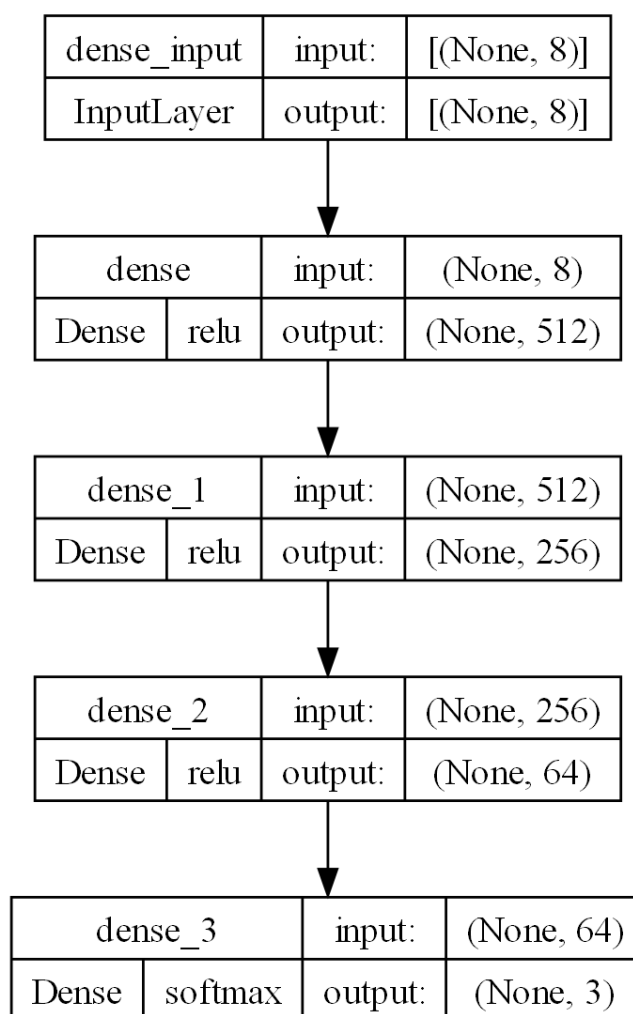


Figure 3.6: Architecture of the DDQN agent.

## 3.6 Model Training

### 3.6.1 Training Environment

This project utilizes TensorFlow version 2.9.1 to implement the DDQN agent for model training. The initial training was carried out on Google Colab. However, given the extended training period required to generate results, it was deemed impractical to continuously monitor the platform due to idle activity. Consequently, the decision was made to conduct the training on a laptop with the hardware specifications presented in Table 3.4. Given the lengthy duration required for the training process to produce the desired results, this hardware configuration has been determined to be adequate for the task at hand. Although the laptop is equipped with a GPU, the training process in this project solely utilizes the CPU.

Table 3.4: Hardware Specification of Training Environment.

Parameters	Values
CPU Model	Intel® Core™ i5-1020U
CPU Frequency	2.11 GHz
Number of CPU Cores	4
RAM Available	12 GB
GPU	NVIDIA GeForce MX230
GPU Memory	7.9 GB

### 3.6.2 Training Workflow

The DRL model is trained using ten sets of ETA data obtained from the Google Maps Platform, with 5000 training episodes. In each episode, the express bus trip is executed first, followed by the no-skip bus trip. The DDQN agent makes slot-by-slot decisions based on the state. For the first bus trip, the reward is calculated based on passenger demand, in-vehicle time, waiting time, and total bus operating time. The state, next state, action, reward, and done status data are collected at each step and stored separately in lists. The overall simulation flow of the first bus trip is illustrated in Figure 3.7.

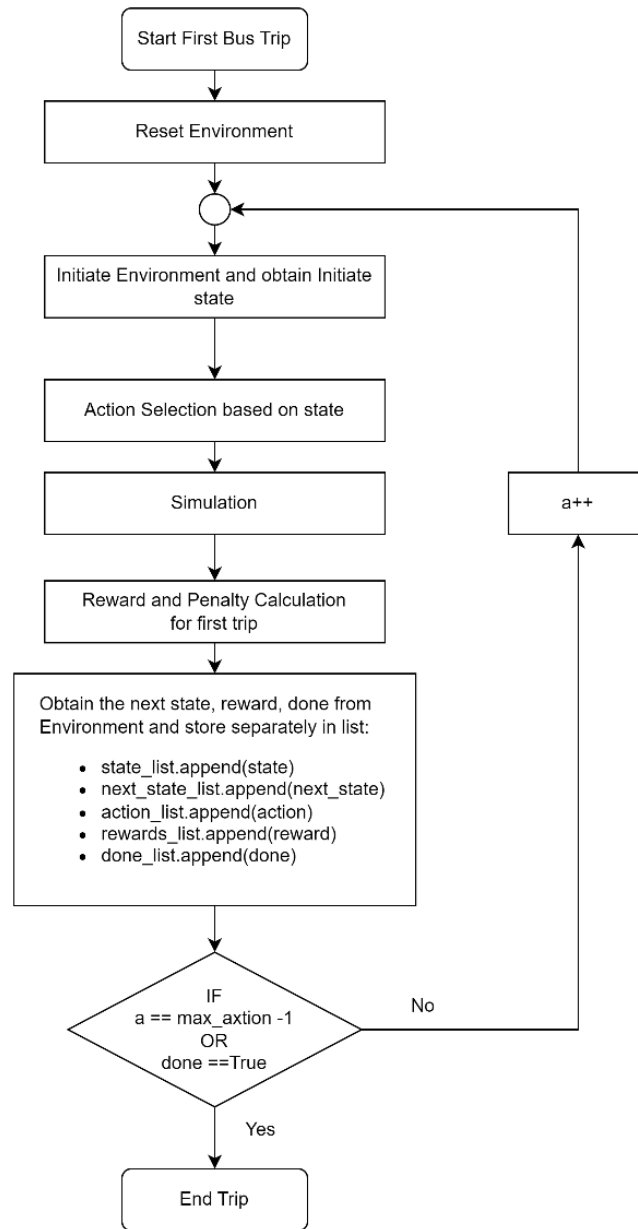


Figure 3.7: Simulation Flow of First Bus Trip.

Similarly, for the second bus trip, the simulation is similar to the first bus trip, but the reward is calculated based on the stranded passengers and their waiting time. Since the action for this trip must be no-skipping, the agent does not need to make any decisions. Figure 3.8 illustrates the simulation flow of the second bus trip.

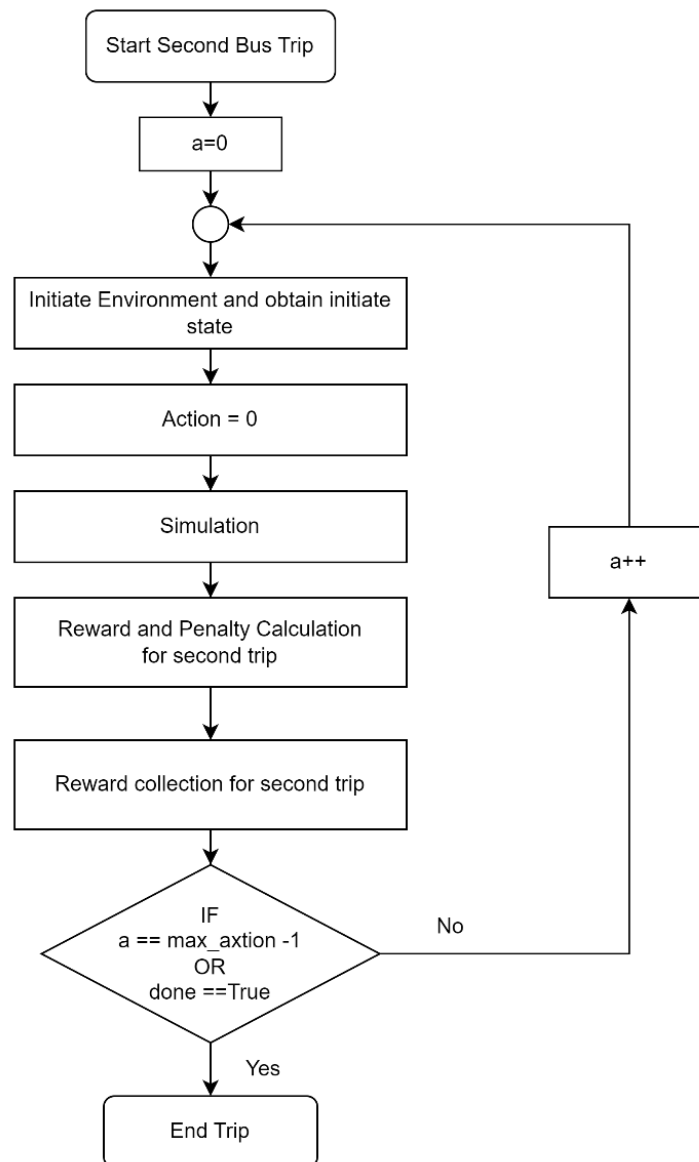


Figure 3.8: Simulation Flow of Second Bus Trip.

Figure 3.9 shows the simulation in an episode, where the reward from the first and second bus trips are combined. The reward for the second trip is interpreted as a penalty for the skip action due to passenger dissatisfaction. The total reward of an episode is equal to the difference between the rewards of the first and second bus trips. The state, next state, action, done status data, and combined reward are stored in the replay memory and used for experience replay at the end of the episode.

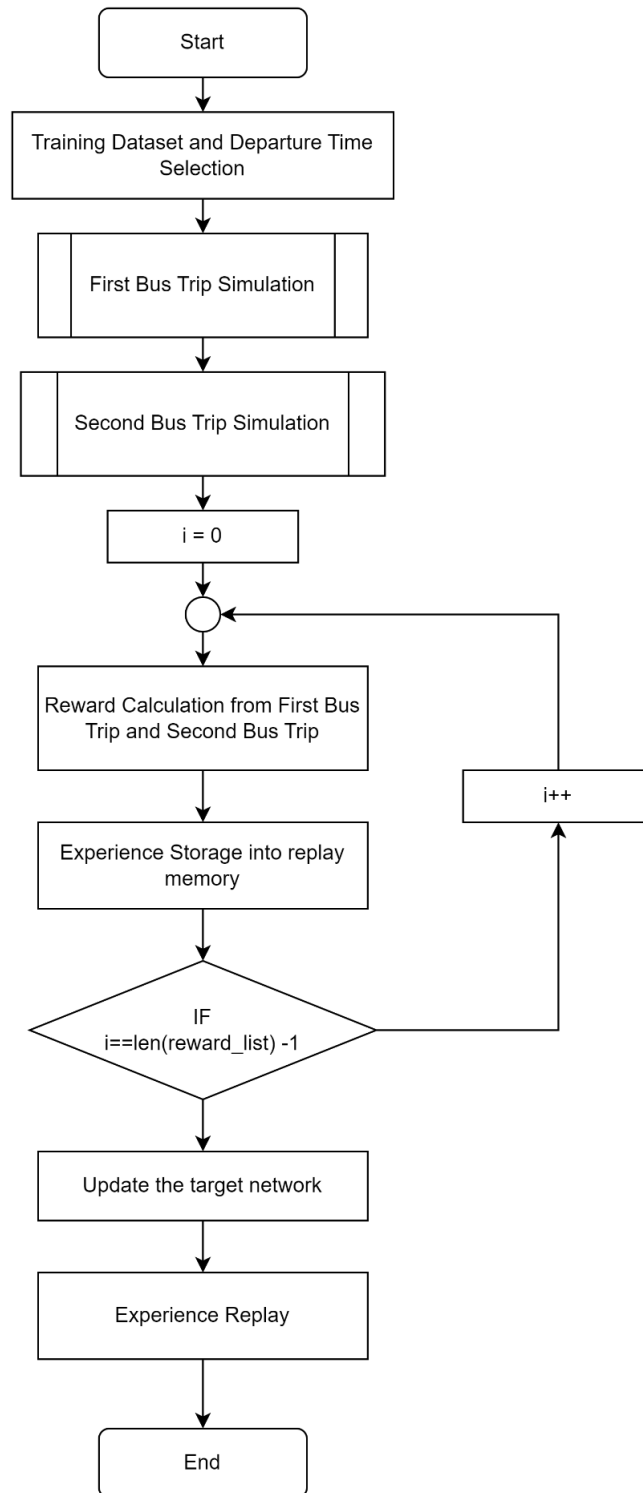


Figure 3.9: Simulation of Model Training in an Episode.

### 3.7 Model Evaluation

The evaluation stage involves comparing the performance of the DRL model with the conventional approach, using the designed reward function. Five sets of ETA data are used to simulate the dynamic bus environment for both

DDQN and conventional approaches. Each approach involves two bus trips in the bus optimization scheme. The rewards from the first and second bus trips for each approach are combined and compared using the reward function as the evaluation metric. The reward function includes factors such as passenger demand, in-vehicle time, total bus trip duration, stranded passengers, and waiting time. This enables a comprehensive analysis of the performance of the two approaches in terms of optimizing benefits for both passengers and operators.

### 3.8 Model Deployment in Web Application

The Flask is integrated with the Maps JavaScript API and developed into a web application. The web application allows the user to select the passenger demand mode and departure time to perform the simulation on the live map. The DRL model is deployed in a Python script and generates the route according to the departure time, real-time ETA data, and passenger demand mode. Real-time ETA data can be obtained using the Distance Matrix API. The primary function of the web application is to simulate the step-by-step route selection process made by the DRL model.

### 3.9 Gantt Chart

Figure 3.10 and Figure 3.11 illustrate the project timeline using two Gantt charts.

No.	Project Activities	Planned Completion Date	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14
1.	Problem Statement and Project Planning	2022-06-26	■	■												
2.	Research on the Google Maps API	2022-07-08		■	■	■										
3.	Integrating the Google Maps API with Python and JavaScript	2022-08-12				■	■	■	■	■	■					
4.	Research on Deep Reinforcement Learning	2022-09-02							■	■	■	■	■	■		
5.	Data Gathering and Report Writing	2022-09-12								■	■	■	■	■	■	■

Figure 3.10: Gantt Chart of Project Timeline for the June 2022 Semester.



No.	Project Activities	Planned Completion Date	Weeks														
			W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	
1.	Research on deep reinforcement learning in stop-skipping operation	2023-02-17	█	█	█												
2.	Reinforcement Learning Model Framework Design	2023-02-24	█	█	█	█											
3.	Reinforcement Learning Model Building - Bus Environment - DDQN Agent - Reward Function	2023-04-14		█	█	█	█										
4.	Model Testing and Model Refinement	2023-04-21			█	█	█	█	█	█	█	█	█	█	█		
5.	Model Training and Model Evaluation	2023-04-28							█	█	█	█	█	█	█	█	
6.	Passenger Demand Model Development	2023-04-07								█	█	█					
7.	Model Deployment in Web Application	2023-04-28										█	█	█	█	█	
8.	Final Report Writing	2023-05-01											█	█	█	█	█

Figure 3.11: Gantt Chart of Project Timeline for the January 2023 Semester.

### 3.10 Summary

This chapter mainly focuses on explaining the methodology and work plan for the entire project. The DRL framework is described clearly, with its important components which include environment, agent, state space, action, and reward function. Designing the reward function was the most challenging task in this project due to the need to consider various factors such as bus operating time, passenger in-bus time, passenger waiting time and passenger demand. The process of model training and evaluation was performed using the ETA datasets collected from Google Maps Platform. Lastly, the model was successfully deployed in a web application with Google Maps embedded.

## CHAPTER 4

### RESULTS AND DISCUSSION

#### 4.1 Introduction

The training results of DRL model for three different passenger scenarios were analyzed and compared. Furthermore, the performance of DDQN and conventional approaches was evaluated according to the reward function designed. Lastly, the model was deployed in the web application to simulate the step-by-step route selection process made by the DRL model.

#### 4.2 Results of Model Training

In this section, the training results for three different passenger demand scenarios is shown. Ten sets of ETA dataset from Google Maps Platform are provided model training. The DRL model was trained with 5000 episodes and its parameter settings is shown in Table 4.1:

Table 4.1: The parameter settings in DRL model.

Parameters	Values
Number of stations	10
Frequency of express (no-skip) trip per trip	30 minutes
Dispatch interval between express and no skip trip	15 minutes
Learning rate	0.0005
Activation function	ReLU
Number of input nodes	8
Number of hidden layers	3
Number of hidden nodes	832
Number of output nodes	3
Batch size	32
Discount rate	0.4
Experience memory size	5000
Epsilon Decay	0.999

The discount factor is set to 0.4 to make the agent consider the immediate reward more heavily. The DDQN agent is comprised of three hidden layers which utilize the ReLU activation function, while the output layer adopts the Softmax function to generate the probability distribution for the three potential actions.

#### 4.2.1 Training Result of Scenario 1

Figure 4.1 illustrates the training performance of DDQN agent over 5000 episodes for static passenger demand. The training performance reached converged as the number of episodes grew.

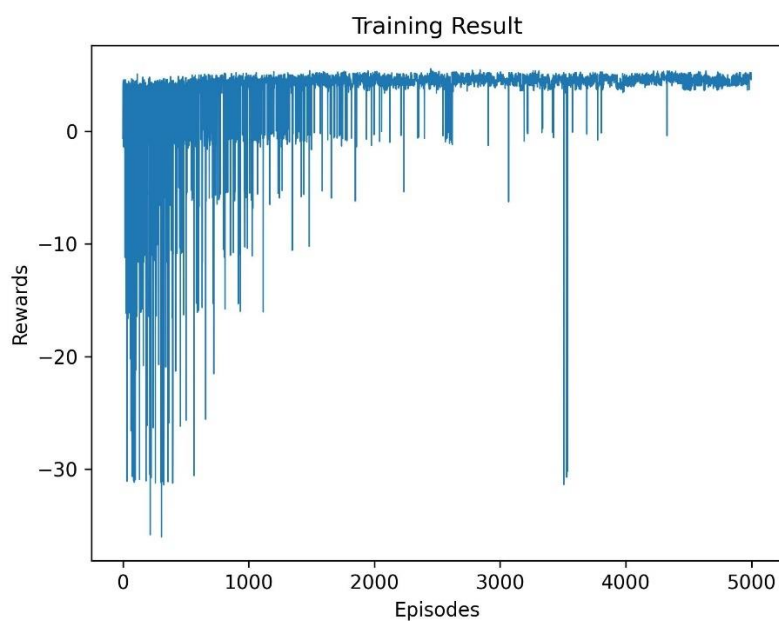


Figure 4.1: Training Results over 5000 episodes for Scenario 1.

#### 4.2.2 Training Result of Scenario 2

In Scenario 2, the passenger demand varies dynamically depending on the time of day. Figure 4.2 illustrates the training results of the DDQN agent for this scenario, demonstrating its ability to make stable decisions over time. The agent was able to select the optimal route based on the fluctuating passenger demand and ETA between stations. The convergence of the results indicates the effectiveness of the DDQN agent in handling dynamic passenger demands while minimizing bus operating costs.

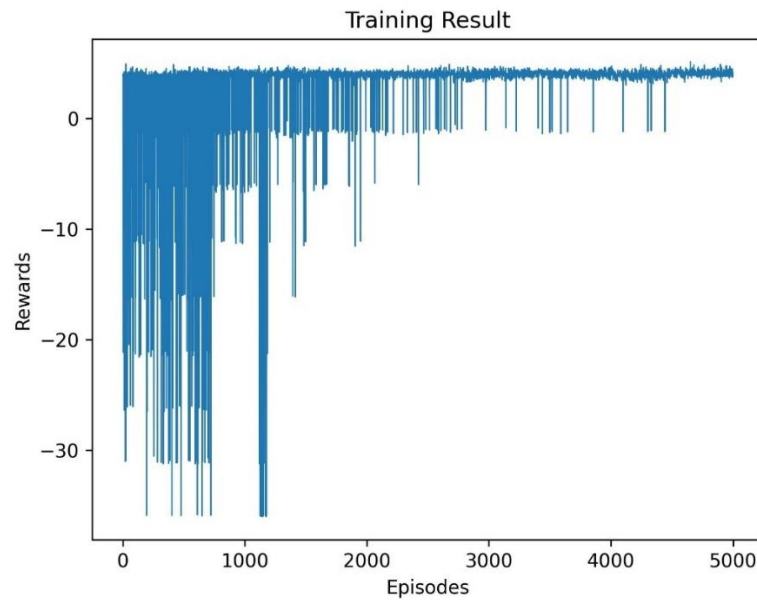


Figure 4.2: Training Results over 5000 episodes for Scenario 2.

### 4.2.3 Training Result of Scenario 3

The passenger demands at each station are randomly assigned in Scenario 3. Figure 4.3 displays the training results of the DDQN agent for this scenario, where the training performance converged as the number of episodes increased.

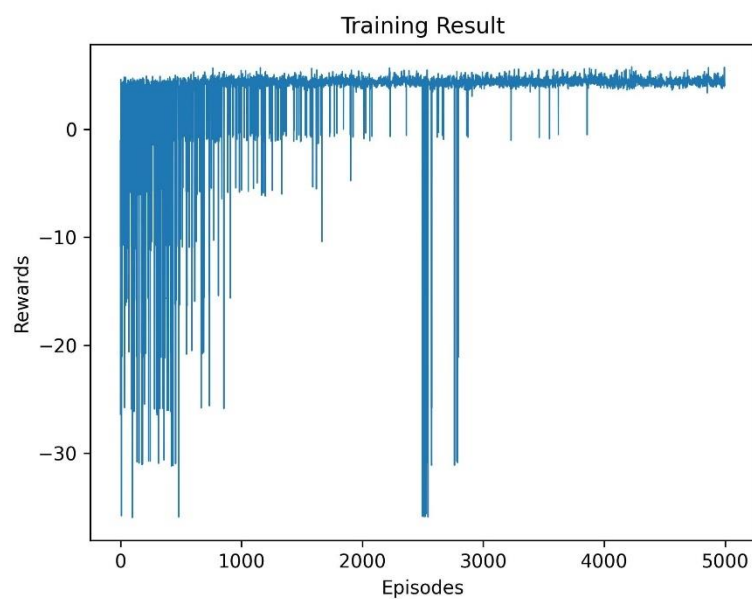


Figure 4.3: Training Results over 5000 episodes for Scenario 3.

#### 4.2.4 Loss of Model Training

The training loss measures the disparity between the predicted Q-value and the true Q-value. Figure 4.4 shows the loss of the DDQN agent in all three scenarios, where it can be observed that the loss decreased with an increasing number of episodes. This indicates that the DDQN agent is approaching the optimal set of weights and biases, which will enable it to make accurate predictions in different environments.

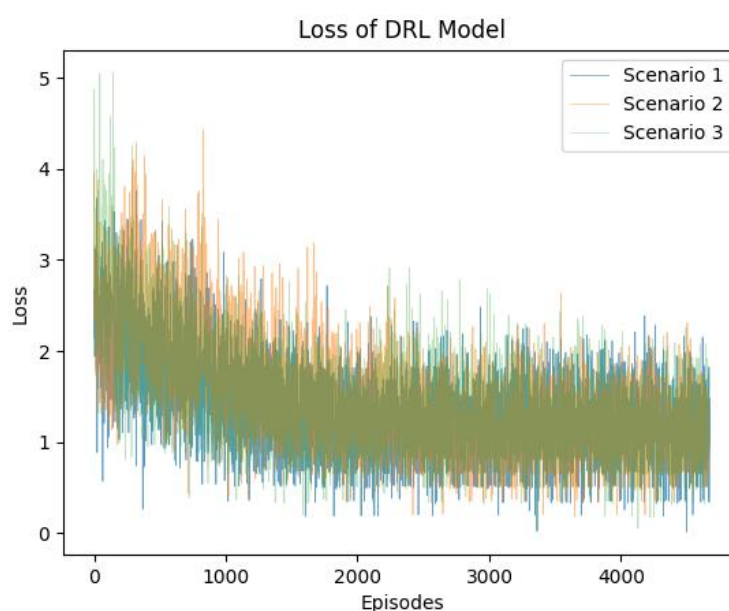


Figure 4.4: Loss of Model in all Three Scenarios.

### 4.3 Results of Model Evaluation

In this study, the evaluation results of the DDQN and conventional methods are compared for three different passenger demand scenarios. The performance of the DRL model with different ETA data obtained from the Google Maps Platform is evaluated using five sets of data. Both the DDQN and conventional approaches are evaluated using a designed reward function.

#### 4.3.1 Evaluation Result of Scenario 1

According to the results illustrated in Figure 4.5, the evaluation result showed that the DDQN approach performed better than the conventional method.

Table 4.2 shows the average reward of both DDQN and conventional approaches over 1000 episodes.

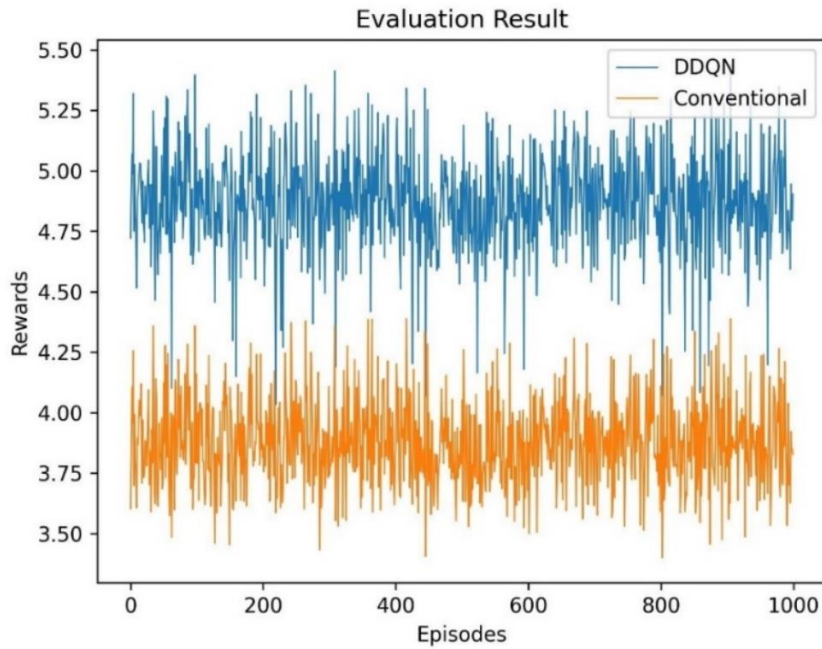


Figure 4.5: Evaluation Result of Scenario 1.

Table 4.2: Average Evaluation Score of DDQN and Conventional Approaches in Scenario 1.

Approach	DDQN	Conventional
Average Score	4.873	3.879

Table 4.3 illustrates how the DDQN agent makes decisions based on the boarding and alighting demands of passengers at each station. It is evident that the agent is capable of stopping at stations with higher passenger demand. In some instances, however, the agent will choose to skip a station with high demand when the ETA between the stations is increased due to traffic conditions. The DDQN approach obtained a higher score than the conventional method, indicating that the DRL model is better equipped to handle static passenger demand efficiently. This is likely due to the reward function being designed to prioritize factors such as the time passengers spend waiting and traveling, bus operating time, and passenger demand.

Table 4.3: Skip-stop pattern of DDQN Agent in Scenario 1.

Stop	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10
<b>Boarding Demand</b>	12	8	0	0	0	3	0	0	0	0
<b>Skip (0) / Stop (1)</b>	1	1	0	1	0	1	1	0	0	1
<b>Alighting Demand</b>	0	3	0	2	0	4	4	0	0	10

### 4.3.2 Evaluation Result of Scenario 2

The performance of the DDQN approach in Figure 4.6 was slightly lower than in Scenario 1, but it still outperformed the conventional method. Table 4.4 presents the average scores of both approaches, and it is observed that the difference between the scores of the DDQN and conventional methods reduced compared to Scenario 1.

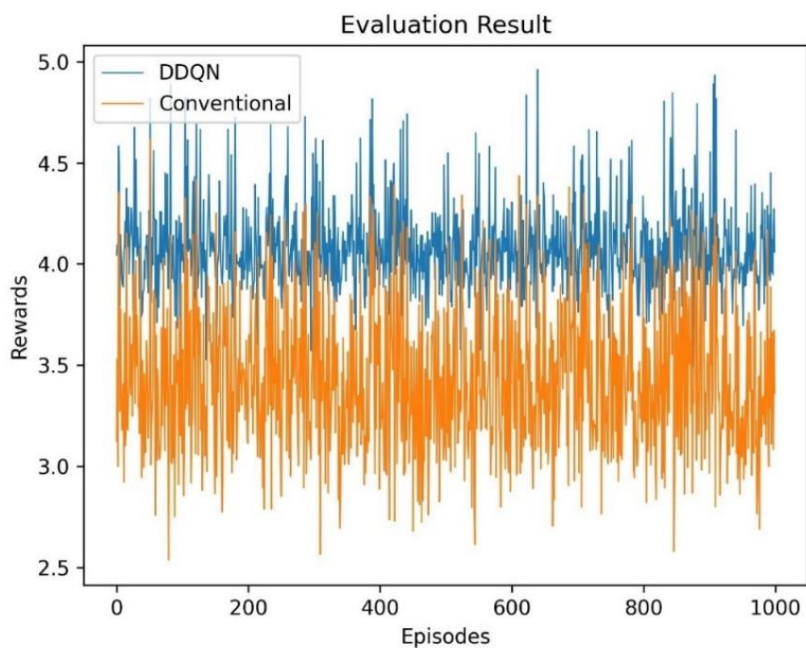


Figure 4.6: Evaluation Result of Scenario 2.

Table 4.4: Average Evaluation Score of DDQN and Conventional Approaches in Scenario 2.

Approach	DDQN	Conventional
Average Score	4.084	3.425

Table 4.5 presents an example of a skip-stop decision made by the DDQN agent based on a set of boarding and alighting demands of passengers. Unlike Scenario 1, the waiting passengers at each station in this scenario are randomly assigned based on time.

Table 4.5: Skip-stop pattern of DDQN Agent in Scenario 2.

Stop	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10
<b>Boarding Demand</b>	10	0	4	4	1	3	2	0	1	0
<b>Skip (0) / Stop (1)</b>	1	0	1	1	0	1	1	0	0	1
<b>Alighting Demand</b>	0	2	0	3	0	5	4	0	0	9

The agent obtains information about the passenger demands of the next three stations from the state space and selects the optimal action based on the passenger demands and ETA. From the skip-stop pattern in Table 4.5, it can be observed that the DDQN agent prioritizes serving stations with higher boarding demand. The slightly reduced performance of the DDQN agent in this scenario may be due to increased bus operating costs.

### 4.3.3 Evaluation Result of Scenario 3

The evaluation results for random passenger demand are presented in Figure 4.7, comparing the performance of the DDQN and conventional approaches. The DDQN approach slightly outperformed the conventional method, but the scores for both approaches in Table 4.6 were closer to each other than in



Scenarios 1 and 2. This suggests that the DDQN approach did not handle the passenger demand as effectively in this scenario.

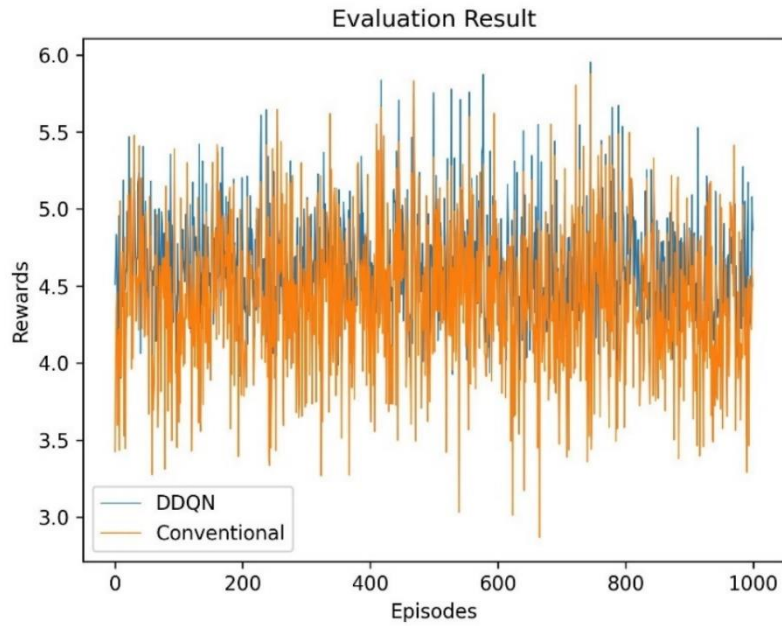


Figure 4.7: Evaluation Result of Scenario 3.

Table 4.6: Average Evaluation Score of DDQN and Conventional Approaches in Scenario 3.

Approach	DDQN	Conventional
Average Score	4.617	4.410

The skip-stop decisions made by the DDQN agent for a set of boarding and alighting demands are shown in Table 4.7. The waiting passengers at each station are randomly assigned without following any rules. Similar to Scenario 2, the DDQN agent tends to select stations with higher boarding demand at each decision point.

Table 4.7: Skip-stop pattern of DDQN Agent in Scenario 3.

Stop	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10
Boarding Demand	11	10	3	5	4	2	1	1	2	0

<b>Skip (0) / Stop (1)</b>	1	1	0	1	0	1	1	0	0	1
<b>Alighting Demand</b>	0	1	0	4	0	6	5	0	0	13

As compared to Scenario 1 and 2, the performance of the DDQN approach has dropped due to the penalties incurred from stranded passengers and their prolonged waiting time. In contrast, the score of the conventional approach increased as the passengers were distributed randomly along the route, which resulted in increased rewards as the bus served all passengers.

#### 4.4 Results of Model Deployment in Web Application

The DRL model has been successfully deployed in the web application. Figure 4.8 shows the user interface design of the developed web application. Google Maps is embedded in the web application using the Maps JavaScript API.

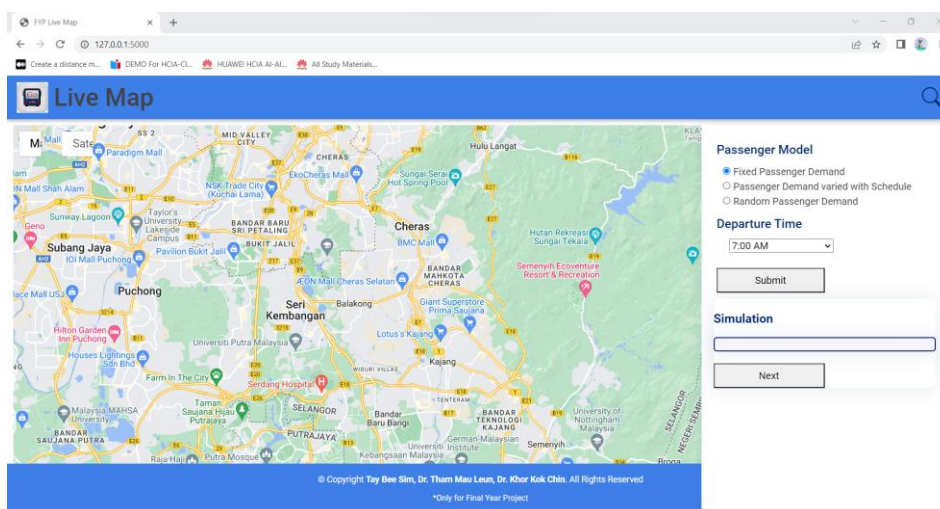


Figure 4.8: User Interface of Web Application.

There is a query panel on the right side of the web application where clients can select the passenger model and scheduled departure time of the bus, as demonstrated in Figure 4.9.

**Passenger Model**

Fixed Passenger Demand  
 Passenger Demand varied with Schedule  
 Random Passenger Demand

**Departure Time**

7:00 AM

7:00 AM

7:30 AM

8:00 AM

8:30 AM

9:00 AM

9:30 AM

10:00 AM

10:30 AM

11:00 AM

11:30 AM

12:00 PM

12:30 PM

1:00 PM

1:30 PM

2:00 PM

2:30 PM

3:00 PM

3:30 PM

4:00 PM

4:30 PM

Figure 4.9: Query Panel of Web Application.

As the client clicks the "Submit" button, the Python web server receives the data on passenger demand and departure time of the bus, and then transmits it to the DRL model to generate the best route. The server then responds to the client with the list of stations that the bus will stop at and marks the first station with a red icon on Google Maps, as shown in Figure 4.10.

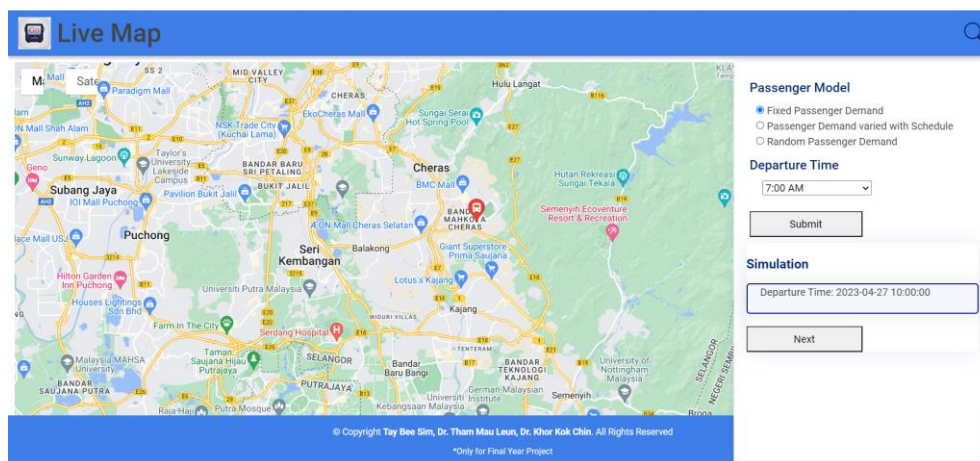


Figure 4.10: Initial Stage of Bus Simulation in Live Map.

The simulation begins when the "Next" button is clicked. Figure 4.11 and Figure 4.12 illustrate examples of simulation results. The scheduled departure time, origin, destination, skipped stations, and estimated time of

arrival are displayed. The blue bus icon represents the origin, while the red icon represents the destination. The simulation ends when the bus reaches the terminal station.

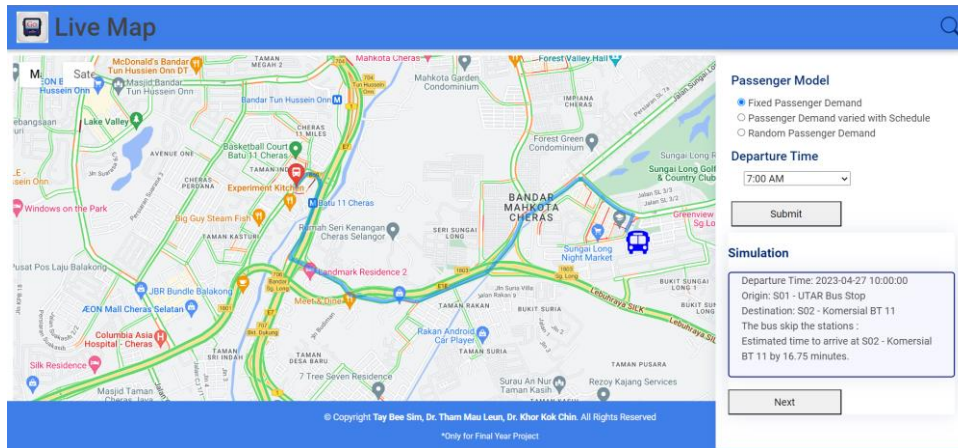


Figure 4.11: Bus Simulation Result from S01 to S02.

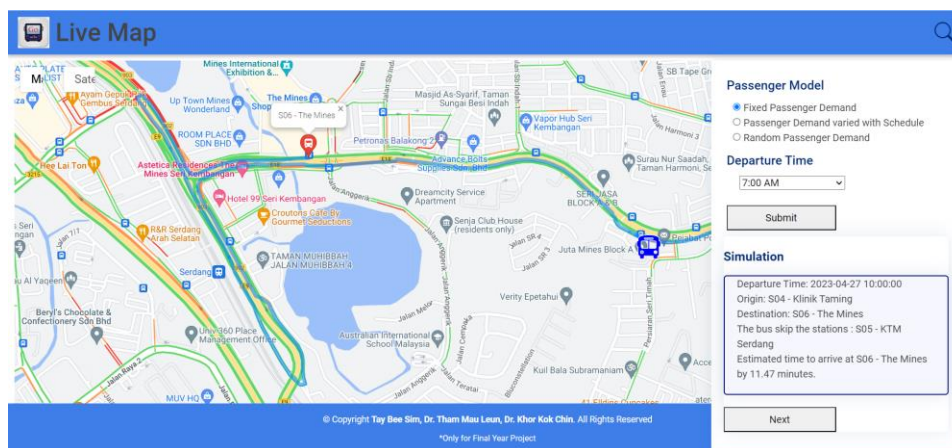


Figure 4.12: Bus Simulation Result from S04 to S06.

## 4.5 Summary

In summary, the performance of a DRL model was evaluated and compared with a conventional method under three different passenger demand scenarios. The evaluation results showed that the DDQN approach outperformed the conventional method in all three scenarios. However, the performance of the DDQN approach decreased with dynamic and randomly distributed passenger demands. The prolonged waiting time of stranded passengers resulted in penalties, causing a decrease in the DDQN approach's score. Conversely, the

conventional approach's score increased due to the random distribution of passengers along the route, leading to an increase in rewards as more passengers were served. The DDQN approach performed best in the static passenger demand scenario, followed by the scenario with dynamic passenger demands according to time, and lastly the randomly distributed passenger demand scenario.

## CHAPTER 5

### CONCLUSIONS AND RECOMMENDATIONS

#### 5.1 Conclusions

The optimization of public transport routes is an alternative solution to reducing the mobility demand of large populations. Stop-skipping operations aim to optimize the operation of buses by distributing unbalanced passenger demand along the bus line. In this study, a bus optimization scheme consisting of one express bus and one regular bus without stop-skipping is applied to develop the dynamic bus environment. A DRL approach is proposed to optimize the route of the express bus.

The DRL model is designed to select the optimal route based on passenger demand and traffic conditions. Since actual passenger flow data cannot be obtained from open or closed sources, three passenger demand models are constructed to examine the performance of optimization under different scenarios. The ETA data between stations can be collected from the Google Maps Platform using the Distance Matrix API.

The reward function is essential in guiding the agent to achieve its goal. To improve passenger satisfaction, the design of the reward function takes into account the passenger demand, in-vehicle time of passengers, the number of stranded passengers, and the average waiting time for both normal and stranded passengers. Other parameters, such as bus travel time and constant operating cost penalty, are also included to minimize bus operator expenditures. The reward for the first bus trip is based on satisfying the passengers and reducing the operating cost of the bus, while the second bus trip serves the stranded passengers and compensates for their penalty. The step reward and cumulative reward guide the agent towards successfully optimizing the route.

The model training utilized ten sets of ETA datasets to train the model with different ETA data, while the model evaluation was performed using five sets of ETA datasets. The performance of the DDQN agent and conventional methods was evaluated using the designed reward function,

which considered passenger demand, in-vehicle time, the number of stranded passengers, and operating cost penalties. Based on the training results, the DDQN agent was successfully trained to select the route for the stop-skipping scheme in 5000 episodes under different passenger demand scenarios. The training results of the three passenger demand scenarios converged as the episodes grew.

The performance of the DDQN agent was compared with that of the conventional method using the reward function. The performance of both approaches was evaluated with respect to passenger satisfaction and bus operator expenditures. Based on the evaluation results, the DDQN performed best in the static passenger demand scenario, followed by the scenario with dynamic passenger demands according to time, and lastly, the randomly distributed passenger demand scenario. The static passenger demand scenario was set with unbalanced passenger demand, and the stop-skipping service was found to perform best in unbalanced passenger demands by comparing the evaluation results of Scenario 1 with those of Scenarios 2 and 3.

Based on the simulation results of stop-skipping patterns, the RL model successfully trained the agent to perform stop-skipping operations based on the designed reward function, with a trend of serving the station with higher boarding demand. Three trained models were deployed in the web application, which performed step-by-step simulations for three different passenger scenarios.

The project's main purpose and objectives were successfully achieved by developing a DRL-based public transport route optimization, comparing the results with conventional methods, and finally implementing the solution in Google Maps.

## **5.2 Recommendations for future work**

In future work, several recommendations can be implemented to further enhance the accuracy and efficiency of DDQN public transport route optimization. One important consideration is incorporating load constraints to ensure the feasibility and realism of the optimized routes. Additionally, it is recommended to take into account the actual cost of bus operators, including

factors like utilization rate, to optimize the routes in a manner that maximizes resource utilization while minimizing operational costs.

Collecting real-time data using the Internet of Things (IoT) can also greatly improve the accuracy of the model. This includes collecting real-time data on bus locations, passenger counts, and other relevant variables, which can be used to make real-time adjustments to the optimized routes as needed. Furthermore, environmental data such as weather and traffic conditions can also be collected and incorporated into the model to further improve its accuracy.

In summary, incorporating load constraints, actual cost of bus operators, and collecting real-time data using IoT and environmental data can all greatly enhance the accuracy and efficiency of DDQN public transport route optimization. These recommendations can be explored in future work to improve the optimization of public transport routes and contribute to a more efficient and sustainable transportation system.



## REFERENCES

- Ai, G., Zuo, X., Chen, G. and Wu, B., 2022. Deep reinforcement learning based dynamic optimization of bus timetable. *Applied Soft Computing*, [e-journal] 131(109752). <https://doi.org/10.1016/j.asoc.2022.109752>.
- Chen, J., Liu, Z., Zhu, S. and Wang, W., 2015. Design of limited-stop bus service with capacity constraint and stochastic travel time. *Transportation research part E: logistics and transportation review*, [e-journal] 83, pp.1-15. <https://doi.org/10.1016/j.tre.2015.08.007>.
- Fu, L., Liu, Q. and Calamai, P., 2003. Real-time optimization model for dynamic scheduling of transit operations. *Transportation research record*, [online] 1857(1), pp.48-55.
- Ganapathy, V., Soh, C.Y. and Joe, H.K., 2009. *Neural q-learning controller for mobile robot*. In 2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics. Singapore, 14-17 July 2009. IEEE.
- Gkiotsalitis, K. and Cats, O., 2021. At-stop control measures in public transport: Literature review and research agenda. *Transportation Research Part E: Logistics and Transportation Review*, [e-journal] 145, p.102176.
- He, R.H., Wang, R., Nie, S.S., Wu, J., Zhang, J.H. and Wang, Z.M., 2021. Deep reinforcement learning for universal quantum state preparation via dynamic pulse control. *EPJ Quantum Technology*, [e-journal] 8(29). <https://doi.org/10.1140/epjqt/s40507-021-00119-6>.
- Ibarra-Rojas, O.J., Delgado, F., Giesen, R. and Muñoz, J.C., 2015. Planning, operation, and control of bus transport systems: A literature review. *Transportation Research Part B: Methodological*, [e-journal] 77, pp.38-75. <https://doi.org/10.1016/j.trb.2015.03.002>.
- IBM, 2020. *Neural Networks*. [online] Available at: <<https://www.ibm.com/cloud/learn/neural-networks>> [Accessed 25 August 2022].
- Iliopoulou, C. and Kepaptsoglou, K., 2019. Combining ITS and optimization in public transportation planning: state of the art and future research paths. *European Transport Research Review*, [e-journal] 11(1), pp.1-16. <https://doi.org/10.1186/s12544-019-0365-5>.
- Jagtap, R., 2020. *Understanding Markov Decision Process (MDP)*. [online] (28 September 2020) Available at: <<https://towardsdatascience.com/understanding-the-markov-decision-process-mdp-8f838510f150>> [Accessed 5 August 2022].

Jiang, Z., Gu, J., Fan, W., Liu, W. and Zhu, B., 2019. Q-learning approach to coordinated optimization of passenger inflow control with train skip-stopping on a urban rail transit line. *Computers & Industrial Engineering*, [e-journal] 127, pp.1131-1142. <https://doi.org/10.1016/j.cie.2018.05.050>.

Kikuchi, S. and Vuchic, V.R., 1982. Transit vehicle stopping regimes and spacings. *Transportation Science*, [e-journal] 16(3), pp.311-331. <https://doi.org/10.1287/trsc.16.3.311>.

Lee, D., 2019. *Reinforcement learning, part 3: the Markov Decision Process*. [online] Available at: <<https://medium.com/ai%C2%B3-theory-practice-business/reinforcement-learning-part-3-the-markov-decision-process-9f5066e073a2>> [Accessed 2 August 2022].

Liu, Z., Yan, Y., Qu, X. and Zhang, Y., 2013. Bus stop-skipping scheme with random travel time. *Transportation Research Part C: Emerging Technologies*, [e-journal] 35, pp.46-56. <https://doi.org/10.1016/j.trc.2013.06.004>.

Li, Y., Rousseau, J.M. and Gendreau, M., 1991. *Real-time scheduling on a transit bus route: A 0-1 stochastic programming model*. In: Proceedings of the Thirty-Third Annual Meeting, Transportation Research Forum, New Orleans, Louisiana. 31 October – 2 November 1991. Transportation Research Forum.

Li, Y., 2017. Deep reinforcement learning: an overview. *Arxiv*[online] Available at: <<https://arxiv.org/abs/1810.06339v1>> [Accessed 20 August 2022].

Lu, K., Han, B. and Zhou, X., 2018. Smart urban transit systems: from integrated framework to interdisciplinary perspective. *Urban Rail Transit*, [e-journal] 4, pp.49-67. <https://doi.org/10.1007/s40864-018-0080-x>.

Martín-Guerrero, J.D. and Lamata, L., 2021. Reinforcement learning and physics. *Applied Sciences*, [e-journal] 11(18), p.8589. <https://doi.org/10.3390/app11188589>.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M., 2013. Playing atari with deep reinforcement learning. *Arxiv* [online] Available at: <<https://arxiv.org/abs/1312.5602v1>> [Accessed 20 August 2022].

Ortega, F.A., Mesa, J.A., Piedra-De-La-Cuadra, R. and Pozo, M.Á., 2019. A matheuristic for optimizing skip–stop operation strategies in rail transit lines. *International Journal of Transport Development and Integration*, [e-journal] 3 (4), 306-316. <https://doi.org/10.2495/TDI-V3-N4-306-316>.

Sun, A. and Hickman, M., 2005. The real–time stop–skipping problem. *Journal of Intelligent Transportation Systems*, [e-journal] 9(2), pp.91-109. <https://doi.org/10.1080/15472450590934642>.

Sutton, R.S. and Barto, A.G., 2018. *Reinforcement learning, second edition: an introduction*. [e-book] MIT Press. Available at: Google Books <<https://books.google.com.my/books?id=sWV0DwAAQBAJ>> [Accessed 2 August 2022].

Van Hasselt, H., Guez, A. and Silver, D., 2016. Deep reinforcement learning with double q-learning. *Proceedings of the AAAI conference on artificial intelligence*, [e-journal] 30(1). <https://doi.org/10.1609/aaai.v30i1.10295>.

Zhang, H., Zhao, S., Liu, H. and Liang, S., 2018. Design of limited-stop service based on the degree of unbalance of passenger demand. *PLoS ONE*, [e-journal] 13(3): e0193855. <https://doi.org/10.1371/journal.pone.0193855>

Zhang, X. and Jia, L., 2021. Discussion on optimization of public transportation network setting considering three-state reliability. *Journal of Advanced Transportation*, [e-journal] 2021, pp.1-7. <https://doi.org/10.1155/2021/6940263>.