UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Deep Reinforcement Learning and Privacy Preserving with Differential Privacy

by Suleiman Abahussein

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of Prof. Tianqing Zhu and Prof. Wanlei Zhou

University of Technology Sydney
Faculty of Engineering and Information Technology

Sep 2023

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, *Suleiman Abahussein* declare that this thesis is submitted in fulfilment of the requirements for the award of *Doctor of Philosophy*, in the *School of Computer Science*, *Faculty of Engineering and Information Technology* at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution. This research is supported by the Australian Government Research Training Program.

SIGNATURE: Production Note: Signature removed prior to publication.

[Suleiman Abahussein]

DATE: 14 th September, 2023

PLACE: Sydney, Australia

Deep Reinforcement Learning and Privacy Preserving with Differential Privacy

By Suleiman Abahussein

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

Under the supervision of

Prof. Tianqing Zhu and Prof. Wanlei Zhou

University of Technology Sydney
Faculty of Engineering and IT

Sep-2023

Dedication

My late father, who instilled in me qualities of independence and determination, my devoted mother, whose presence has been essential for my accomplishments and triumphs, and my wife and sons all serve as dedications for this thesis...

ACKNOWLEDGMENTS

To commence, I thank almighty God for everything and for directing me and granting me both mental and physical fortitude during my pursuit of my PhD. I extend my deepest appreciation to my supervisors, Prof. Tianqing Zhu and Prof. Wanlei Zhou, for their priceless counsel, unwavering support and patience throughout my doctoral journey. Their extensive experience and vast knowledge have consistently enhanced my daily life and bolstered my research endeavors. I would also like to express my gratitude to Dr. Dayong Ye for his invaluable guidance and unwavering support throughout my research journey. His expert advice and assistance have been instrumental in refining my papers and enhancing the quality of my research. His profound expertise and extensive background have consistently fortified my research pursuits and everyday existence. I am really appreciative and thankful to Prof. Tianqing and Dr. Dayong Ye for offering invaluable guidance and great ideas they gave me through which I was able to solve challenging problems and enhance the depth of my research ultimately strengthening its overall quality and robustness. I would like to thank Prof. Bo Liu for unlimited support. I want to thank all external collaborators, Umer Siddique, for his support and guidance in my research and experiments. Furthermore, I would like to extend my appreciation to the entire staff at the School of Computer Science for their invaluable assistance in resolving various challenges faced by PhD students. I consider myself so lucky to be a member of our esteemed research group, Cyber Security and Privacy. Additionally, I am deeply indebted to my research companions, including Steven Cheng, Congcong Zhu, Sheng Shen, Tao Zhang, Lefeng Zhang, Chi Liu and countless others, for their unwavering support throughout my doctoral journey, both in terms of research and daily life. I would like to express my special thanks Steven, Sheng and Congcong for their support and guidance in my research and experiments.

List of Publications

Related to the thesis :

1. S. Abahussein, Z. Cheng, T. Zhu, D. Ye, and W. Zhou, Privacy-preserving in double deep-Q-network with differential privacy in continuous spaces, Australasian Joint Conference on Artificial Intelligence, (2022), pp. 15-26 (Published).

2. S. Abahussein, T. Zhu, D. Ye, Z. Cheng, and W. Zhou, Protect trajectory privacy in food delivery with differential privacy and multi-agent reinforcement learning, International Conference on Advanced Information Networking and Applications, (2023), pp. 48-59(Published).

3. S. Abahussein, D. Ye, C. Zhu, Z. Cheng, U. Siddiq, and S. Shen, Multi agent reinforcement learning for online food delivery with location privacy preserving, Information,(2023), 14(11), p.597 (Published).

Not related to the thesis :

1. A. Alorini, S. Abahussein, A. Bin Sawad, I. Mckie, M. Prasad, and A. Kocaballi, Understanding Privacy Protection Behaviour of Smart Speaker Users: A Systematic Review (Draft)

Abstract

With the rapid advances in technology in the current era and the emergence of multiple technologies that have transformed society, Deep reinforcement learning (DRL) offers promising solutions and enhanced capabilities with demonstrated superior results. Deep reinforcement learning is a subfield of artificial intelligence that has attracted significant research attention and development over the past few years. Reinforcement learning (RL) is enabled by deep learning to address the intractable problems previously encountered, for example, how an agent learns to play video games with only pixels as input. In the field of robotics, deep reinforcement learning algorithms are employed where control policies for robots can be learned directly from camera inputs in the real world. Deep RL aims to maximize the cumulative reward through the process of trial and error to find the optimal policy. The learning method is carried out by executing the action, receiving corresponding rewards and then moving to the next state. In some complex problems, it is necessary to have more than one RL agent, which leads to the idea of multi-agent reinforcement learning, where more than one agent works together and shares the same environment to achieve a certain goal. An example of multi-agent RL is multiple robotics working to rescue an individual.

Nowadays, deep reinforcement learning is used in various areas, such as recommendation systems, robotics, and health applications. While there are enormous benefits to using these technologies, there are also significant privacy concerns associated with them. The learning process in deep reinforcement learning involves performing the action, receiving the reward, and moving to the next state. Deep reinforcement learning is vulnerable to adversary attacks, and private information can be inferred by an adversary using recursive querying. The trained policy could be released to the client side, which could enable the adversary to infer private information from the trained policy, pose a real risk, and constitute a breach of privacy. This research focuses on deep reinforcement learning and multi-agent reinforcement learning and the related privacy issues. The contributions made by this research are as follows:

- This research proposes a solution for online food delivery services to increase the number of food delivery orders and thereby increase the long-term income of couriers. The solution involves leveraging multi-agent reinforcement learning by employing two multi-agent reinforcement learning algorithms to guide couriers to areas with a high demand for food delivery orders.

- This research proposes a solution to protect privacy in Double and Dueling Deep Q Networks by adopting the Differentially Private Stochastic Gradient Descent (DPSGD) method and injecting Gaussian noise into the gradient.
- This research proposes the Protect User Location Method (PULM) to protect customer location information in online food delivery services. This method injects differential privacy Laplace noise based on two factors: the size of the city and the frequency of customer orders.
- This research proposes a Protect Trajectory and Location in Food Delivery (PTLFD) method to maintain the privacy of the customer's stored data in online food delivery services. This method leverages multi-agent reinforcement learning and differential privacy to protect customer location information.

Keywords: Multi-agent reinforcement learning, Deep reinforcement learning, Privacy, Differential privacy, Online food delivery, Trajectory

Table of Contents

List of Figures

Figure **Page**

List of Tables

Table **Page**

Part I

CHAPTER 1

INTRODUCTION

Artificial intelligence (AI) refers to simulating critical thinking and intelligent behaviour similar to a human being using computers and technology. The term AI described as the science and engineering of making intelligent machines was first proposed in 1956 by John McCarthy [62]. Machine learning is a subfield of artificial intelligence that has been used broadly in various areas in recent years. Machine learning has emerged as the method of choice to develop practical software such as natural language processing, computer vision, speech recognition robot control, and other applications. Various applications have used machine learning and have achieved remarkable results in a wide variety of domains.

Machine learning is a technical domain lying at the intersection of statistics and computer science and is at the core of data science and artificial intelligence. Machine learning addresses the question as to how to build computers that improve automatically through experience. Numerous studies have been conducted to make machines learn by themselves without being explicitly programmed and several approaches have been applied by mathematicians and programmers to find a solution to this issue. The recent advancement in machine learning has been achieved by the development of theory and new learning algorithms and the huge availability of low-cost computation and online data [44]. Training machines on how to handle data more efficiently is one of the essential roles of machine learning. In cases where it is difficult to interpret the information extracted from data, machine learning can be used to assist this process.

The application of machine learning is increasing, with 65% of companies adopting it to help them make faster and more accurate decisions. Machine learning is robust as it is able to achieve about 92% accuracy in predicting COVID-19 mortality. By employing a machine learning algorithm that makes personalized content recommendations for users, Netflix was able to save $1 billion [103] [61].

1.1 Deep Reinforcement Learning

Deep reinforcement learning (DRL) is a subfield of machine learning that can be used to learn valuable representations for problems with high dimensional raw data input [66]. DRL has revolutionized the machine learning (ML) field and has enabled autonomous systems to be built with a greater ability to understand the visual world. DRL uses algorithms that integrate deep learning models into reinforcement learning algorithms, achieving great success [58]. A deep reinforcement learning agent interacts with its environment and seeks to learn an optimal policy by trial and error for sequential decision-making problems [52]. Recently, there has been an interest and increase in the use of deep reinforcement learning, which has led to recent advances in deep reinforcement learning algorithms. DRL algorithms are used in the field of robotics, where control policies for robots can be learned directly from camera inputs in the real world [11]. Despite most attention and development of algorithms being in gaming, Deep reinforcement learning algorithms have mostly been applied to gaming, however, they can be applied in other fields, such as self-driving, medical diagnosis and face recognition [78] [58] [106] [4].

1.2 Multi-agent Reinforcement Learning

Reinforcement learning and deep reinforcement learning recently received significant attention from the research community due to its ability to solve a large range of problems without the need for prior knowledge of the dynamics of the problem to solve. Although deep reinforcement learning has achieved great success, there are a number of real-world problems that a single agent cannot solve alone and more than one agent is required to solve it [21]. Multi-agent reinforcement learning deals with the problem of sequential decision making where multi-autonomous agents operate in the same environment. Every agent aims to maximize its long-term return by interacting with other agents and the environment. Generally, multi-agent reinforcement learning algorithms can

be classified into three types, depending on the types of settings they address: fully competitive, fully cooperative, and a mix of the two. In the cooperative setting, agents collaborate to maximize the common long-term return; in the competitive setting, the return of agents is usually summed to zero; and the mixed setting involves both cooperative and competitive agents [120]. Multi-agents can experience high computational complexity with an increase of the discrete state-action space in the number of state and action variables. Also, multi-agents can face non-stationarity issues, as in a multi-agent environment, all the agents learn simultaneously. Thus, the optimal policy could change because other agents' policies change [20].

1.3 Privacy

With the increase in the use and popularity of machine learning and deep reinforcement learning, serious privacy concerns have been raised. There is a strong possibility that private information may be leaked in some recent machine learning models. For instance, in a black-box membership inference attack, it is possible that the data points of individuals that are used to train the model can be identified where the adversary who tries to infer some private information by indirectly sending inputs to the machine learning model and receives output where the adversary later try to construct a shadow model from the data collected [87] [85]. Moreover, the trained policy in deep reinforcement learning could be released to the client side, which is trained based on some inputs and signals of rewards that usually rely on sensitive data. An example is the recommendation system in DRL, which frequently may use reward signals that based on user historical records. This could enable the adversary to infer private information from the policy [4] [104]. Pan et al. [74] conducted an experiment to examine private information leakage in deep reinforcement learning in the Grid World environment. The agent in this environment is tasked with finding a path from his current location to a particular destination while avoiding obstacles in the Grid World environment. After being trained using DQN, the agent is able to find the path to the goal without colliding with obstacles. Surprisingly, when all obstacles are removed, the agent still follows the same trajectory. This experiment indicates that optimal actions can be revealed from the trained policy, and the structure map can be inferred using this information which is based on the optimal action of the agent at every location [74]. Moreover, certain vulnerabilities in deep reinforcement learning can be exploited by adversaries with the means to change or disrupt control policies, resulting in unintended and potentially harmful actions. For

example, the manipulation of navigation policies and obstacle avoidance learned by autonomous unmanned aerial vehicles allow the systems to be used by the adversary as kinetic weapons by inducing actions that lead to intentional collisions [15].

Different defence methods have been proposed to tackle these issues such as differential privacy, or anonymity. The anonymity aim to hide the real identity to protect the privacy [26] [23]. Differential privacy is one of the robust methods that can be used to protect privacy. The method of differential privacy was proposed by Dwork et al. [35] [36] in 2006. It is a privacy guarantee method that has a robust standard for algorithms on aggregate databases. It has various characteristics that make it beneficial in applications like group privacy, robustness to auxiliary information and composability [3].

1.4 Research Objective

This research investigates deep reinforcement learning, multi-agent reinforcement learning and the related privacy issues. Initially, we designed a multi-agent reinforcement learning method that increases the number of received online food delivery orders and correspondingly increases the long-term income of the courier. However, deep reinforcement learning suffers from several privacy issues, such as inference attacks. Therefore, we also investigate privacy protection in Double and Dueling Deep Q Networks, which is a variant of deep reinforcement learning from being inferred by the adversary.

As it is possible that the personal information of a user of a food delivery service could be disclosed by the adversary, particularly the user's location, two privacy methods are proposed. First, we propose the Protect User Location Method (PULM) to protect information on the customer's location. This method injects differential privacy noise using two factors: the city area size and the frequency of the customer's online food delivery orders. Also, we propose the Protect Trajectory and Location in Food Delivery (PTLFD) method to protect information on the customer's location by leveraging multi-agent reinforcement learning and differential privacy.

1.5 Main Contributions

This section describes in detail the main contribution of this research.

1.5.1 Multi-Agent Reinforcement Learning for Online Food Delivery Services

Online food delivery services are becoming increasingly popular these days and demand for this service has increased sharply, particularly during the COVID-19 pandemic.

Improving the efficiency of this service is critical, which will benefit the company owner, couriers, customers and restaurants. We propose a solution to increase the number of orders received by couriers and thereby increase the couriers' long-term income. This method employs two state-of-the-art multi-agent reinforcement learning algorithms, QMIX and IQL, to run multiple agents concurrently to simulate real-world problems. We report our result with the average accumulated rewards of multi-agent reinforcement learning and the average number of received orders, and we attempt to compare our result with single-agent reinforcement learning. Our experiment is conducted on two datasets (Iowa USA and Shenzhen China) and one random synthetic dataset.

1.5.2 Privacy Preserving in Double and Dueling Deep Q Networks With Differential Privacy

Deep reinforcement learning is the combination of two concepts, deep learning and reinforcement learning, which uses a neural network instead as an approximator instead of a table. Recently, improvements into deep reinforcement learning algorithms have emerged, such as prioritized experience replay [83], Double Deep Q Network [102] and Dueling Deep Q Network [108]. While these algorithms show good improvement, there are potential privacy vulnerabilities. The trained policy could be released to the client side, which could allow the adversaries to infer the customer's demographic information from the policy. Moreover, in the model, there are many parameters, some of which could include sensitive information implicitly, which enables the adversary to infer sensitive information.

In this part, we consider the Double DQN and Dueling DQN algorithms and how to preserve privacy using these algorithms. We adopt the Differentially Private Stochastic Gradient Descent (DPSGD) method which is a sophisticated approach aiming to protect the privacy of training data and manage the added noise to protect the data without destroying the utility by injecting the controlled noise into the gradient. We show our results with average rewards and different amounts of injected noise [3].

1.5.3 Protecting User Location In Online Food Delivery Services

With the increased use of online food delivery services, particularly after COVID-19, significant privacy concerns have been raised about the protection of customer information when using these services. An adversary could infer and disclose private information about the user, such as the user's location or behavioural patterns, leading to privacy breaches.

In this part, we consider this issue and we propose the Protect User Location Method (PULM) to protect the customer's location information when using online food delivery services. PULM injects differential privacy Laplace noise into the user's location and courier's trajectory where the city area size and customer frequency of online food delivery orders are used to determine privacy parameters ϵ. We utilized several metrics to analyse our results, such as privacy parameter intensity and trajectory data utility.

1.5.4 Protecting the Customer's Location in Food Delivery Service with Differential Privacy and Multi-Agent Reinforcement Learning

Protecting the customer's information when using online food delivery services is a critical issue. Therefore, we continue focusing on this issue and try to propose another more advanced method that leverages and combines differential privacy and multi-agent reinforcement learning.

In this part, we propose the Protect Trajectory and Location in Food Delivery (PTLFD) method, which leverages the differential privacy Laplace mechanism and multi-agent reinforcement learning QMIX algorithm for improved results. After the courier receives the order and delivers the order to the customer, the agent constructs N number of obfuscated trajectories with different privacy parameters ϵ. The multi-agent reinforcement learning then chooses one of the obfuscated trajectories from the constructed trajectories. The selected trajectory is then evaluated in terms of three factors: the similarity between the selected trajectory and the original trajectory, the sensitivity of the destination location and the frequency with which the customer uses online food delivery services.

1.6 Research Challenges

1.6.1 Multi-Agent Reinforcement Learning for Online Food Delivery Services

The popularity of online food delivery services has skyrocketed recently, especially after the COVID-19 pandemic. While there is an increased use of online food delivery services, there is a challenge faced in improving service efficiency and improving their services to be able to serve more people and thereby increase their long-term income. To solve this challenge, we propose a method based on a multi-agent reinforcement learning algorithm and using QMIX and IQL that aims to increase the number of orders received by couriers and thereby increase long-term income.

1.6.2 Privacy Preserving in Double and Dueling Deep Q Networks With Differential Privacy

The deep reinforcement learning (DRL) algorithm is a promising algorithm that performs well. The developer recently focused on this algorithm, and different variants, such as Double DQN and Dueling DQN algorithms, recently appeared. While this algorithm shows great results, some challenges face the algorithm, such as privacy vulnerability. The trained policy could be released to the client side, which could allow the adversaries to infer the customer's demographic information from the policy. Moreover, in the model, there are many parameters, some of which could include sensitive information implicitly, which enables the adversary to infer sensitive information.

As the importance of the new variants of DRL such as Double DQN and Dueling DQN algorithms we focus on these algorithms. To solve the privacy challenge in these particular algorithms we consider preserving privacy by adopting the Differentially Private Stochastic Gradient Descent (DPSGD) method to protect the privacy of training data and manage the added noise.

1.6.3 Protecting User Location In Online Food Delivery Services

Online food delivery services have recently gained popularity, particularly after the COVID-19 pandemic. While this service is highly demanded, there is a challenge related to significant privacy concerns, especially related to disclosing private information about the user, such as user location information.

To solve this issue, we propose two methods to protect customer location information in online food delivery services. We propose the Protect User Location Method (PULM), which injects differential privacy Laplace noise into the user's location and courier's trajectory where the city area size and customer frequency of online food delivery orders are used to determine privacy parameters ϵ. The second method is the Protect Trajectory and Location in Food Delivery (PTLFD) method, which leverages the differential privacy Laplace mechanism and multi-agent reinforcement learning QMIX algorithm for improved results.

1.7 Significance of this Research

This research focuses on new robust technology and how to utilize this to solve real-world problems. Deep reinforcement learning and multi-agent reinforcement learning achieve promising results in various areas and are durable enough to solve sequential decision-making problems. We focus on online food delivery services, which are frequently used today and particularly during the COVID-19 pandemic. We propose a solution using multi-agent reinforcement learning to enable couriers to receive more food delivery orders and thereby increase their long-term income.

While these technologies achieve significant results, there are significant privacy concerns that need to be addressed. We focus on the inference attack in some new variants of deep reinforcement learning. We use an advanced defence method to protect the model from inference attacks during the training process. Moreover, it is possible that the personal information of a user of a food delivery service could be disclosed by the adversary, particularly the user's location. Thus, we consider the privacy issues which arise when using online food delivery services and how to protect the customer's private information. We propose two defences methods to protect the customer's location information in online food delivery services.

1.8 Thesis Organization

This thesis contains two parts and is divided into seven chapters. A summary of the details of each chapter is as follows.

- **Part 1**:
- **Chapter 1**: presents an overview of the research, outlining the main objective, contributions and significance of the research.
- **Chapter 2**: presents preliminary information about reinforcement learning and privacy, including an overview of deep reinforcement learning, a description of the basic components of deep reinforcement learning, and an overview of multi-agent reinforcement learning. Also, it provides an overview of privacy, differential privacy and privacy attacks. Moreover, it presents a literature review for a selected topic of privacy and deep reinforcement learning.
- **Part 2**:
- **Chapter 3**: presents the issue of online food delivery services and how to use multi-agent reinforcement learning to increase the number of food delivery orders collected by the couriers and correspondingly increase their long-term income.
- **Chapter 4**: presents the privacy issue in the Double and Dueling Deep Q Network and how to protect the customers' privacy by using DPSGD with different amounts of noise.
- **Chapter 5**: presents the privacy issue in online food delivery services and explains the PULM, which is proposed to protect information on the customer's location when using online food delivery services.
- **Chapter 6**: presents the privacy issue in online food delivery services and explains the PTLFD, which is proposed to maintain the privacy of the customers' data when using online food delivery services by leveraging differential privacy and multi-agent reinforcement learning.
- **Chapter 7** concludes the thesis and summarizes the main contribution and future work.

CHAPTER 2

PRELIMINARY

This chapter shows the preliminary and literature review on the topics of deep reinforcement learning and privacy preservation. Initially, it presents an overview of deep reinforcement learning with its definition and key elements, such as agent, reward, actions and the Bellman Equation. Also, it shows the recent deep reinforcement learning methods such as double deep Q networks, dueling deep Q networks and multi-agent reinforcement learning. The second part presents an overview of privacy and an overview of differential privacy with differential privacy key elements. Finally, it presents a literature review for a selected topic of privacy and deep reinforcement learning.

2.1 Reinforcement Learning

Reinforcement learning is part of machine learning that works with continuous decision-making [38]. The agent in reinforcement learning attempts to learn behaviour by trial-and-error interactions in the dynamic environment to solve the problem they face [54]. Reinforcement learning allows self-decision makers, such as traffic signal controllers, to learn, observe and select the best action needed to take. For example, in order to manage traffic, it seeks to decide on suitable traffic timing and phases [114]. Reinforcement learning is learning from punishments and rewards [33] [45].

The main aspect of reinforcement learning is that the agent learns the desired behaviour from trial and error. The agent in reinforcement learning acquires or modifies

Notations	Explanation
s	state of the environment before action
s'	state of the environment after action
$\mathscr{S}$	set of states
a	action
$\mathscr{A}$	set of action
t	time sequence
r	reward
$Q(s,a)$	Q function
π	policy
T	transition function
α	learning rate
γ	discount factor

Table 2.1: Reinforcement learning notations

new skills and behaviours incrementally. One of the essential features of reinforcement learning is that it uses and practices trial-and-error in the opposite, other technologies should suppose to know all needed knowledge of the environment in advance like dynamic programming. Therefore, reinforcement learning doesn't need to have control or full knowledge of the environment, and it needs to collect and interact with the environment. The experience is acquired prior in an offline setting then it is used for learning as a batch. While, the data in the online setting, available in sequential order, update the agent behaviour continuously. In both situations, the learning algorithms are basically the same, but the main difference is the agent in an online setting can influence how it obtains the experience so that it is more beneficial to learn. The agent needs to deal with the exploitation and exploration dilemma while learning which it considers an additional challenge. The agent in an online setting can get information on the interesting part of the environment, which makes learning useful. For this reason, the reinforcement learning method provides the most computationally effective approach in practice, even if the environment is entirely known, as compared to other methods like dynamic programming that would be ineffective due to this shortage of specificity [38].

2.1.1 Reinforcement Learning Notation

There are some basic notations used in reinforcement learning, and the following is the description of the main notation. For example, s represents the state of the environment, a represents the action, r represents the reward, and t represents the time sequence.

More reinforcement learning main notations are available in Table **??** [85].

2.1.2 Basic Design of Reinforcement Learning

The principal concept of reinforcement learning is learning by interaction [11]. Figure 2.1 represents the primary design of the system of reinforcement learning with learning loops and their action. In this figure 2.1, the agent learns by interaction with the environment in order to choose the more suitable possible action (a_t) within the given state(s_t) in the environment at step t. The agent action changes the environment state from s_t to s_{t+1} and produces a reward r_t for the agent. After that, the agent determines and takes the appropriate action for the new state (s_{t+1}) , that way getting reward r_{t+1}, these actions are repeated till the agent reaches the desired destination after the numbers of reparations referred to in the conducted experiment, where the agent learns and trains himself and improves his ability to make the next decisions by choosing the most suitable action that can be taken in a certain environment state, using the received rewards during the process of training. The main environment role is to give the agent the probable and possible states that the agent may face for this problem and that the agent needs to react with. To help the agent in the learning process the environment gives a penalty as a negative reward or reward based on the action that has been taken by the agent at each particular state. Therefore the reward is a function for both state and action and not only just for the action. This means the same action may get a different reward with different states [84].

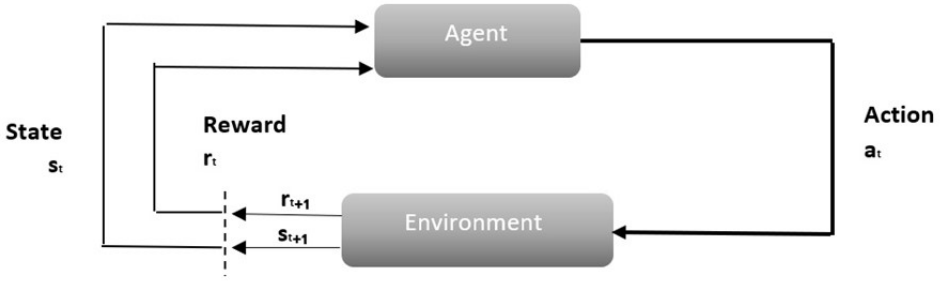


Figure 2.1: Basic design of Reinforcement Learning

2.1.2.1 Examples of Reinforcement Learning

The following are some examples of reinforcement learning to illustrate the concept of reinforcement learning better.

Robots are advancing technology, and reinforcement learning is deployed heavily in this technology, which allows Robots to learn and work independently. An example is the robots that are enabled by reinforcement learning to obtain optimal behaviour by using trial-and-error. In table tennis, the robot is trained to be able to return the ball over the net, the robot needs to learn from the mistake he made and improve his playing later [48]. Another example of reinforcement learning is a chess player. The player makes movements based on the player's planning and anticipating of probable replies. The player in the game learns from his mistakes over time and improves his performance [94]. Moreover, traffic congestion in cities has become a noisy phenomenon, and it has negative effects, many of which include psychological and financial. The idea was to adopt reinforcement learning to monitor and control traffic lights. Reinforcement learning can learn what are the quiet times and busy times are and can provide better control of traffic lights, which results in reducing congestion and waiting times for the driver [114].

2.1.3 Elements of Reinforcement Learning

There are four main elements of reinforcement learning: policy, a value function, a reward signal and the model of the environment. The following are more details about these elements.

In reinforcement learning, the policy π determines the way how the agent learns the behaviour at a given time. Generally, the policy π links the states in the environment to the action that is intended to be taken. It corresponds to what are called associations or stimulus-response rules in psychology. The policy π in some cases may look up a table or a simple function and, in other cases, it may contain extensive computations, possibly a search process. The policy π is the most important element in reinforcement learning as it determines the behaviour of reinforcement learning.

A reward r signal determines the goal of reinforcement learning. The environment on each time step sends to the agent of reinforcement learning reward. The agent's main goal is to get as much as they can of rewards r over time. The reward r signal determines what is the bad and good events for the agent. The reward r method in reinforcement learning is similar to a biological system, and it is analogous to pain or pleasure experiences. The agent receives the reward r at any time based on the current action of the agent and the current situation of the agent's environment. The reward r

mechanism can not be altered or changed by the agent. The agent can only influence the reward signal by the agent's actions, as it can directly affect to reward or indirectly through changing the status of the environment. The simplest example of reward signals is when the agent has a goal of eating breakfast. In this case, the agent of reinforcement learning directs his behaviour to receive different reward signals when eating breakfast based on the level of hunger. Also, it includes his current mood and other features of his body, the primary basis for altering the policy is to obtain the reward signal. When the agent does actions based on the current policy and get a low reward in the future, the policy may change to pick up other action for the same situation. Generally, the function of reward signals may be stochastic for the state of the actions taken and the environment [94].

A value function determines what is beneficial in the long term, while the reward signal defines what is good in the current time and represents the agent preference in a succinct form [10]. The reward specifies the environmental state immediate intrinsic desire, and the value represents the long-term desire with consider the states that are likely to follow and the rewards available in those states. For instance, a state might continuously produce immediate low rewards however, it still has a high value as it is frequently followed by other states that produce high rewards and the opposite could also be true. For example, in our life, when pain (low) and pleasure (high) are somewhat considered as rewards, whereas values correspond to farsighted judgment and a further refinement of how displeased or pleased we are in a particular state of our environment. The rewards are in a primary sense, while values, like reward predictions, are secondary. If there are no rewards, there might be no values, and the only objective of value estimating is to collect more rewards. However, the values are the most targeted at the time of evaluating and making decisions. We are looking for action that gives a higher value of states not actions that produce the highest reward, as these actions bring a higher amount of rewards over a long period.

The fourth main element of reinforcement learning is the environment model. The model of the environment is the representation of the agent for the environment, including the reward model and the transition model [52]. It simulates the environment's behaviour or, more generally, allows the making of inferences about what will be the behaviour of the environment. For instance, the resultant of the next state and the next reward could be predicted by the model from given a state and action. The main

purpose of modes is for planning, by which we mean for any decisions on the action and by assessing potential future situations before they are actually experienced [95].

2.1.4 Goals and Rewards

The goal of the agent in the environment is to increase accumulative reward as the feedback that the agent receives from collected reward shows to the agent which action led to failure or success [109]. The goal or the purpose of the agent in reinforcement learning is summarized in terms of the agent receiving a reward signal from the environment.

The reward in every time step t is number $r_t \in R$. The primary objective of the agent is to increase the total number of rewards. The maximizing of rewards does not mean immediate rewards but long-run accumulative rewards. This idea could be stated as the reward hypothesis, and one of the most unique characteristics of reinforcement learning is using a reward signal to represent the goal. In the beginning, it seems the meaning of goals in the context of reward signals is very limited, but it demonstrated that it is broadly applicable and flexible. For instance, in order to teach the robot how to walk, it should be given a proper reward for every time step that the robot forward motion. Also, to teach the robot to escape from the maze, it could be given -1 of reward for each time step passed prior to escape, so by this method the agent will try and learn to escape faster. Furthermore, in order to make recycling for empty cans, we can teach the robot how to collect cans by giving the agent zero rewards of the time and for every time the agent collects can get +1 reward. Also, the robot can get -1 reward when it crashes with anything or if someone shouts at it. Also, the agent can learn to play chess or checkers by giving +1 when it wins and -1 when the agent loses.

From the previous examples, it can be seen that the agent usually tries to learn how to increase the earned rewards. In the same way, we can determine any goal that we want the robot to perform and give the agent a proper reward when it wins and a negative reward when it loses. Therefore, when we set up the rewards in the right way, we can achieve our goal. Additionally, not only the rewards are used to give the agent prior information about how to accomplish our goal. For instance, the agent in the game of chess-playing game must get a reward only when achieving the main goal, not sub goal like taking a chess piece from the opponent or controlling the centre of the board. When we give the agent a reward for the sub goal, the agent will try to collect rewards

from sub-goal and not focus on the main goal [94].

2.1.5 Markov Decision Process

The Markov Decision Process (MDP) is formed of two terms: Markov, which refers to the Markov Property and the Decision Process. MDP is defined as a discrete time stochastic control process, and it uses the Markov Chain property to provide a Decision Process. Reinforcement learning could be defined as a Markov decision process that contains a group of action A, a group of states S with the start state $p(s_0)$, dynamic transition $T(s_{t+1}|s_t, a_t)$ to link a state-action at time t into a distribution of states at time t_{+1}, instantaneous/ immediate reward function $R(s_t, a_t, s_{t+1})$ and a discount factor $\gamma \in [1, 0]$, a higher emphasis on immediate rewards when lower values.

Generally, the policy π links the states into the probability distribution over actions. After every episode of length T the state reset if the MDP is episodic. In the episode, a series of actions, states and rewards constitute a rollout or trajectory of the policy. Accumulating rewards from the environment for every rollout of a policy giving $R = \sum_{t=0}^{T-1} \gamma^t r_{t+1}$. Finding the best policy π that gains from all states the maximum expected return is the goal of reinforcement learning [11] [84].

2.1.5.1 Bellman Equation

The name of Bellman Equation has been named based on the American mathematician Richard Bellman. This equation provides a recursion solution for Markov Decision Process problem. The optimal action and value function follow the Bellman equation [65]. The form of this recursive is adopted to overcome the problem of Markov Decision Process by using the algorithms of computer science such as linear programming and dynamic programming. It provides solution mathematically to evaluate the value of action for Q-function and value function.

2.1.5.2 The Action Value/Q Function

The quality of specific actions in a state is represented by Q-Function [57]. The Value Function has been discussed previously in the above section and how the agent uses their decision to get the best state and how later it picks up the suitable action to increase

the chance of obtaining a good state. This is a more indirect method to pick up the most desirable action. Decisions can be made by defining the most suitable action in a given current state, this is the " Action Value Function ", which is symbolised as $Q(s,a)$. To make it more clear the action-value function $Q(s,a)$, also called Q-Function is not the same as the like value function. The Value Function just only function of state s whereas Q-Function or action-value function $Q(s,a)$ is a function of both the action a and the state s.

2.1.5.3 Deep Q-Networks

In Deep Q Networks, the word Deep refers to the Deep Convolutional Neural Networks (CNN). Convolutional Neural Networks (CNN) are deep learning architecture work such as the brain visual cortex area, like humans to interpret and understand the received images from sensors [84]. In the algorithm of reinforcement learning, tabular style like SARSA or Q-learning has a lookup table known as Q-values or Q-table to represent the expected value estimates of a state, or state-action pair. With a small environment, this kind of approach works well, but this approach gets a lot of issues when the number of states increases as the table gets larger. As well, the number of actions increased in an action-value version of SARSA or Q-learning leads to increases in the table size, and the table becomes infinite if agent actions are continuous. Replacing the table with an approximation function is the popular solution to this issue. Building approximates function is an alternative option of storing the map of how states and actions alter the expected return. The agent looks at the current state-action pair at each time step and predicts the expected value. To solve this, there are a wide variety of regression algorithms [84].

2.1.6 Double Deep Q Network

This algorithm was initially introduced by Hado Hasselt in 2010 [39], and in 2016, it came up with an update for deep reinforcement learning Hasselt et al. [102]. In general, the state space and state size may be extremely large in some situations of deep reinforcement learning. The agent could take more time to learn sufficient information about the environment and which state/actions give the agent more rewards. In this situation, the exploration may be overwhelmed, and the agent may get stuck to estimated state-action combinations and exploit the explored that have relatively higher values. This could result in Q Value overestimation that causes suboptimal training. Also, in

DQN and Q-learning, the same max operator used the same values for both evaluating and selecting an action which is more likely to select overestimated values. Split the selection from the evaluation was used to prevent this issue. In Double DQN, they split the Q network into two separate Q networks. Figure 2.2 demonstrate more details about the two networks, the first one is an online/active Q network, and the second is a target Q Network. The target Q network is not updated immediately only after a certain number of steps does it get an update. This gave more stable target values in the target Q Network [84]. In double DQN the greedy policy is evaluated based on the online network, and using the target network to evaluate its value, the Double DQN can achieved based on the following [52].

$$y_t^{Double-DQN} = r_{t+1} + \gamma Q(s_{t+1}, \arg\max_a Q(s_{t+1}, a_t; \theta_t); \theta_t^-) \tag{2.1}$$

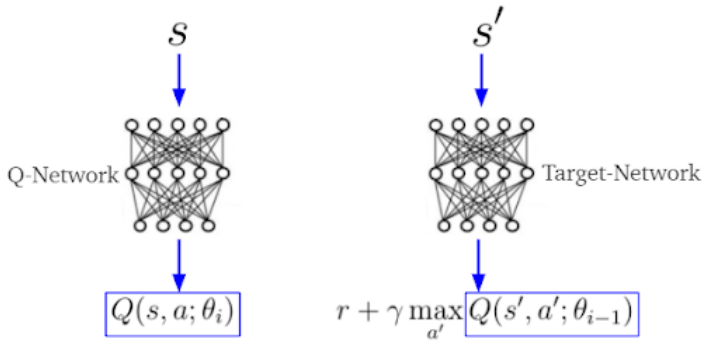


Figure 2.2: Double DQN diagram Oppermann A [73]

2.1.7 Dueling Deep Q Network

The architecture of Dueling network architecture was proposed by Wang et al. in 2016 [108]. The main idea of Dueling network architecture is that estimating the value of each action for many states is unnecessary. This method seeks to determine which states are valuable without having to learn the effect of each action for each state. For instance, in the game setting of Enduro, learning whether to move right or left is only needed when a crash is imminent [108]. However, in Dueling network architecture rather than using a single sequence of fully connected convolutional layers, they used two streams of fully connected layers. The Dueling network architecture works by estimating the advantage function $A(s;a)$ and state value function $V(s)$, which later they estimate

action value function $Q(s;a)$ by combining them. Figure 2.3 shows the architecture of the Dueling network architecture as it shows there are two CNN layer FC streams that are used by the Dueling network in order to estimate the value function and advantage function separately, which they later combine to estimate action-value function. Thus to get $Q(s;a)$, it needs to combine the $V(s)$ and $A(s;a)$. To obtain $Q(s,a)$ in the Dueling network we use the following equation:

$$Q(s,a;\theta,\alpha,\ \beta)\ = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \max_{a'} A(s,a';\theta,\alpha)\right) \tag{2.2}$$

The parameters of the convolutional layers are denoted by θ, while α and β are the parameters of the two streams of fully-connected layers [84].

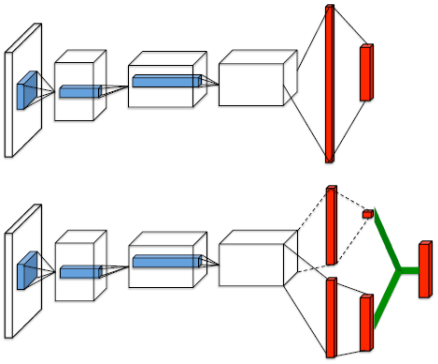


Figure 2.3: Double DQN diagram Wang et al. [108]

2.1.8 Multi-Agent reinforcement learning

One of the widespread solutions for sequential decision-making problems is reinforcement learning. In the setting of multi-agent reinforcement learning (MARL) we move to the problem of having more than one agent in the environment and abandon the problem of having one single agent. In this setting, there are more than one agent involved and multi-agent reinforcement learning deal with sequential decision-making problems. More specifically, the joint actions of all agents affect the evolution of the system state and the reward received by each agent. Each agent owns his reward and needs to optimise it this will be become all agent's policies function [120]. Figure 2.4 shows a multi-agent reinforcement learning diagram and how the agents in multi-agent reinforcement learning interact with their environment [101] [27].

Multi-agent reinforcement learning is the integration of two concepts: reinforcement learning to accomplish tasks and agents interacting with other agents. The Markov decision process is used as a model in the fully observation domain of multi agent reinforcement learning. It can define the multi agent fully observable Markov decision process as 6 tuples containing (1)$I = \{1,...,j\}$, representing the set of agents and j is the number of the agent. (2) S, at each time step, the environment's true state. (3) $A = \{a_1,\cdots,a_n\}$, is the agent available actions, whereas the action index is represented by the symbol n. (4) The transition function T, which is depend on the action of the agent. (5) The agent reward function R. (6) Each agent discount factor γ. In the domain of partial observation, it can be described as 8 tuples which comprised (7) A set of observations O. (8) Based on agent action, observation function B. The domain at each step t is in state $s \in S$, the agent pickup an action $a \in A$, and this action transfers the domain to a new state s' with probability $T(s\prime|s,a)$, the agent then gets a reward r based on $R(s,a)$, this process is repeated until agent stop. Learning the policy π is the primary goal of the agent, which will raise its expected discounted future reward $E[\sum_{0\leq t<\infty}\gamma^t r^t]$ as the number of sequential steps is t [27].

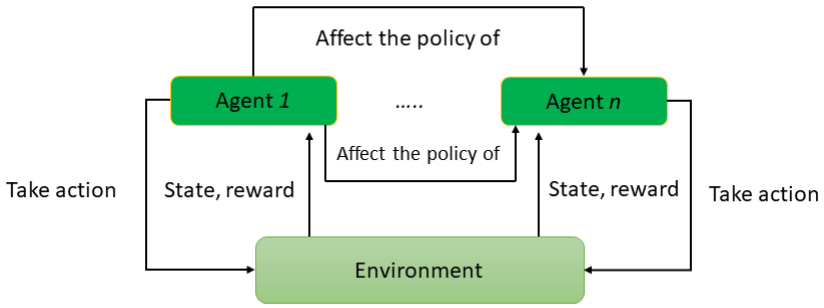


Figure 2.4: Multi-Agent reinforcement learning diagram

2.1.9 QMIX algorithm

In a multi-agent framework, every agent selects an action that creates a collective action a_t, and then the global immediate reward r_t is shared to assess the collective action taken. For the collective action, there is collective agent-value function $Q_{tot}(s_t,a_t) = \mathbb{E}_{s_t+1:\infty,a_t+1:\infty}[R_t|s_t,a_t]$, where at time t the discounted return is R_t. One of the main challenges in multi-agent reinforcement learning is how to assess every agent contribution individually and accurately to get a separate value function from the collective action-value function. The agent A_i individual value functions are represented

by $Q_i(o_t, a_t)$.

QMIX is an advanced technique of value-based aims to train uncentralized policies in an end-to-end centralized method [79]. To implement the QMIX method, there are two challenges that need to be overcome. The first challenge is in the process of centralized training where every agent needs to calculate the influence action based on a single global reward. The second challenge is when agents interact in a decentralized manner, and we need to guarantee that the ensemble of the optimal actions of agents is the optimal action of the ensemble of agents [122]. There is a belief in QMIX that it can interpret the total action-value function as the mixing network of each action-value function or a linear combination of each action-value function. Thus, it is expected that the QMIX-based method acts in cooperation with nearby agents. [118].

2.1.10 Independent Q-learning IQL

Independent Q-Learning (IQL) was proposed by Tampuu et al. [96]. The Independent Q-Learning is an algorithm that abandons centralized training where for each agent in IQL the Q-learning is performed separately. In some way, this method avoids the implementation issue of CTDE framework. Consequently, in a large probability, the single agent action can interfere with the overall environment and other agents of the state. Therefore, this makes the IQL can't converge in a complex environment. The IQL algorithm still has a number of applications in small scenarios of reinforcement learning applications such as Atari games.

In IQL, each agent trained decentralised Q-functions, while QMIX is a method learned in end-to-end of decentralised policies in a centralised setting. QMIX is composed of networks of agents representing every Q_a, and a mixing network that merges them into Q_{tot}. The most commonly used method in multi-agent learning is Independent Q learning (IQL), which consists of a group of concurrent single-agent that use and share the same environment where every agent learns individually [118] [79] [80] [34]. In our research, we intend to use different multi-agent reinforcement learning settings in order to show more results from our experiments.

2.2 Privacy

The issue of privacy is of profound importance to the world. In almost every nation, numerous regulations, judicial decisions and constitutional rights seek to preserve privacy. In the constitutional laws of some countries, privacy is enshrined as a primary right. Even though the US Constitution didn't mention the word privacy clearly, it protects the confidentiality of communications and the sanctity of the home from government intrusion. Additionally, the US Supreme Court has emphasised that the Constitution should preserve a "zone of privacy" including decisions that people make about their health, birth control, and sexual conduct, as well as protecting their personal information from unwarranted disclosures by the government [88].

The vast majority of countries explicitly protect privacy in their constitutions. For example, in Brazil, it was mentioned in the Brazilian constitution which was issued in October 1988, that "protects the right to privacy, including the secrecy of correspondence, telegraphic, telephone and data communications" [2]. While in Australia, they have introduced The Australian Privacy Act 1988, which contains 13 privacy principles to protect the individual's privacy and to regulate how organizations and Australian Government agencies should handle personal information [69].

Additionally, privacy is recognized as a primary human right. The United Nations has a declaration on Universal Human Rights of 1948 that nobody should be subjected to the interference of their privacy, correspondence, home, family or attacks on their reputation and honour [67]. Also, Article 8 of the European Convention on Human Rights states that everybody has the right to respect for his family life, privacy and their correspondence and their home [71]. Therefore, there is worldwide agreement on the importance of privacy and the necessity for privacy protection, as privacy is a primary right of the human being, the protection of human dignity and essential to autonomy, serving as the basis upon which several other human rights are built.

In terms of privacy definitions, there are numerous definitions the following some definitions of privacy. According to Solove (2008) [88] privacy is a comprehensive concept containing thought freedom, our ability to control our bodies, control of our personal information, isolation at home, reputation protection, nobody is spying on us and protection from search and questioning. According to Post (2000) [76], privacy is a value, very

complicated, very entangled in competing and contradictory dimensions, so engorged with different meanings that I feel sometimes despair if I can be usefully addressed at all. Privacy has been defined by The Office of The Australian Information Commissioner as the fundamentals of human rights that support freedom of association, thought and expression, as well as the freedom from discrimination. Privacy, from a technical perspective is aims to invent and creation of privacy-preserving mechanisms to preserve privacy in information and statistics data [68]. The privacy should include the following rights:

- Should be free from intrusion and interference.
- Associate freely with anyone you want.
- Should be able to control who can use or see information about you [70].

With the rapid development of technologies, communication and mobile devices. A large amount of data is collected from several different entities, such as governments, organizations, companies and social media. These data contain sensitive information such as personal information, health and bank, which raises serious concerns about privacy, as this information can be exposed or poorly used and threaten an individual's privacy.

Today, in the information technology field, there is a lot of research on preserving privacy that aims to protect privacy and not disclose private information. In general, these research methods can be categorized as follows: anonymity, perturbation, and encryption [112]. However, some of these methods are only able to solve some of the privacy issues, while there are other issues that are not solved. Background knowledge or linkage attacks still cause a potential risk of sensitive information leakage, making the anonymity method ineffective. In spite of this, encryption schemes provide high integrity and confidentiality of the data but it is not precisely linked to privacy in some sense. Additionally, after encryption, data cannot be used by a third party anymore. Moreover, the encryption schemes need high computational costs in practice, which may limit their usage.

Data perturbation is the most common approach in privacy preservation, where it is a technique for masking the data. The data perturbation uses the random value perturbation-based method to preserve the privacy of the data by adding random noise [46]. In 2006, Cynthia Dwork proposed differential privacy, which is a new promising

privacy model. Differential privacy is a form of data perturbation based on a probability that guarantees that the adversary's ability to create any harmful for the individual data in a dataset is similar. Differential privacy is resisting the majority of the attacks on privacy and provides a reliable privacy guarantee [125].

2.3 Differential Privacy

Today, with the advancement of data analysis and mining, data privacy threats are vastly increasing. Differential privacy, mainly, is a mathematical model that ensures the privacy of a statistical dataset [60]. It is a common model for privacy that provides privacy guarantees without knowing anything about the in trader's background knowledge. The differential privacy model is introduced to reduce the potential of disclosure risk that occurs when returning a result of the queries on a database [90]. The data collector, also called the curator, which is responsible for publishing data, provides aggregated information to the users that the users can use for further investigation. Figure 2.5 shows the privacy model located among the untrusted public users and trusted curator. The following is the definition of differential privacy.

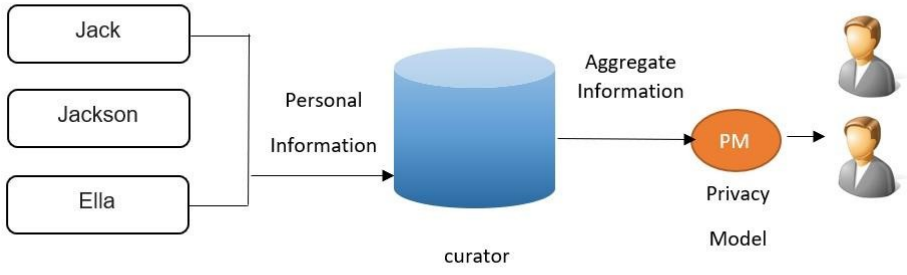


Figure 2.5: Privacy model

Definition 1. (ϵ, δ) differential privacy is a randomized mechanism $\mathscr{M}$ gives (ϵ, δ) for each set of $\mathscr{Z}$ output and for any D and D' neighboring datasets if $\mathscr{M}$ satisfies

$$\mathbb{P}[\mathscr{M}(D) \in \mathscr{Z}] \leq exp(\epsilon) \cdot \mathbb{P}[\mathscr{M}(D') \in \mathscr{Z}] + \delta \tag{2.3}$$

If $\delta = 0$, the $\mathscr{M}$ randomized mechanism gives ϵ differential privacy by its strictest definition. Differential privacy gives the freedom to break strict differential privacy

Notations	Explanation
χ	Universe
D	Dataset
D'	Neighbour dataset
$\mathscr{D}$	Data distribution
r	Record
d	Dataset dimension
n	Size of dataset
N	size of histogram
f	Query
F	Query set
m	Query number
$\mathscr{M}$	Mechanism
$\widehat{f}$	Noisy output
$\mathfrak{n}$	Noise
ϵ	Privacy budget
Δf	Sensitivity
c	Concept
C	Concept set
h	Hypothesis
H	Hypothesis set
$l(.)$	Loss function
θ	Threshold
δ	Confidence parameter
α, β	Accuracy parameter

Table 2.2: Differential privacy notations

for low probability events. The notation ϵ known as pure differential privacy and (ϵ, δ) differential privacy with $\delta > 0$ known as approximate differential privacy.

2.3.1 Differential Privacy Preliminaries

2.3.1.1 Notation

This section presents and describes some of the main notations used in differential privacy. The χ symbol indicates the finite data universe with the size of $|\chi|$. The symbols D and D' represent the neighbor dataset that has different records where r indicates to the record in the dataset and n is the number of records in the dataset. The f symbol represent the function, F represents the group of function, and m is used to represent

the number of queries in F. Table 2.2 shows more notation with description.

2.3.1.2 Privacy Budget

The privacy budget is a parameter that defines the privacy guarantee level of the mechanism $\mathscr{M}$ where the notation of ϵ is used to represent the privacy budget. Smaller ϵ defines a stronger privacy level and lower data utility [112]. For privacy budget, there are two theorems commonly used: parallel composition and sequential composition.

Theorem 1 (Parallel Composition). Assume there is group of privacy mechanisms $\mathscr{M} = \{\mathscr{M}_1, ..., \mathscr{M}_m\}$. If every $\mathscr{M}_i$ gave ϵ_i differential privacy assurance on a disconnected subset of whole dataset, $\mathscr{M}$ will produce $(\max\{\epsilon_i, ...\epsilon_m\})$ differential privacy.

Theorem 2 (Sequential Composition). Assume we have group of privacy mechanisms $\mathscr{M} = \{\mathscr{M}_1, ..., \mathscr{M}_m\}$ performed sequentially on dataset , every $\mathscr{M}_i$ produces ϵ differential privacy assurance, Thus, $\mathscr{M}$ will provide $(m \,.\, \epsilon)$ differential privacy [125].

2.3.1.3 Sensitivity

Sensitivity is a parameter that defines how much perturbation is required for a specific query in a mechanism.

Definition 2 (Sensitivity). For a query $f : D \rightarrow R,$ and D and D' dataset, the definition of sensitivity is :

$$\Delta f = \max_{D,D'} ||f(D) - f(D')|| \tag{2.4}$$

The absolute maximum distance of $|f(D) - f(D')|$ is the definition of sensitivity as D and D' are neighbouring inputs [3].

2.3.1.4 Principle of Differential Privacy Mechanisms

Any mechanism could be considered as differentially private if it meets the definition of (2.3). To guarantee differential privacy, two main mechanisms widely used for this purpose are the Laplace and the exponential mechanisms.

2.3.1.5 Laplace Mechanism

The Laplace Mechanism performs differential privacy by adding independent noise randomly from the Laplace distribution to the output [105]. The symbol $Lap(b)$ is used to show the noise sample of a Laplace distribution with an amount of b.

Definition 3 (Laplace mechanism). over dataset D and for each function $f : D \longrightarrow R$, the mechanism $\mathscr{M}$ provides ϵ differential privacy

$$\mathscr{M}(D) = f(D) + lap\left(\frac{\Delta f}{\epsilon}\right) \tag{2.5}$$

2.3.1.6 Exponential Mechanism

For non-numeric queries, the exponential mechanism is used to randomize the results where the score function $q(D,\phi)$ is used to evaluate the quality of the mechanism output ϕ. Determining function score depends on an application which leads different applications to several score function.

Definition 4 (Exponential mechanism): assume $q(D,\phi)$ is the score function of the D dataset which evaluates the quality of output $\phi \in \Phi$, where Δq represents the sensitive of ϕ then exponentially. Mechanism $\mathscr{M}$ satisfies ϵ differential privacy if:

$$\mathscr{M}(D) = \left(return\ \phi \propto exp\left(\frac{\epsilon q(D,\phi)}{2\Delta q}\right)\right) \tag{2.6}$$

2.3.1.7 Gaussian Mechanism

Definition 5 (Gaussian mechanism): The gaussian mechanism is defined based on

$$\mathscr{M}(D) \stackrel{\Delta}{=} f(D) + \mathscr{N}(0, \Delta\mathscr{S}_f^2\ .\ \sigma^2) \tag{2.7}$$

where the normal (Gaussian) distribution is $\mathscr{N}(0, \mathscr{S}_f^2\ .\ \sigma^2)$ with mean 0 and standard deviation $\mathscr{S}_{f\sigma}$. The Gaussian mechanism for single application to sensitivity Δf satisfies differential privacy (ε,δ) if $\delta \geq \frac{3}{4}\ exp(-(\sigma\varepsilon)^2/2)$ and $\varepsilon < 1$ [3].

2.3.1.8 Utility Measurement of Differential Privacy

There are various utility measurements used for data publishing and analysis when the privacy level is fixed to ϵ.

- Noise size measurement: how much noise is added to the query result is the easiest way to calibrate the noise. The approach of utility measurement is extensively used in data publishing.
- Error measurement: It can measure the utility by calculating the variance between the private and non-private outputs. The accuracy parameters usually represent the error measurement [125].

2.3.2 Differentially Private Data Publishing

The primary role of data publishing of differentially private is to provide aggregate information to the public without disclosing any individual record. It can be demonstrated this problem as follows: if there is a group of queries $F = \{f_1, \cdots, f_m\}$ received by the curator which has a dataset D. The curator has to answer every query and follow differential privacy constraints. In the publishing scenario, there are two settings: non-interactive and interactive. In the setting of an interactive scenario, it cannot issue the query f_i till published the previous query f_{i-1}. In the setting of non-interactive the curator receives all queries at one time, and with full knowledge, the curator provides answers to the query set. To illustrate the difference between the two settings, let's suppose the curator received these queries:

- f1: How many patients have blood pressure at the age of 40 to 55?
- f2: How many patients have blood pressure at the age of 55 to 75?

Assume for each query, the privacy budget ϵ is fixed. The curator in the interactive setting will begin with f_1 and start counting the number of patients who have blood pressure at the age of 40 to 55 and add independent Laplace noise to the result where the sensitivity equal is 1, $Lap(1/\epsilon)$. When the curator moves to second query f_2 the sensitivity will be 2, so the total noise will be $Lap(1/\epsilon) + Lap(2/\epsilon)$. On the other hand, all queries submitted to the curator one time in the setting of non-interactive. The sensitivity in this case will be 2 and the total added noise is $2 \times Lap(2/\epsilon)$, which is bigger than the interactive setting [125].

2.3.3 Differentially Private Data Analysis

The Differentially private data analysis primary role is to convert and extend existing non-private algorithms to be differentially private algorithms. This task can be achieved by different frameworks, generally categorized into private learning frameworks and exponential/Laplace frameworks. In terms of optimization, the frameworks of private learning consider data analysis as a learning problem. A series of objective functions have been defined to solve the learning problems. A private learning framework has a precise aim when compared to the exponential/Laplace framework, and in terms of risk bound or sample complexity, the results produced by a private learning framework are easier to compare. However, the private learning frameworks work only with limited learning algorithms, whereas the implementation of almost all types of analysis algorithms can be undertaken on the Laplace/exponential framework [125].

2.3.4 Data Inference Attack

The form of data inference attacks can be modelled as an inversion attack or membership inference attack, where every one of them has a different goal. The target model in these attacks is assessed by the owner and it is open to public users and the attackers as well. Only in the manner of black-box the model can be accessed, as the attacker receives the corresponding output when inputting the data sample into the model. The following shows more details about the membership inference attack and the inversion attack.

2.3.4.1 Membership Inference Attack

The goal of a membership inference attack is to infer and check if the training datasets in the target model have a particular data record. The model's prediction scores can be accessible by the attacker in these types of attacks. Therefore, to infer the record membership, the normal strategy is to train the model of attack that is commonly a binary classifier. The input can be any of the following: the given data record confidence score vector or its label or both. The output is the prediction of whether the record exists in the training datasets. It needs to be trained in a shadow model on an auxiliary dataset that is drawn from the same data distribution before training the attack model. Then training the model of the attack model on the confidence score vectors of non-membership/membership predicted by the shadow model.

2.3.4.2 Model Inversion Attack

By using confidence score vectors that the target model predicts, the model inversion attacks have the objective of reconstructing the input data. Separate attack models are trained by the attacker on an auxiliary dataset that works to the inverses the target model. The target model confidence score vectors are used by the attack model as input and then the attacker tries to produce the target model actual input data [115].

2.3.5 Trajectory Protection

Assume universe of locations is $\mathscr{L} = \{L_1, L_2, \cdots, L_{|\mathscr{L}|}\}$ and $|\mathscr{L}|$ is the universe size and it assumes the locations is discrete spatial areas in a map and it assumes a list of order locations that are drawn from the universe form the trajectory.

Table 2.3: Sample trajectory database

Rec #	Path
1	$L4 \rightarrow L5 \rightarrow L6$
2	$L2 \rightarrow L3$
3	$L2 \rightarrow L4 \rightarrow L5$
4	$L3 \rightarrow L5 \rightarrow L6$
5	$L4 \rightarrow L5$
6	$L1 \rightarrow L3$
7	$L3 \rightarrow L2$
8	$L1 \rightarrow L3 \rightarrow L5 \rightarrow L2$

Definition (Trajectory): With length of $|T|$ the trajectory T is an ordered list of locations $T = t_1 \rightarrow t_2 \rightarrow \cdots \rightarrow t_{|T|}$, where $\forall_1 \leqslant i \leqslant |T|$

In T the locations could be occur successively and could occur multiple times in T. Therefore, it is a correct trajectory $\mathscr{L} = \{L_1, L_2, L_3, L_4\}$, T = $L_1 \rightarrow L_2 \rightarrow L_3$. The timestamps, in some cases, could be included in the trajectory. The trajectory database consists of a multiple trajectories and the record owner of movement history is shown in every trajectory. The following is the formal definition:

Definition (Trajectory Database): With size of $|D|$, the trajectory database D is a multiset of trajectories $D = \{D_1, D_2, \cdots, D_{|D|}\}$.

A sample of database trajectory has been shown in Table 2.3 with $\mathscr{L} = \{L_1, L_2, L_3, L_4\}$ [24]. The attacker may get the user's location via the analysis of trajectory data. In this research, a defence method based on Differential privacy has been provided for some attack models existing in trajectory analysis. A properly calibrated of randomization mechanism injected with a meaningful amount of differential private noise drawn from a trajectory sensitivity function described in section 2.3. This will create differential private spatio-temporal data, the sanitization of a trajectory path is a result of the perturbation of these traces. Therefore, the perturbed trajectory path will prevent the attacker from determining the original trajectory path and preserve user privacy.

2.4 Literature Review

This section presents a literature review on certain topics related to deep reinforcement learning, multi-agent reinforcement learning, and preserving privacy.

2.4.1 Multi Agent Reinforcement Learning for Online Food Delivery

The delivery problem has been broadly studied in the literature and remains challenging. This part presents previous work on the delivery service problem and briefly explains our solution.

Chen et al. [25] focus on same-day delivery by drones and vehicles. By using vehicles multiple packages in one route can be delivered while the travel is fairly slow. In drones, travel is much faster, but it frequently requires battery charging and has limited capacity. Their proposed method is based on a deep Q-learning approach. The method learns to assign the delivery to the new customer to either vehicles, drones or the option of not providing the service at all. Liu et al. [56] focus on route planning. In order to capture the preferences of the delivery men from their historical GPS trajectories and recommend their preferred routes, they developed a deep inverse reinforcement learning (IRL) algorithm. Also, the Dijkstra algorithm was adopted in their work instead of value iteration to define the current policy and compute the IRL gradient. Xing et al. [110]

consider path problems. To optimize the delivery path, they use the improved method Heuristics in Deep Reinforcement learning and they analyze the aspects of delivery such as time constraints, timeliness and high coordination. Also, they compare their method with the traditional tabu search algorithm.

Ding et al. [32] consider an alternative to traditional delivery such as delivery by the crowd. Based on public transport, they develop a crowdsourcing delivery system, and they consider multiple aspects like multi-hop delivery, time constraints and profits. From massive package data and passenger data, reinforcement learning has been used to learn the optimal order dispatching strategies. Zou et al. [126] propose a reinforcement learning Double Deep Q Network (DQN) framework that gradually learns and tests the dispatch order policy by communicating with an Online to Offline (O2O) business simulation model designed by SUMO. Bozanta et al. [19] propose a model incorporating the order destination, order origin and courier location. Every courier has a task to gather the given order and deliver it to the wanted place. This model is designed to increase income from served requests on a limited number of couriers over a period of time. The model of the Markov decision process is considered to simulate actual food delivery service. The model has been applied to Q-Learning and Double Deep Q-Networks. Jahanshahi et al. [42] propose a model that employs deep reinforcement algorithms to solve a meal delivery service problem. The primary objective is to increase total profit by giving the orders to the most suitable couriers, reducing the expected delays and postponing or rejecting orders. Their result shows that the model significantly enhances the overall quality of service with incorporating the restaurant's geographical locations, customers, and the depot. Hu et al. [40] consider the dispatch problem in instant delivery services, where they dispatch a large number of orders to a few numbers of couriers, especially during peak hours. Based on multi-agent actor-critic, the method of Cross-region courier displacement has been proposed to solve this problem. The method not only can balance supply (courier's capacity) and demand (picking up orders), but it can also enhance the effectiveness of delivering orders by decreasing idle displacing time.

Many research tackles the delivery problem by using different methods, such as assigning the delivery request to the most appropriate courier, trying to minimise the rejected request or proposing an effective dispatch mechanism. In this work, we consider this issue and we employ multi-agent reinforcement learning. We used the novel algorithm QMIX and IQL to guide the courier to the area with high order demand to increase

the number of orders received by the courier and raise the long-term income.

2.4.2 Privacy-Preserving in Double and Dueling Deep-Q-Network With Differential Privacy

This part presents the related work for privacy-preserving in Double Deep-Q-Network and Dueling Deep-Q-Network and briefly explains our solution. Recently, there have been some researches that discuss privacy-preserving approaches for reinforcement learning. Tang et al. [97] proposed a method called Heda, which is privacy-preserving for machine learning that combines of differential privacy and a holomorphic cryptosystem that employs some techniques for managing privacy budget and reducing sensitivity. Balle et al. [14] proposed differentially private algorithms built on top of Monte Carlo methods for policy evaluation in the full MDP setting. Furthermore, [86] [117] [116] discuss preserving privacy in multi-agent system. Other research focuses on privacy-preserving approaches to protect neighbouring rewards from being distinguished in bandit problems, such as Tossou and Dimitrakakis [98], by using mechanisms to add noise to the estimates of the reward distribution. Also, Ma et al. [60] proposed differentially private mechanisms for $\epsilon - greedy$ and Softmax to achieve differentially private guarantees in the K-armed bandit problem. Wang and Hegde [104] discuss privacy-preserving approaches in reinforcement learning that use neural networks (DQN) in continuous space by protecting neighbouring rewards.

These researches did not apply the privacy-preserving in Double Deep-Q-Network and Dueling Deep-Q-Network methods proposed by van Hasselt et al.[102] and Wang et al.[108] as an improvement to normal DQN. In this work, we extend the research and apply differential privacy into Double Deep-Q-Network and Dueling Deep-Q-Network and inject noise into the gradient. Our research result has been presented by performing four experiments of Double Deep-Q-Network and Dueling Deep-Q-Network in order to demonstrate the ability to preserve privacy.

2.4.3 Protecting User Location in Online Food Delivery Services

Many scholars and experts have recently conducted much research on location and trajectory privacy protection. Mainly, location and trajectory privacy protection methods can

be classified into two main methods: differential privacy and anonymity or pseudonyms where researchers commonly use the model of k-anonymity to achieve trajectory protection in the anonymity [26] [123]. Moreover, deep reinforcement learning achieves great success in various areas, and some studies adopt deep reinforcement learning to enhance their solution to preserving location privacy. This part presents some previous studies of protecting location information and trajectory by using various methods and briefly explains our solution.

The anonymity method is prevalent for achieving privacy protection in location and trajectory, following some previous research in this field. Zhou and Wang. [124] propose a defence algorithm based on k-anonymity and fog computing. A scheme of trajectory protection has been designed for the protection of offline data in trajectory publication and the protection of real-time trajectory data with continuous queries. The mobility and local storage provided by Fog computing to guarantee physical control and the cloaking region for each snapshot constructed by k-anonymity. Khacheba et al. [47] propose a context-based location privacy scheme (CLPS). Their developed scheme provides a pseudonym changing strategy that lets the vehicle based on its context change pseudonyms. Also, the scheme provides a cheating attack or linkability threat that affects the proposed pseudonym changing strategy negatively. This mechanism aims to let vehicles to catch misbehaving vehicles that establish the attack of cheating and evaluate the success of the pseudonym change. Hwang et al. [41] proposed a comprehensive trajectory privacy technique and combined ambient conditions to protect the information of the location depending on the user privacy profile in order to avoid the user trajectory malicious LBS reconstructing. In the beginning, they preprocess a group of similar trajectories R to hide the actual trajectory of a service user by using the r-anonymity mechanism. They then combine k-anonymity with s road segments to safeguard user privacy. The sequence of the query issuing time for a service user breaks by a time-obfuscated technique to confuse the LBS. Zhang et al. [121] and based on the trusted anonymous server (TAS), propose a scheme of a trajectory privacy-preserving that aims not to allow the location-based service provider (LSP) to perform the inference attack. A group of requests generated by TAS meets the spatial k-anonymity of the user group. The TAS is the continued query that checks if the user is going to leave its security zone and determines if the group request needs to be resent in order to decrease the chance that the LSP rebuilds the actual trajectory of the user. Chiba et al. [28] propose an algorithm that in a certain range, when the position information is acquired their algorithm will mismatch the time

with the position information. They defined indicators that represent position distortions and information of the time. Tu et al. [99] focus on semantic attacks, as if the data of trajectory is published without appropriate handling could lead to severe privacy leakage where the existing solutions did not provide adequate protection to protect against semantic attacks. This means that the attacker could obtain an individual's private information by using the semantics features of frequently visited locations in the trajectory. Therefore, they propose an algorithm that provides high privacy safeguards against semantic attacks and re-identification while keeping the data utility at the highest level

The differential privacy model has been favoured by many scholars as it has a rigorous mathematical background, following some previous studies that use differential privacy with location. Andres et al. [9] propose a formal privacy notion of geoind for location-based systems that protect the user's exact location by permitting to release of approximate information that is usually required to get a certain service. By adding to the user's location managed random noise, the geoind achieves privacy preservation. Deldar and Abadi. [31], study how not to increase the risk of a privacy breach and how different geographical map locations can meet the requirements of user individual privacy protection. To achieve user location preserving privacy, the personalized-location differential privacy (PLDP) concept was introduced for the database of trajectory. A personalized noisy trajectory tree is used by PLDP-TD, which is built from the underlying trajectory database to provide a response to statistical queries by using the differential private method. Zhao et al. [123] proposed based on clustering and using differential privacy, a novel trajectory privacy-preserving method. In the cluster, to prevent the attack of continuous query, a Laplace noise is added to the count of trajectory location. The radius-constrained Laplace noise is added to the trajectory location data in the cluster to avoid too much noise affecting the clustering. The noise clustering centre in the cluster is obtained according to the noise location data and the count of noise location. Yang et al. [111] consider the issue of user location privacy protection. As the centralized server requires to get each user location precisely to ensure optimal task allocation, this will raise a privacy concern of exposing the workers' exact locations. To tackle this issue, a crowdsensing data release mechanism that meets differential privacy has been proposed to provide strong protection of worker locations. Pang et al. [75] consider the Internet of vehicles and location privacy. The vehicle user is able to offload computing tasks in different areas to MEC servers by using the wireless channel. The vehicle, when

performing the task, has to send the real location. This could cause the vehicle location privacy disclosure, in this sense they used differential privacy to protect the location privacy information.

Moreover, some studies adopt reinforcement learning or deep reinforcement learning in their data protection method. The following presents some of these studies. Min et al. [63] proposed a sensitive semantic location privacy protection scheme based on reinforcement learning. This scheme selects the perturbation policy based on the sensitivity of the semantic location and the attack history and uses the idea of differential privacy to randomize the released vehicle locations. A deep deterministic policy gradient based semantic location perturbation scheme (DSLP) is developed to solve the location protection problem with high-dimensional and continuous-valued perturbation policy variables. Wang et al. [107] consider the privacy of the trajectory for VANET. Based on the differential privacy mechanism, they propose reinforcement learning (RL) to randomise the released vehicle locations and use reinforcement learning to select the obfuscation policy to protect the vehicle's semantic trajectory. Berri et al. [16] consider the issue of broadcast scheduling throughput optimisation from road-side units (RSUs) to vehicles with protecting the vehicles privacy by allowing them to submit obfuscated location information to the server and use reinforcement learning (RL) to learn suitable obfuscated policy. Chen et al. [26] consider vehicular ad hoc network vehicle trajectory protection. For vehicular ad hoc networks, they proposed a scheme of optimized privacy differential privacy with reinforcement learning. To achieve a good balance between semantic security and geolocation obfuscation, the privacy budget allocation is dynamically optimised by the proposed schema for each location on the vehicle trajectory. The experiment results show that their scheme can ensure the balance between utility and privacy and decrease the risk of geographical and semantic location leakage. Erdemir et al. [37] study the trade-off between privacy-utility in location sharing such as LBS. They propose the mechanism of information theoretically optimal privacy preserving location release, which focuses on temporal correlations. They reformulate the problem as a Markov decision process (MDP) to tackle the history-dependent mutual information minimization. Zhang et al. [119] consider providing the balance between availability and privacy in LBS services. By using reinforcement learning and differential privacy, they developed Hasse Diagram Sensitivity to balance availability and privacy in LBS services. Min et al. [64] consider the privacy protection mechanism for 3D location by perturbing the location of the user based on 3D geoindistinguishability. To balance the quality of

service and privacy protection and based on reinforcement learning they proposed a scheme to alter the perturbation policy.

The user location information is vital, and it is necessary to protect this information and not disclose it. Multiple research has studied this issue, and a method such as k-anonymity and differential privacy has been used to safeguard the user's location. We considered protecting user location information in online food delivery services and proposed two different methods for this purpose.

The first method is the Protect User Location Method (PULM) which aims to protect the user's location in online food delivery services. The PULM employs differential privacy and injects Laplace noise into the user location along with the courier trajectory. This method considers two crucial factors, the city area size and customer frequency of online food delivery orders to identify the amount of injected noise. The second method is the Protect Trajectory and Location in Food Delivery method (PTLFD) which aims to protect the customer location information in online food delivery services. This method leverages multi-agent reinforcement learning and differential privacy Laplace mechanism to obfuscate the customer location and trajectory of the courier.

Part II

CHAPTER 3

MULTI AGENT REINFORCEMENT LEARNING FOR ONLINE FOOD DELIVERY

Today, online food delivery services are considered essential and this industry has attracted significant attention worldwide. Many companies and individuals are involved in this field, as it provides a good income and numerous job opportunities. In this chapter, we delve into the problem of online food delivery services and how the number of orders can be increased, thereby increasing long-term income for couriers. Multi-agent reinforcement learning is employed to guide couriers to areas with a high demand for online food delivery orders. The map of the city is divided into small grids, each grid representing a small area of the city. The agent has to learn which grid has the high demand for online food delivery orders to select. To demonstrate the results of this experiment, two datasets were used, Shenzhen, China and Iowa, USA.

3.1 Introduction

Emerging advanced technologies such as smartphones have grown rapidly and substantially impacted customer behaviour in online shopping, specifically during the COVID-19 pandemic. Today, online food delivery businesses are considered one of the most widespread businesses worldwide that have grown globally. It is expected that online food delivery will grow to 2.5bn users by 2027, and it is expected that in the grocery delivery segment, the average revenue per user will be US$449.00 in 2023 [93]. Today,

many people, especially in urban areas, do not have enough time to prepare meals for many reasons such as long working hours hence they often turn to online food delivery services which connect restaurants or food outlets with couriers, who then deliver the food to the customer.

Online food delivery applications are considered essential for many people nowadays. However, many of these applications experience many operational issues that reduce their efficiency. In the city, each area has a different number of food delivery orders compared with other areas. Most food delivery couriers rely on their experience to find areas with high food delivery order demands. However, sometimes they may go to areas with low demand. This issue could reduce the number of orders received by the couriers and lead to a decrease in their income in the long term and can increase the customer waiting time in a busy area as there are not enough couriers, which could reduce customer satisfaction. This is why many companies seek to improve their applications and attempt to increase the number of acquired food delivery orders to help to increase the company and couriers' income, reduce the waiting time for customers in busy areas and increase customer satisfaction.

This study introduces a method based on multi-agent reinforcement learning for online food delivery services that utilizes two multi-agent reinforcement learning algorithms. The primary objective of this method is to increase the number of received food delivery orders and increase the long-term income for the courier. This method also helps to reduce the waiting time for customers in busy areas by guiding couriers to areas with a high demand for food orders. The map of the city is split into small grids, and each grid represents a small city area and the agent has to learn to locate the area with high food delivery order demands. This approach enables the courier to find an area with a high demand for food delivery orders makes the courier get more orders and helps them to increase their long-term income [5].

This work considers online food delivery services and how to increase the number of received orders and increase long-term income. The contributions of this chapter are as follows:

- Improving the efficiency of online food delivery applications by guiding the couriers to areas with high demand, thereby increasing their income in the long term.

- To obtain more results and to make more comparisons, this research employs two multi-agent reinforcement learning methods, QMIX and IQL, to increase the courier's number of received orders and increase their long-term income.
- Use two datasets with different city sizes and different geographic areas to obtain more results.
- Consider weekdays and weekends factors in the agent's learning process to obtain better results as the number of orders can vary on weekdays than at weekends.

The remainder of this chapter is organized as follows. Section 3.2 presents the problem statement. Section 3.3 details the methodology. Sections 3.4, 3.5 and 3.6 represent the experiment design, results and conclusion.

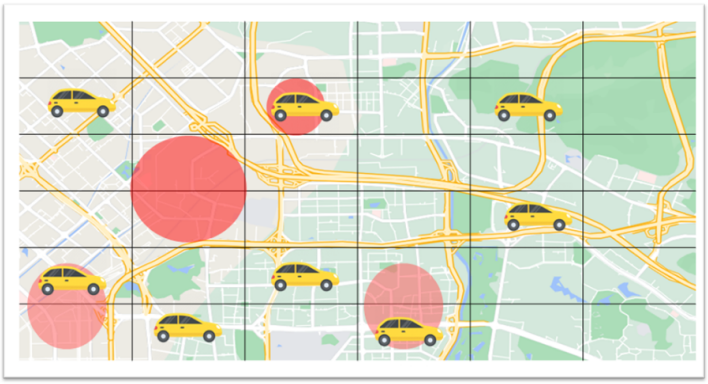


Figure 3.1: Overview of the model. The yellow car represents the delivery cars, and the red circle represents areas with a high demand for food delivery orders

3.2 Problem Statement

This section presents a detailed explanation of the problem statement, model overview and multi-agent reinforcement learning formulation, such as agent actions, reward function and state transition.

3.2.1 Model Overview

In most cities, particularly large cities, the demand volume varies from one particular area to another and from time to time. To increase the number of received online food delivery orders, many food delivery couriers try to locate areas with a high food delivery

demand based on their experience. However, sometimes they may go to an area with a low food delivery demand and receive fewer food delivery requests.

To tackle this issue, we propose a solution that aims to increase the number of orders received by the courier and thereby increase the courier's long-term revenue. This also could help minimize the waiting time of the customer during rush hours as there will be enough couriers in areas with a high demand for food delivery orders. We leverage multi-agent reinforcement learning to increase the number of orders received by the courier, using two multi-agent reinforcement learning algorithms, QMIX and IQL.

We consider the urban area and specific area of the city, which we then divide into multiple parts. We assume this area to be a rectangular shape that can be split into $N \times M$ cells. Every divided cell indicates a small area in the city, as shown in Fig 3.1. The yellow car represents the couriers, and the red circle represents areas with a high demand for food delivery requests. The courier of food delivery is waiting and receiving the delivery request from a customer without any previous data as customer data is only available after the customer requests the service. The courier has to learn to select the cell with a high number of food delivery requests based on an analysis of the collected data by an agent. Each time, the courier tried to improve himself by selecting the most appropriate cell to increase the number of received orders and increase long-term income.

3.2.2 Multi Agent Reinforcement Learning Formulation

To design our model, we used the Markov decision process where the transition probability in our case is unknown. The idea of model-free is more appropriate for such situations. Details of the main components of the model and the Markov decision process formulation are given in the following sections.

3.2.2.1 Agent

In our environment, more than one agent created $\{A_1, A_2, ...A_n\}$, to allocate the learning policy to the agent. In this multi-agent reinforcement learning problem, the courier is considered to be the agent. The agent's main goal is to select the cell with high demand for food delivery orders, which enables the courier to increase the number of food delivery requests received and thereby increase their income. Initially, the agent's

policy is uncertain and requires interaction with the environment to learn. The agents then evaluate his policy and iterate consecutively.

3.2.2.2 State

The global state s_t is maintained at each time t, and the courier's spatial distributions are considered. In our dynamic model, the states are defined based on grid location and time. The state space for our problem is characterized as follows:

- **The grid location**: The certain area of the city map is considered and divided into $N \times M$ cells. Every divided cell indicates a small area in the city. The agent interacts with the environment and updates his status, the agent then analyses and computes the available information about the environment and tries to select the cell with a high food delivery demand.
- **Weekdays and weekends**: On the one hand, during weekdays and weekends, different situations need to be notable and considered in the model. Therefore this model considers the weekdays and weekends during the learning process for better performance.

3.2.2.3 Action

For each state, the agent chooses the cell that will enable the courier to receive more food delivery orders and thereby increase their long-term revenue. The map is split into $N \times M$ cells, and each cell represents a particular area that may have high demand or not or perhaps there is no restaurant at all. The model analyses and predicts the cells that have a high food delivery demand based on the current information. The agent has to perform an action on this by selecting the most proper cell to increase his rewards. In this case, the agent has $N \times M$ actions and needs to choose one cell at each time.

3.2.2.4 Reward Function

The agent in the environment performs the action, and accordingly, the agent receives the rewards from the environment. Only if the agent is able to increase the number of received orders and reach or above the approximate average number A the received reward is positive; otherwise, the reward is negative, and this can be defined based on the following:

$$r(k) = \begin{cases} 1, & if\ \mathcal{N} \geq A\ . \\ 1-, & \text{Otherwise}\ . \end{cases} \tag{3.1}$$

$r(k)$ is the function of reward, $\mathcal{N}$ is the number of orders, and A is the approximate average number of orders for every agent.

3.2.2.5 State Transition

In our setting, the action is passed from agents and received by the environment. The state s_t then is generated, and the new state is observed by the agent and takes collective action a_t. This action is evaluated by the environment, and the environment returns a reward r_t. A tuple of state transition $\{s_t, a_t, r_t, s_{t+1}\}$ is formed at this point, and then multi-agent reinforcement learning is used to find the relationships between these tuples.

3.3 Methodology

3.3.1 Dueling Network Architecture

Dueling DQN is a deep reinforcement learning algorithm model-free solving Bellman equation iteratively [113], proposed in 2015 by Wang et al. [108] as an improvement to DQN. The main insight of this new architecture is that there is no need to estimate the value of each action choice for many states. For example, knowing whether to move left or right in the Enduro game setting only matters when a crash is close. It is essential to know which action to take in some states, while in other states, the chosen action does not affect what happens [108]. In this algorithm, the main improvement is that the Q-values $Q(s,a)$, which the network tries to approximate, can be split into two parts: the advantage of actions of the state $A(s,a)$ and the value of the state, $V(s)$. Based on the definition $Q(s,a) = V(s) + A(s,a)$, the advantage $A(s,a)$ is assumed to bridge the gap from $A(s)$ to $Q(s,a)$. We could suppose advantage $A(s,a)$ is delta, telling us how much reward we can earn from each specific action in each particular state. In general, the advantage could be negative or positive or can be any magnitude [49]. There is no sequential architecture such as deep learning in Dueling DQN. The model layers are divided into two different streams after the convolutional layers. Each stream has its own fully connected layer and output layers [84]. The outputs of the two separate estimators can be integrated as follows:

$$Q(s,a;\theta,\alpha,\ \beta)\ = V(s;\theta,\beta)+\left(A(s,a;\theta,\alpha)-\frac{1}{|\mathscr{A}|}\sum_{a'}A(s,a';\theta,\alpha)\right) \tag{3.2}$$

The parameters of the convolutional layers are denoted by θ, while α and β are the parameters of the two streams of fully-connected layers.

3.3.2 QMIX Algorithm

Every agent selects an action in a multi-agent framework resulting in collective action a_t, and then the global, immediate reward r_t is shared, which evaluates the collective action taken previously. For the collective action, there is collective agent-value function $Q_{tot}(s_t,a_t)=\mathbb{E}_{s_t+1:\infty,a_t+1:\infty}[R_t|s_t,a_t]$, where at time t the discounted return is R_t. One of the main challenges in multi-agent reinforcement learning is how to assess every agent contribution individually and accurately to get a separate value function from the collective action-value function. The agent A_i individual value functions are represented by $Q_i(o_t,a_t)$.

The QMIX is an advanced technique of value-based aims to train uncentralized policies in an end-to-end centralized method [79]. To perform this, it needs to overcome two challenges by QMIX: the first challenge is the process of centralized training, where every agent needs to calculate the influence action based on a single global reward. The second challenge is that we need to guarantee that the ensemble of the optimal actions of agents is the optimal action of the ensemble of agents when agents interact in a decentralized manner [122]. There is an assumption that the QMIX can interpret the total action-value function as the mixing network of each action-value function or a linear combination of each action-value function. Thus, it is expected that the QMIX-based method acts in cooperation with nearby agents. [118].

3.3.3 Independent Q-Learning IQL

Independent Q-Learning (IQL) was proposed by Tampuu et al. [96]. The Independent Q-Learning is an algorithm that abandons centralized training where for each agent in IQL, the Q-learning is performed separately. In some way, this method avoids the implementation issue of the CTDE framework. Consequently, in a large probability, the single agent action can interfere with the overall environment and other agents of the state. Therefore, this makes the IQL can't converge in a complex environment. The IQL

Algorithm 1 QMIX algorithm for online food delivery environment

With size $\mathcal{N}$ initialize reply buffer $\mathcal{D}$
With random weight θ initialize action value function
With $\bar{\theta} \leftarrow \theta$ initialize target action value function
Mixing network ϕ is initialized
while Episodes training not finished **do**
    Environment reset
    while the state not in termination **do**
        for agent in Agents **do**
            Observing the state o_i^t by the agent
            Select a random action a_t with probability ε
            Else select $argmax_{a_t}(o_t, a_t : \theta)$
        end for
        Perform action a_t, move next state s_{t+1} and obtain reward r_t and termination information
        Insert (o_t, a_t, s_{t+1}, r_t) to $\mathcal{D}$ reply buffer
        Random sample $\mathcal{K}$ minibatch from $\mathcal{D}$
        $Q_{tot} = mixing((Q_1(o_t^1, a_t^1) + Q_2(o_t^2, a_t^2) + ...Q_n(o_t^n, a_t^n))$
        Loss function computed $\mathcal{L} = \left(r_t + \gamma max_{a_t} Q_{tot}(s_t, a_t|\bar{\theta}) - Q_{tot}(s_t, a_t|\theta)\right)^2$
        On $\mathcal{L}$ implement gradient descent step to the network parameter θ
        Updating the $\bar{\theta}$ parameter of the target network
        Rate ϵ of exploration updated
    end while
end while

algorithm still has a number of applications in small scenarios of reinforcement learning applications such as Atari games.

In IQL, each agent trained decentralized Q-functions, while QMIX is a method learned in end-to-end of decentralised policies in a centralised setting. QMIX is composed of networks of agents representing every Q_a, and a mixing network that combines them into Q_{tot}. The Independent Q learning (IQL) is the most commonly applied method in multi-agent learning, and it consists of a group of concurrent single agent that use and share the same environment where every agent learns individually [118] [79] [80] [34]. In our research, we intend to use different multi-agent reinforcement learning settings in order to show more results from our experiments.

3.3.4 Multi Agent Reinforcement Learning for Online Food Delivery

An overview of the QMIX algorithm for online food delivery has been presented in algorithm 1. During the training process and using gradient descent, the loss function is minimized by considering parameter θ at iteration i, where the neural network is employed as an approximator in this process. In the beginning, with size $\mathscr{N}$, we initialize reply buffer $\mathscr{D}$. In lines 2 to 4, with random weight θ, we initialize the action value function and copy weight $\bar{\theta} \leftarrow \theta$ and initialize the target action value function and then initialize the QMIX mixing network. In lines 5 and 6, if the episode has not ended, the environment then resets. In lines 9 to 11, the current observation is observed by the agents, and the agent then chooses the action based on probability ϵ, else chooses the highest value from $argmax_{a_t}(o_t, a_t : \theta)$. In line 13, the action is performed and the corresponding rewards are obtained, then the agent observes the next state. In line 14, the experience tuples are inserted into memory such as the action taken, the state, the next state and the obtained reward. From the reply buffer $\mathscr{D}$, the sample is extracted and loss function $\mathscr{L}$ is computed. In line 18, implement the step of gradient descent. In lines 19 and 20, the parameter $\bar{\theta}$ of the target network and exploration rate ε are updated.

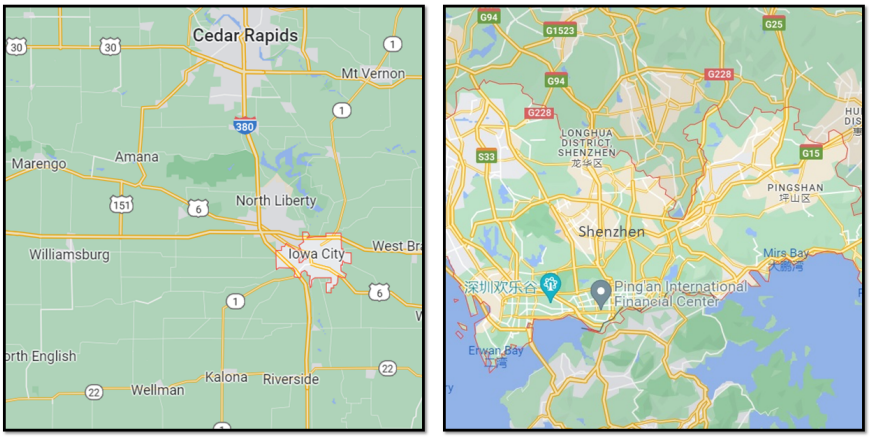


Figure 3.2: Comparison of size of Iowa (left) and Shenzhen (right)

3.4 Experiment Design

The following section gives a breakdown of details on how each experiment of online food delivery method was set up. We conduct some experiments to show how the agent

Table 3.1: Sample records from the Shenzhen dataset

Seq	Day	Sender lat	Sender long	Receiver lat	Receiver long	Send hour	Send min	Receive hour	Receive min	Order cnt
0	0	22.47	113.91	22.46	113.89	11	18	11	32	1
1	0	22.48	113.89	22.48	113.89	9	31	9	41	1
2	0	22.48	113.89	22.51	113.89	11	25	11	57	1
3	0	22.48	113.91	22.48	113.89	18	28	18	52	1

can accumulate rewards using different algorithms of deep reinforcement learning on various datasets and by how much the agent can increase their received number of food delivery orders over time.

3.4.0.1 Applyin Different Deep Reinforcement Learning Algorithms

An extensive experiment was conducted to illustrate the results of using multi-agent reinforcement learning for online food delivery services. Two multi-agent reinforcement learning algorithms were used with one single agent. We used QMIX and IQL to implement multi-agent reinforcement learning and Dueling Deep Q Networks to implement single agent. We attempted to simulate the results from a single agent by multiplying the results from a single agent to be equal to the number of agents in multi-agent reinforcement learning. We run these algorithms with different datasets from different cities that have different city sizes. In each episode, the agents could run multiple times. The datasets that were used are Shenzhen, China and Iowa, USA. Moreover, we apply the algorithm to synthetic dataset (random data) to show more comparisons and the effects of our proposed method.

3.4.0.2 Number of Food Delivery Orders

In this experiment, we implement multi-agent reinforcement learning for the online food delivery method, showing the number of orders the driver can obtain and how much increase in revenue the driver can receive over time. QMIX was used on two datasets, Shenzhen and Iowa. The results of the total number of orders for three agents were shown. The experiment demonstrates to what extent all agents can gain more orders in Shenzhen and Iowa datasets.

3.4.1 Dataset Description

Two different datasets with different city area sizes were used for our experiments to test the approach for multi-agent reinforcement learning. Figure 3.2 shows the difference

Day	Sequence	Time	Customer location	Restaurant number	Time food ready
1	1	1	41.63790381, -91.50545655	67	5
1	2	1	41.68937752, -91.49441381	47	11
1	3	3	41.69378731, -91.57940298	22	10
1	4	4	41.64175802, -91.57129212	56	8

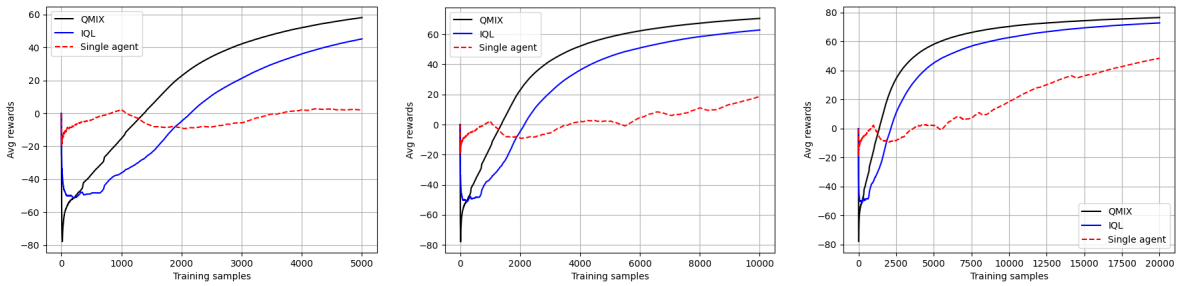
Table 3.2: Sample records from the Iowa dataset

in the size of Iowa compared to Shenzhen.

The first dataset for Shenzhen, China, provided by Alibaba Local Service Lab [32] contains on-demand food delivery order data. This dataset has 1048575 orders and 93 restaurants, with the location of each restaurant and the delivery destination locations and time of pickup of the meal from restaurants and delivery time, table 3.1 shows a sample of the dataset records. The second dataset used in our experiment is provided by Ulmer et al. [100]. This dataset contains data of meal delivery services in Iowa, USA. In this dataset, there are 111 restaurants and 1200391 delivery records, table 3.2 shows sample records of this dataset. In this dataset, the actual location is used with random generation of order and equal request probability for each point of time and location. For both cities, we consider a particular area to conduct our experiment. The data may have been modified from the source for certain purposes, such as obfuscating the exact location or any other reasons.

3.5 Experiment Results

In multiple domains, it has shed light on deep reinforcement learning to resolve complicated sequential decision-making problems [122]. In this section, we detail our result by using the multi-agent reinforcement learning QMIX and IQL along with a single agent of Dueling Deep Q Networks on different datasets, and we demonstrate how much the couriers can increase the number of food delivery orders they receive.

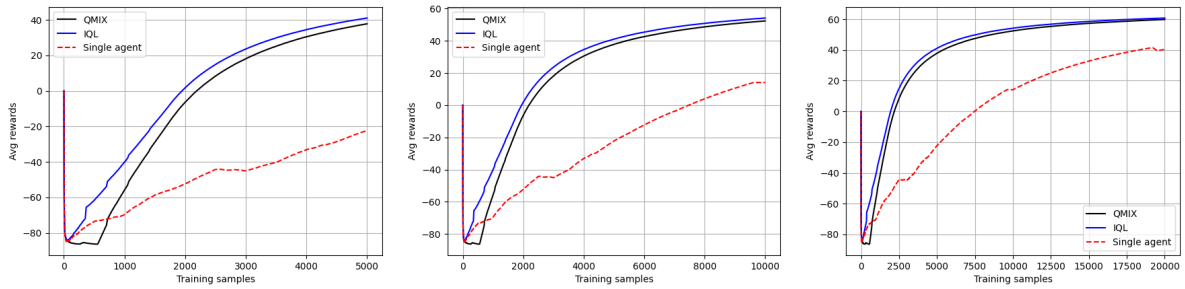


((a)) ((b)) ((c))

Figure 3.3: Comparison of results of the average accumulated reward of multi-agent reinforcement learning for the online food delivery method using the Shenzhen, China dataset with two multi-agent reinforcement learning methods QMIX, IQL and single the agent Dueling Deep Q Networks, with different runs of 5000, 10000 and 20000.

3.5.1 Applying Different Deep Reinforcement Learning Algorithms

Figure 3.3 shows the results of the average accumulated reward using QMIX and IQL, represented by black and blue lines, respectively and a single agent of Dueling Deep Q Networks represented by the red line. The Shenzhen, China dataset was used in this experiment, and we show the result of different runs of 5000, 10000 and 20000. The agents in this experiment increase their accumulative rewards over time and are able to reach the highest level. The single agent increases his accumulative rewards over time, which shows that multi-agent reinforcement learning achieves higher results. This experiment has been repeated in the Iowa, USA dataset and a synthetic random dataset. Figures 3.4 and 3.5 show that agents can increase their average accumulated rewards over time and be able to reach the highest level whereas in Figure 3.4 the agent is able to increase the average accumulated rewards from about -80 to about 60 and Figure 3.5 the agent able to increase the average accumulated rewards from about -40 to about 40. The single agent increases his accumulative rewards over time, which demonstrates that multi-agent reinforcement learning achieves higher outcomes in comparison with a single agent.

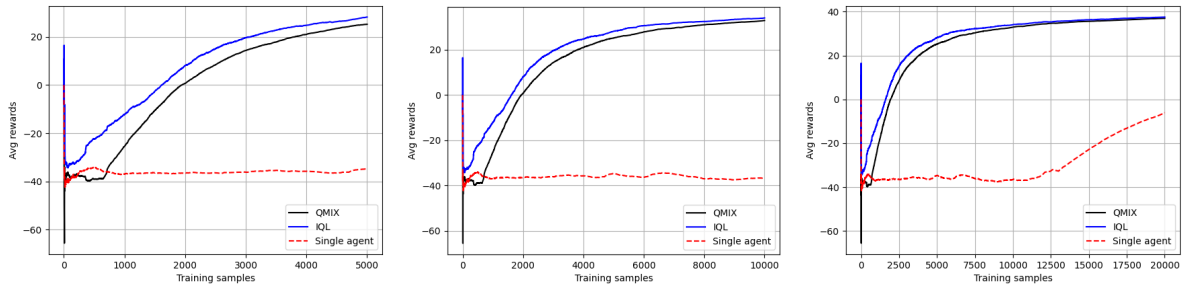


((a)) ((b)) ((c))

Figure 3.4: Comparison of the results of the average accumulated reward of multi-agent reinforcement learning for the online food delivery method using Iowa, USA dataset with two multi-agent reinforcement learning methods QMIX, IQL and single agent Dueling Deep Q Networks, with different runs of 5000, 10000 and 20000.

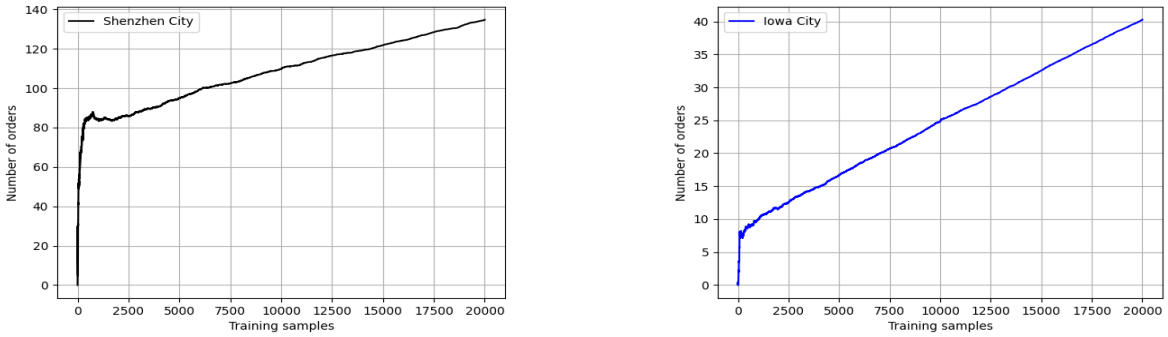
3.5.2 Number of Food Delivery Orders

Figure 3.6 illustrates the average number of food delivery orders that the driver can gain after implementing multi-agent reinforcement learning for online food delivery methods using QMIX with the two datasets for Shenzhen and Iowa. It shows that the agents were able to increase the number of orders from 80 orders to 140 orders for all agents in the Shenzhen dataset and from 10 to 40 in the Iowa dataset for all agents.



((a)) ((b)) ((c))

Figure 3.5: Comparison of results of the average accumulated reward of multi-agent reinforcement learning for the online food delivery method using a random dataset with two multi-agent reinforcement learning methods QMIX, IQL and single agent Dueling Deep Q Networks, with different runs of 5000, 10000 and 20000.



((a)) ((b))

Figure 3.6: The average number of food delivery orders that the driver can receive after implementing multi agent reinforcement learning for the online food delivery method using QMIX with the two datasets Shenzhen and Iowa.

3.6 Conclusion

This research considered how to increase the number of online food delivery orders received by couriers and thereby increase their income in the long term. Multi-agent reinforcement learning trained and guided the agent to areas with a high food delivery order demand. The city map was divided into small grids, and each grid represented a certain area of the city. The agent had to learn which grid had the highest food delivery demand and had to select the most appropriate grid that can increase the number of received orders. We used two datasets, Shenzhen, China and Iowa, USA, to demonstrate our experiment results. The experiment result showed there was an increase in the average accumulated reward using QMIX and IQL in Shenzhen and Iowa datasets and the agent was able to increase the average number of orders from around 80 orders to around 140 orders for all agents in the Shenzhen dataset and from around 10 to around 40 in the Iowa dataset for all agents.

CHAPTER 4

PRIVACY-PRESERVING IN DOUBLE AND DUELING DEEP-Q-NETWORK WITH DIFFERENTIAL PRIVACY

With extensive applications and remarkable performance, deep reinforcement learning is becoming one of the most important technologies. Many applications use deep reinforcement learning, such as robotics, recommendation systems, and healthcare systems. These systems could collect data about the environment or users, which may contain sensitive information and therefore, pose a real risk when these data are disclosed. With the success of deep reinforcement learning specifically Deep Q Networks(DQN) many other enhancements have been presented and show more advance such as Double DQN, Dueling DQN and Prioritized experience replay. In this chapter, we aim to preserve the privacy of Double and Dueling Deep-Q-Network models by adopting the method of differentially private SGD and by injecting Gaussian noise into the gradient. To demonstrate our results, we applied our method in four separate settings, and we used various amounts of noise with different numbers of a run for each setting in order to show the effectiveness of using this method.

4.1 Introduction

Deep reinforcement learning has had a significant impacts on the development of artificial intelligence (AI) over the past few years, and it is considered one of the most crucial machine learning research directions [59]. The technique of deep reinforcement

learning aims to maximize the acquired rewards and reach desired goals via enhancing the policy iteratively when the agent interacts with the environment [113]. The agent of deep reinforcement learning uses trial-and-error, interacts with its environment and recognizes the results of its actions, which enable it to learn and alter its behaviours in response to the rewards received [11]. The game of deep reinforcement learning, such as ATARI2600 games, received a large part of the advancement and development. However, other applications such as health systems [7], search engines [81], recommendation systems [53], and robotic control [12] have also received a lot of attention and development. With the success of deep reinforcement learning specifically Deep Q Networks(DQN) many other enhancements have been presented such as Double DQN [102], Dueling DQN [108] and Prioritized experience replay [83].

However, privacy issues have been raised recently, especially with applications that include sensitive information such as medical [77] or financial [17]. Recently many researchers have focused their efforts on these issues. The trained policy could be released to the client side, which would enable the adversary to infer the demographic information from the policy. Moreover, in the model, there are many parameters, some which contain sensitive information implicitly, allowing the adversary to infer this sensitive information. Hence, maintaining privacy is very important. Some studies show the vulnerability of deep reinforcement learning, such as the work conducted by Pan et al. [74], which shows how private information could leak and the ability of the adversary to infer sensitive information. In their experiment, the agent in the Grid World has a task to navigate from its current place to a specific destination and avoid colliding with obstacles. Figure 4.1 shows the environment where, the red box denotes the agent, the grey boxes represent obstacles and the green box shows the goal of the agent. After training the agent, it was able to follow the red line to the goal and based on their observations in the experiment, they found that even when they took out all the obstacles, the agent followed the same trajectory. This experiment shows that there is a leak of private information in the Grid World, as the trained policy will display the optimal actions on the actual map without seeing the map. This experiment shows that they are successfully able to infer floor plans from some trained Grid World with a recovery rate of 95.83%.

In this work, we consider Double DQN and Dueling DQN, where the agent interacts with it is environment for the period of a fixed length episode. The state reveals to the agent at each time step, and the agent responds with suitable action and receives

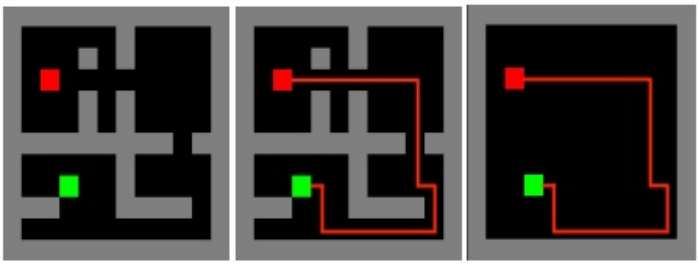


Figure 4.1: The environment of Grid World. Map of Grid World (left), trained agent (middle), and environment after removing obstacles (right). Pan et al. [74]

a corresponding reward. We concentrate on situations when the states and rewards provided by the user to the agent could contain sensitive information [104]. Therefore, by adopting privacy preserving algorithm, we aim to maintain privacy in this particular and by using differential privacy to inject Gaussian noise into the gradient during the training. The following is a list of the main contributions:

- Adopt differential privacy for the Double DQN method to guarantee privacy level.
- Adopt differential privacy for the Dueling DQN method to guarantee privacy level.
- To test the performance of our method, we conduct extensive experiments.

The remainder of this chapter is organized as follows. Section 4.2 overviews the methodology. Sections 4.3 present the experiment design. Sections 4.4 and 4.5 detail the results and the conclusions.

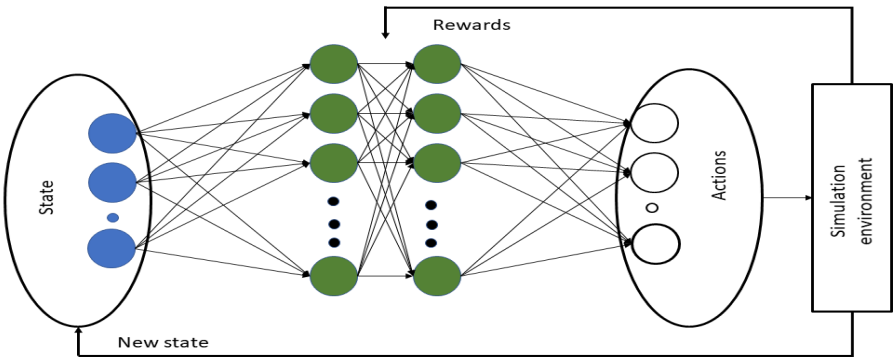


Figure 4.2: Double DQN interacting with its environment: Double DQN uses two neural networks and aims to reduce Q-Value overestimations by splitting the max operator into action selection and action evaluation.

4.2 Methodology

4.2.1 Overview of Double Deep Q Network privacy method

Double DQN shows good results on ATARI2600 games and achieves good performance in overcoming the overestimation problem that experienced by DQN. As the same Q function is used for both the selecting and evaluating action in DQN, the overestimation of the value could occur, which could lead to a decrease in the performance of DQN. The Double DQN uses two neural networks in its architecture: the primary network and the target network. In our experiment, each network has three hidden layers. Figure 4.2 shows the Double DQN architecture and how it interacts with its environment. In this experiment, the agent uses the epsilon-greedy method, which gives the agent the opportunity to explore and learn from a random action. To do this, the epsilon is defined to start from 1 to 0.1 with a decreased rate of 0.999. A random number is generated by the code at each step. If the epsilon value is greater than the generated random number, then the action will be chosen randomly else, the model will choose the optimal action.

Algorithm 2 High level overview of Double Deep Q Network model

Initialize: Primary network θ
Initialize: Target network θ^{-}
Initialize: Reply buffer $\mathscr{D}$ with size $\mathscr{N}$
while Episodes training not finished **do**
    Reset
    while The state not in termination **do**
        Observe state s_t
        With probability ε select a random action a_t
        Otherwise select $a_t = argmaxQ(s_t, a : \theta)$
        Execute action a_t and observe next state s_{t+1} , receive reward $r_t = R(s_t, a_t)$ and termination info
        Add (state s_t, action a_t , reward r_t , next state s_{t+1}) to reply buffer $\mathscr{D}$
        Sample random minibatch (s_j, a_j, r_j, s_{j+1}) from $\mathscr{D}$
        If episode terminates set $Y_j = r_j$ otherwise
        $Y_j = r_j + \gamma Q(s_{j+1}, \underset{a}{\text{argmax}} Q(s_{j+1}, a_j; \theta_j); \theta_j^{-})$
        Call DPSGD
        Perform a gradient descent step
        In every $\mathscr{C}$ step update target network parameters
    end while
end while
Output : Trained DDQN

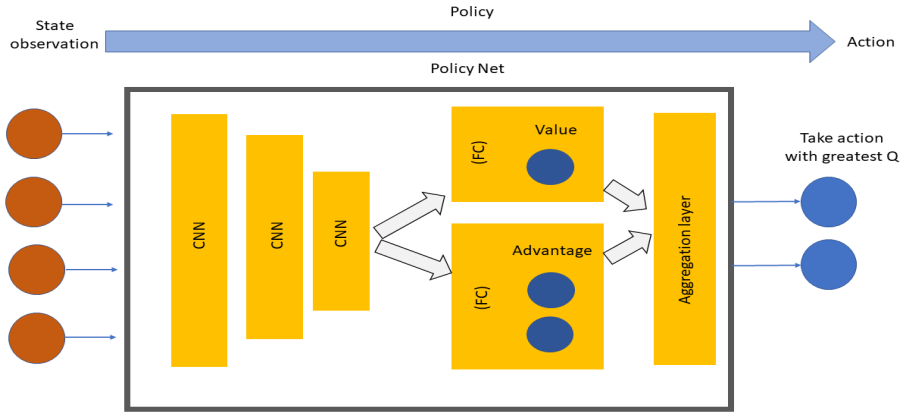


Figure 4.3: Dueling DQN architecture and how it interacts with its environment. The network is split into two separate streams, one for estimating the state value and the other for estimating the state-dependent action advantages after that, the two streams are combined by an aggregation layer.

Algorithm 2 presents the high-level overview of the Double DQN model that is used in our experiment and demonstrates the training process in the model. First, in lines 1-2, with the random weight, the primary neural network and target neural network are initialized. The reply buffer memory $\mathscr{D}$ is initialized with a size of $\mathscr{N}$ in line 3. The neural network is trained extensively according to the input data. The Double DQN mode has two neural networks which play an essential role in selecting and evaluating the actions in the system, respectively. As the epsilon greedy method is used, the action is selected randomly or by the model prediction based on the epsilon value. In line 11, after the action is executed, the action, reward, current state, and next state are stored in the reply buffer. In line 15, the DPSGD is called to perform gradient descent with privacy-preserving by using the differential privacy technique along with a moments accountant method for tracing the privacy loss that allows the deep neural networks to train under a modest privacy budget. In line 17, after a certain number of iterations, the target network parameters are updated.

4.2.2 Overview of Dueling Network Architecture privacy method

The Dueling DQN was proposed as an improvement to normal DQN in 2015 by Wang et al. [108]. The Dueling DQN is a combination of Dueling DQN and Double DQN. Therefore, it is able to solve the overestimation problem that exists in normal DQN, as it uses the Double DQN method. The Dueling DQN is built based on a Markov decision

process as a standard reinforcement learning method [49]. Figure 4.3 shows Dueling DQN architecture and how it interacts with the environment. In the Dueling DQN neural network structure, there are two fully connected layers streams after the CNN layers to estimate advantage function $A(s,a)$ and value function $V(s)$ separately. The two separate streams are combined to estimate action value function $Q(s,a)$ [52]. In our experiment, the agent uses the epsilon-greedy method, which gives the agent the opportunity to explore and learn from random actions. To perform this, the epsilon is defined to start from 1 to 0.1 for the Health System Simulations experiment and from 1 to 0.05 for the Ambulance Location Problem experiment with a decrease rate of 0.999. In each run step, a random number is generated by the model, and if the epsilon value is greater than the given random number, then the action will be chosen randomly otherwise, the model will choose the optimal action.

Algorithm 3 High level overview of Dueling Deep-Q-Network model

Initialize: primary network θ
Initialize: target network $\theta^{-} = \theta$
Initialize: reply buffer $\mathscr{D}$ with size $\mathscr{N}$
Initialize: exploration procedure as ε_t greedy policy.
while Episodes training not finished **do**
    Reset environment
    while The state not in termination **do**
        Observe state s_t
        With probability ε select a random action a_t
        Otherwise select $a_t = \ argmax\ Q(s_t,\ a\ : \theta)$
        Execute action a_t and observe next state s_{t+1} , receive reward $r_t = R(s_t, a_t)$ and termination info
        Add (state s_t, action a_t , reward r_t , next state s_{t+1}) to reply buffer $\mathscr{D}$
        Sample random minibatch (s_j, a_j, r_j, s_{j+1}) from $\mathscr{D}$
        calculate value function using
        $Q(s,a;\theta,\alpha,\ \beta)\ = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \frac{1}{|\mathscr{A}|}\sum_{a'} A(s,a';\theta,\alpha)\right)$
        Call DPSGD
        Perform a gradient descent step
        In every $\mathscr{C}$ step update target network parameters
    end while
end while
Output : trained Dueling DQN

The algorithm 3 presents the high-level overview of the Dueling DQN model that is used in our experiment and demonstrates the training process used in the model. First,

in lines 1-2 with the random weights, the primary neural network and target neural network are initialized. The reply buffer memory $\mathscr{D}$ is initialized with a size of $\mathscr{N}$ in line 3. The neural network is trained extensively according to the input data. As the epsilon greedy method is used, the action is selected randomly or by the model prediction based on the epsilon value. In line 12, after the action is executed, the action, reward, current state, and next state are stored in the reply buffer. In line 16, the DPSGD is called to perform gradient descent with privacy-preserving by using the differential privacy technique along with a moments accountant method for tracing the privacy loss that allows the deep neural networks to train under a modest privacy budget. In line 18, after a certain number of iterations, the target network parameters are updated.

4.2.3 Privacy Mechanism

Differentially private stochastic gradient descent (DPSGD) is a sophisticated approach to protect the privacy of training data and manage the added noise to protect the data without destroying its utility. In this work, we adopt DPSGD [3] with the Double and Dueling DQN algorithm to preserve privacy. The privacy is achieved by adding noise into stochastic gradient descent directly by minimizing the empirical loss function $\mathscr{L}(\theta)$ with parameters θ. By selecting a subset of random examples from the stored data of the reply buffer, at each step of SGD, the gradient $\nabla_\theta \mathscr{L}(\theta, x_i)$ is computed for each gradient clipping the ℓ_2 norm, we calculate the average and to protect privacy add noise and of this average noisy gradient take a step in the opposite direction. In the end, the privacy loss of the mechanism needs to be computed according to the information maintained by the privacy accountant.

Norm clipping: which is bounding the influence of each individual example on $\widetilde{\mathbf{g}}_t$ is required to prove and guarantee differential privacy. We clip each gradient in the ℓ_2 norm since there is no prior bound on the size of the gradient. For example, the vector $\mathbf{g}$ of the gradient is replaced by $\mathbf{g}/\max(1, \frac{\|\mathbf{g}\|_2}{C})$ for a clipping threshold C. The main purpose of this clipping is to ensure if the $||\mathrm{g}||_2 \leq C$, then the $\mathbf{g}$ is preserved, as it gets decreased to be of norm C if $||\mathrm{g}||_2 > C$. In DPSGD, consider each layer of multi-layer neural networks separately, which allows to set a different noise scales σ and clipping thresholds C for each layer. Furthermore, the noise and clipping parameters could be different with the number of training steps t. In a group of examples, the $\mathscr{L}$ gradient is estimated by calculating the gradient of the loss and obtaining the average. This average offers an estimator that is unbiased, the variance of which shrinks rapidly with the size

of the group. This group of examples is called a lot, the main purpose of giving this name is to distinguish a lot from the computational grouping that is usually named a batch.

Privacy accounting: is calculating the training overall privacy cost which is an important issue for differentially private SGD. Differential privacy composability allows us to perform " an accountant" procedure that calculates the privacy cost at each time and uses the training data, and accumulates this cost during the training. The gradient is typically required in training every step at multiple layers, and the accountant collects the cost corresponding to them.

Moments accountant: There are many researchers trying to study the privacy loss for a specific noise distribution and privacy loss composition. For the Gaussian noise, which is used by DPSGD if it chose σ to be $\sqrt{2\log\frac{1.25}{\delta}}/\varepsilon$, with respect to the lot, this will lead to be each step is (ε,δ) differentially private. As the lot is a database random sample, the theorem of the privacy amplification indicates that with respect to the full database, each step is $(O(q\varepsilon),q\delta)$ differentially private where $q = L/\!/N$ is the sampling ratio per lot and $\varepsilon \leq 1$. The DPSGD have moments accountant, which is considered a strong accounting method that lets us prove that the DPSGD is differentially private $\left(O(q\varepsilon\sqrt{T}),\delta\right)$ for specific chosen settings of clipping threshold and the noise scale.

4.2.4 Rationale for using DPSGD

The technique of DPSGD allows deep neural networks to be trained with a modest privacy budget, along with using the moments accountant method for tracing the privacy loss. The approach of differentially private versions of stochastic gradient descent is not a new approach there are some previous works such as Song et al. [89]. The DPSGD followed the same previous work with some number of modifications and extensions, such as moments accountant. Tracking privacy loss is a critical part of differential privacy, a new tool has come to tracking of the privacy loss in privacy is moments accountant. It permits the privacy loss of complex composite mechanisms to be tightly automated analysed. The DPSGD achieve high accuracy by training the differential privacy with deep neural networks and with a fair total of privacy loss, as it shows for MNIST experiments reach 97% training accuracy and 73% accuracy for CIFAR-10 with $(8,\ 10^{-5})$ - differential privacy [3]. In this research, we used DPSGD proposed by Abadi et al. [3] as it is robust and uses advanced tools such as moments accountant and showed good results in training deep neural networks with differential privacy.

4.2.5 Theoretical Analysis of DP-SGD

A query with sensitivity Δ and $\mathscr{L}2$ satisfying (ϵ', δ') with the mechanism of Gaussian need to add noise with standard deviation $\sigma' = \frac{\sqrt{2ln(1.25/\delta')}\Delta}{\epsilon'}$.

Thus with $\Delta = 2l, \sigma = \frac{16l\sqrt{T\ ln(2/\delta)\ ln(2.5T/\delta n)}}{n\epsilon}$, every noisy gradient is $(\frac{n\epsilon}{4\sqrt{2T\ ln(2/\delta)}}, \frac{\delta n}{2T})$-DP.

Next, in the choice of i_t using privacy amplification by subsampling take into account the randomness each noisy gradient is in fact $(\frac{\epsilon}{2\sqrt{2T\ ln(2/\delta)}}, \frac{\delta}{2T})$-DP.

We obtain that DP-SGD is (ϵ, δ)-DP as it is an adaptive composition of T DP mechanisms.

4.3 Experiment setup

4.3.1 Double DQN experiment setup

We performed our experiment on two different Double DQN applications to demonstrate the ability to preserve privacy by applying DPSGD [3] in Double DQN. The first experiment was applied to the Health System Simulations model of Double DQN [7]. This application was built as a simulation model of health systems and developed using the OpenAI Gym framework and PyTorch to build the agent. This health system work to simulate a real health system, and manages beds in hospitals such as those in emergency departments. Patients went to the hospital and stay for a certain time to receive treatment and then leave. When the patient arrives at the hospital, the Health System Simulations check the available staffed beds and allocates a bed to this patient. There is a limited number of beds available in the hospital at any time, and the system has to manage the number of available staffed beds with the number of patients and try to give each patient a bed. At any given time, the system aims to ensure at least 5% free of the number of staffed beds in the hospital. The agent has to learn how to minimize the loss for each unoccupied bed or patient who has no bed. The reward is calculated based on the currently available bed and targeted bed, and usually, the reward is zero or negative. The number of staffed beds can be requested to be changed by the system, and this request can be performed after two days. By default, the simulation runs for 365

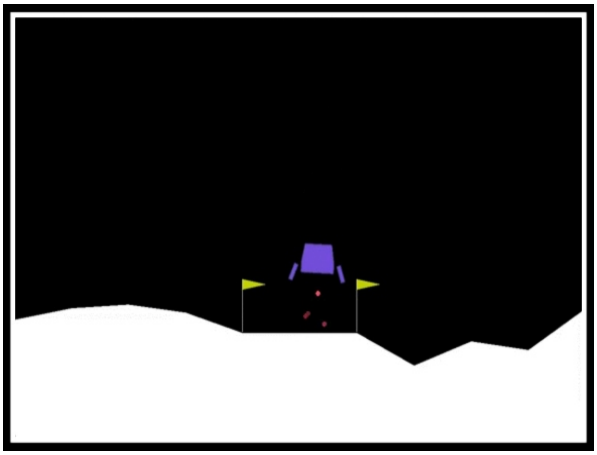


Figure 4.4: The Lunar Lander environment: where Double DQN used to solve lunar lander challenge to make the ship land softly on the ground

days. The number of patients who arrive at the hospital is not same all days, the default number of patients who arrived is 50/day. This number changes as the number of arrival of patients at the weekend is 50% of average arrival numbers, and the number of arrival of patients on the weekdays is 120% of average arrival numbers. The average stay of patients in the hospital is 7 days. In this experiment, four levels of noise were used to measure the effect of injected noise, the ability of the agent to learn with each different amount of noise and the ability to preserve privacy. The first time the agent runs with no noise, then it runs with 10^{-10}, 10^{-5} and 10^{-3} of noise respectively. In this experiment, the agent of Double DQN has been trained for 500, 1000, 2000 and 3000 runs in order to analyze the behaviour of the agent with each noise level and the different number of runs. In this experiment, have been used epsilon-greedy to give the agent the opportunity to explore and learn from random actions. The epsilon is defined to start from 1 to 0.1 with a decreasing rate of 0.999, and we use 0.9 of the moments accountant.

The second experiment was performed on Double DQN LunarLander-v2 OpenAI Gym, figure 4.4 shows the Lunar Lander environment. The agent in this environment has the goal of learning to land successfully on a landing pad located at coordinates $x = 0$ and $y = 0$. The agent gets a reward when moving from the top of the screen to the landing pad with zero speed. The lander loses rewards if it moves away from the landing pad. For each time, the lander firing receives -0.03 points for each frame and the agent gets a negative reward if the lander crashes or a positive reward if the lander is able to land successfully. There are four possible actions: fire left orientation engine, fire right orientation engine, fire main engine and do nothing, and there is no limit on fuel. In this

experiment, four levels of noise were used. The first time the agent runs with no noise, then it runs with 10^{-10}, 2×10^{-6} and 9×10^{-3} of noise, respectively. In this experiment, the agent of Double DQN has been trained for 10000, 30000, 50000 and 80000 episodes, respectively, with the use of 0.9 of moments accountant and epsilon-greedy start from 1 to 0.1 with a decrease rate of 0.999.

4.3.2 Dueling DQN Experiment setup

We performed our experiment on two different Dueling DQN applications to demonstrate the ability to preserve privacy by applying DPSGD [3]. The first experiment was applied to the Health System Simulations model of Dueling DQN, which has the same settings as the Health System Simulations of Double DQN [7]. In this experiment, four levels of noise were used to measure the effect of injected noise, the ability of the agent to learn with each different amount of noise and the ability to preserve privacy. The first time, the agent runs with no noise, and then it runs with 10^{-10}, 10^{-5} and 10^{-2} of noise, respectively. In this experiment, the agent of Dueling DQN has been trained for 500, 1000, 1500 and 2000 runs in order to analyze the behaviour of the agent with each noise level and the different number of runs. In this experiment, we have used the epsilon-greedy to give the agent the opportunity to explore and learn from random action. The epsilon is defined to start from 1 to 0.1 with a decreasing rate of 0.999 and we use 0.9 for moments accountant. Figure 4.6 shows the exploration rate for the Health System Simulations experiment with 250 runs, where the Y axis represents the epsilon and the X axis represents the number of runs. The figure shows a sharp decrease in the epsilon over time.

The second experiment was conducted on the Ambulance Location Problem [8]. This is a Dueling DQN application built based on the OpenAI framework. Figure 4.5 illustrates how the agent interacts in his environment. The main objective of the agent is to help the ambulance choose the best location, which can help to reduce the travel time from the dispatch point to the scene of the accident, thus it can minimize the patient waiting time for the ambulance. The Ambulance Location Problem simulation occurs within areas of the world with fixed dimensions and size of $50km^2$. When incidents occur in this area, the ambulances are sent from the dispatch points to the incident location. The ambulance takes the patient and carries him to the hospital. After this, the agent guides the ambulance to the optimal dispatch point and the ambulance becomes available to respond to any other incident. In this simulation, there is only one hospital and there are 25 dispatch points across the $50km^2$ area. On average, ambulances respond to eight

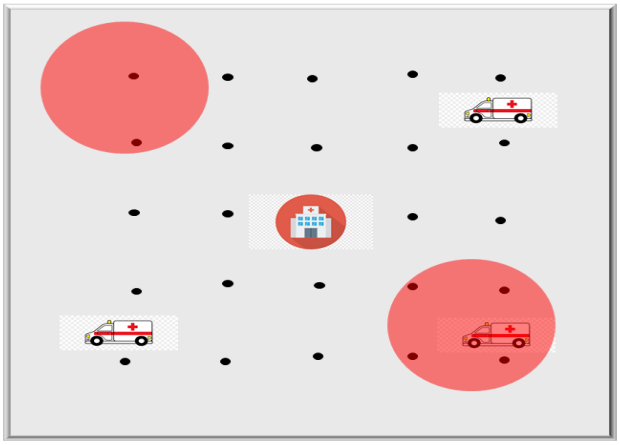


Figure 4.5: The environment of the Ambulance Location Problem and showing the dispatch location of the ambulance and the random ambulance location and the hospital

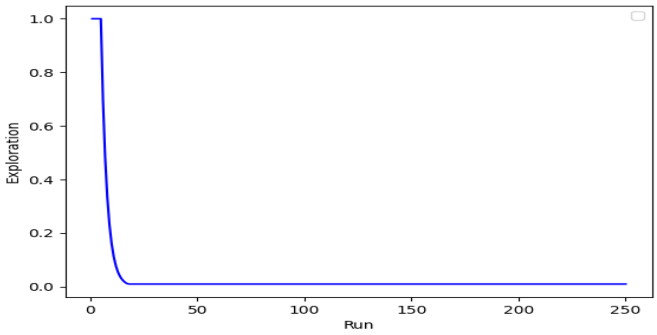


Figure 4.6: Exploration of the Health System Simulations experiment: the agent acts based on an epsilon-greedy policy. If the generated number is smaller than the epsilon, the action will be chosen randomly. Otherwise, the model will choose the optimal action.

incidents per day. Also, four levels of noise were used in this experiment to measure the effect of injected noise and the ability of the agent to learn with each different amount of noise. We run the agent the first time with no noise, and then it runs with 5×10^{-3}, 10^{-2} and 5.0 of noise, respectively. The agent in this experiment has trained for 300, 500, 700 and 1000 runs. We used epsilon-greedy to give the agent the opportunity to explore and learn from random action. The epsilon is defined to start from 1 to 0.5 with a decreasing rate of 0.999 and we use 0.5 for moments accountant.

4.4 Results and Analysis

This section presents the experiment results of the two Double DQN settings and the experiment result of two settings of Dueling DQN after applying Differentially Private SGD algorithms.

4.4.1 Double DQN

This section illustrates the behaviour of the two synthetic Double DQN applications after applying DPSGD algorithms. The results of the conducted experiments on the Double DQN model are as follows.

4.4.1.1 Performance metric 1 (accuracy)

Figure 4.7 and Figure 4.8 show the results of the Health System Simulations experiment and LunarLander-v2 experiment, respectively, where the x axis shows the number of run times and the y axis shows the average rewards. For the Health System Simulations experiment, different noise levels are used with a different number of runs. Figure 4.7 shows the results of the Health System Simulations experiment where in the beginning, there is very high fluctuation when the agent starts, especially in the first 100 runs with all levels of noise. This is due to the use of epsilon greedy as the agent starts taking action randomly for a certain time then the agent uses their prediction later to take action. This method gives the agent the opportunity to explore and learn from random actions. The red line represents the performance and accuracy of the agent with no noise, as it appears that his performance improves gradually and then stabilizes at the highest level. The black and blue lines show the performance and the accuracy of the agent with a noise level of 10^{-10} and 10^{-5}, respectively. There is a slight decrease in the agent's accuracy with these noise levels and the agent is able to increase his performance over time. Even though the agent was able to improve its performance, the agent in this experiment wasn't able to reach the same level of accuracy, such as no noise, even after 3000 runs. The green line shows the performance and accuracy of the agent with noise level 10^{-3} this is the highest noise level in the experiment, and this demonstrates the effect of increasing noise as it shows that the agent performs with very low accuracy and is far from no noise.

Figure 4.8 shows the experiment results of LunarLander-v2. In all the experiments, the agent starts with a very high fluctuation because of the use of the epsilon-greedy

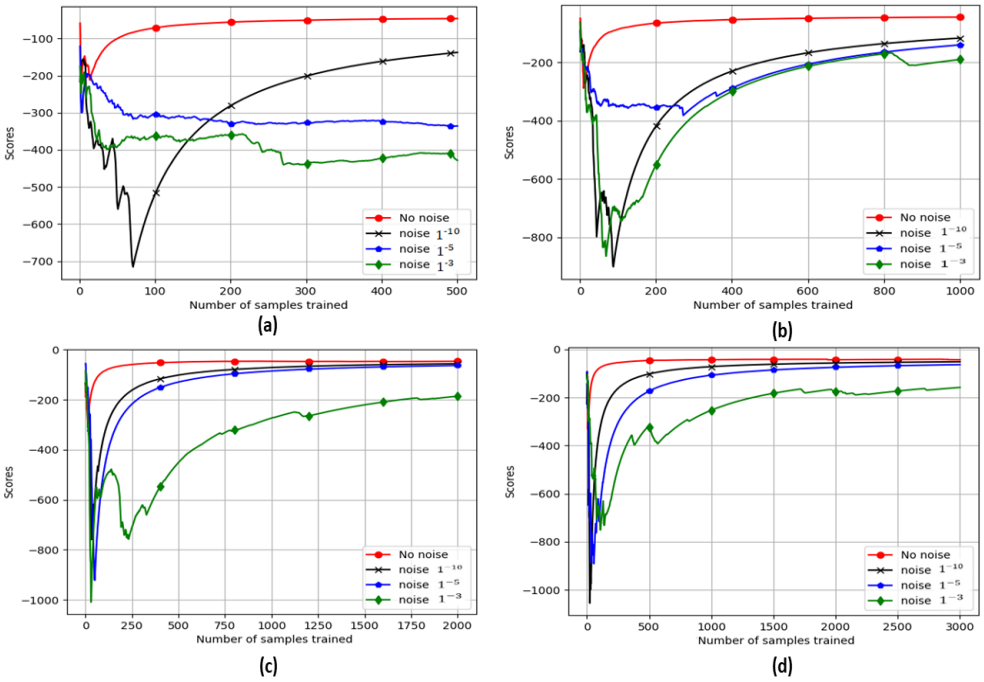


Figure 4.7: The results of the Double DQN Health System Simulations experiment after training the agent for 500(a), 1000(b), 2000(c), and 3000(d) runs

method as the agent starts taking action randomly for a certain time, and then the agent uses his prediction later. The red line represents the performance of the agent with no noise as it shows that the performance improves steadily and indicating how the agent learns over time. The black and blue lines demonstrate the performance and the accuracy of the agent with a noise level of 10^{-10} and 2×10^{-6}, where the agent starts with lower accuracy and increases over time. The agent in these two lines is not able to reach the same level of accuracy, such as no noise, even after 80000 runs. The green line represents the agent's accuracy with a noise level of 9×10^{-3}. The agent performs very poorly with this level of noise and its accuracy drops dramatically. The agent is not able to increase its accuracy even after 80000 runs.

4.4.1.2 Performance metric 2 (convergent rate/speed)

The converging is when the agent converts into an optimal policy and the speed of convergence is affected by different parameters such as the discount factor [18]. In this section we compare the convergent rate for the agent after applying DPSGD with different amounts of noise. Figure 4.7 shows the performance of the Health System Simulations agent, and the red line representing the agent with no noise, where it is able

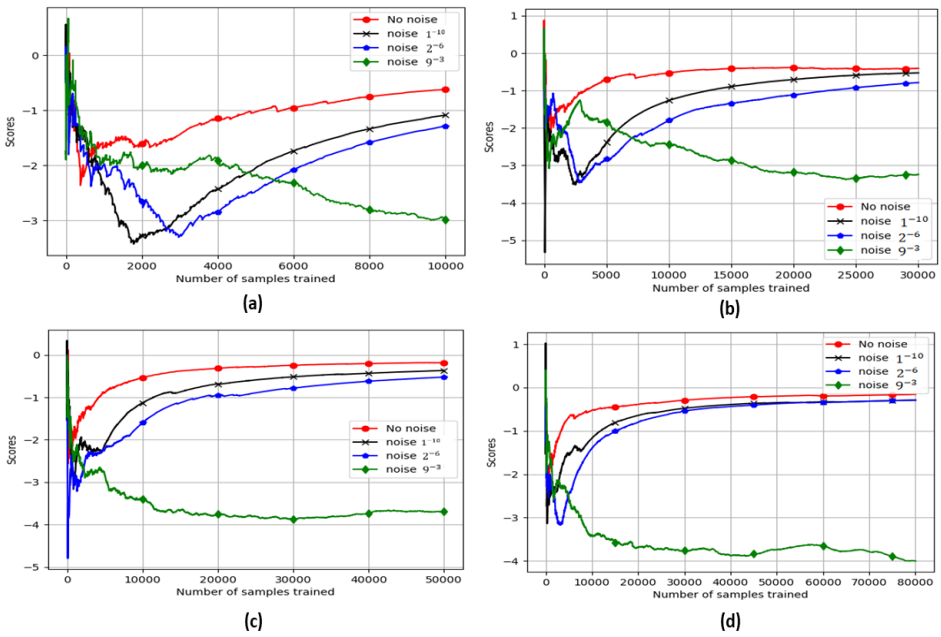


Figure 4.8: The result of Double DQN LunarLander experiment after the agent is trained for 10000(a), 30000(b), 50000(c) and 80000(d) runs

to converge and find the optimal policy very quickly after 100 samples of training. The agents are represented by the black and blue lines with a low amount of noise 10^{-10} and 10^{-5} and they need more samples to train to reach to close level of the red line and almost they are very close to converging after 1000 samples of training. On the other hand, the agent with a higher level of noise 10^{-3} is far away from being converted to optimal policy and still needs more samples to train. Figure 4.8 demonstrate the LunarLander-v2 performance, with the red line showing the agent with no noise is able to converge to the optimal policy after 11000 sample training. The agents represented by the black and blue lines with a low level of noise 10^{-10} and 2×10^{-6} are very close to achieving convergence to the optimal policy after 30000 runs. On the other hand, the agent in the green line with a higher level of noise 9×10^{-3} is far away from being converged to optimal policy because of injecting high noise into the gradient by Differentially Private DPSGD.

4.4.2 Dueling DQN

This section illustrates the behaviour of the two synthetic Dueling DQN after applying Differentially Private SGD algorithms. The result of the conducted experiments on the Dueling DQN model are detailed as follows.

4.4.2.1 Performance metric 1 (accuracy)

Figure 4.9 and Figure 4.10 show the result of the Health System Simulations and Ambulance Location Problem experiments, where the x axis shows the number of run times and the y axis shows the average rewards that the agent receives over time. In these experiments, we used different random levels of noise with different numbers of runs. Figure 4.9 demonstrates the experiment results for the Health System Simulations experiment using different noise levels with different numbers of runs in the experiment. In the beginning, there is a very high fluctuation when the agent starts, especially in the first 60 runs with all levels of noise. This is due to the use of the epsilon greedy method, as the agent starts taking action randomly for a certain time, and then the agent uses their prediction later to take action. This method gives the agent the opportunity to explore and learn from random actions. The red line represents the performance and accuracy of the agent with no noise, showing its performance improves gradually in the beginning and then stabilizes at the highest performance level. The black and blue lines represent the performance and the accuracy of the agent with lower noise levels of 10^{-10} and 10^{-5}, respectively. With this amount level of noise, we see there is a very slight decrease in the agent's accuracy with these noise levels and the agent is able to increase its performance over time and reach to level very close to no noise, specifically after 600 runs. The green line shows the performance and accuracy of the agent with the highest level of noise in our experiment with noise level of 10^{-2}. This is the highest noise level in the experiment, and it demonstrates the effect of increasing noise as it shows that the agent performs with very low accuracy. Even though the agent was able to slightly improve his performance, the agent in its experiment was far away from being close to the level of no noise.

Figure 4.10 shows the results of the Ambulance Location Problem experiments. In all the experiments, the agent started with very low accuracy and the accuracy increased over time. The red line represents the performance of the agent with no noise showing that its performance increases steadily, which indicates how the agent learns over time. The black and blue lines represent the performance and the accuracy of the agent with a noise level of 5×10^{-3} and 10^{-2}, where the agent starts with lower accuracy and increase over the time. The agent represented by two lines is not able to reach the same level of accuracy, such as no noise, even after 1000 runs. The green line represents the agent's accuracy with a noise level of 5.0 showing that the agent performs very poorly with this level of noise and shows that the accuracy drops dramatically, and sometimes the

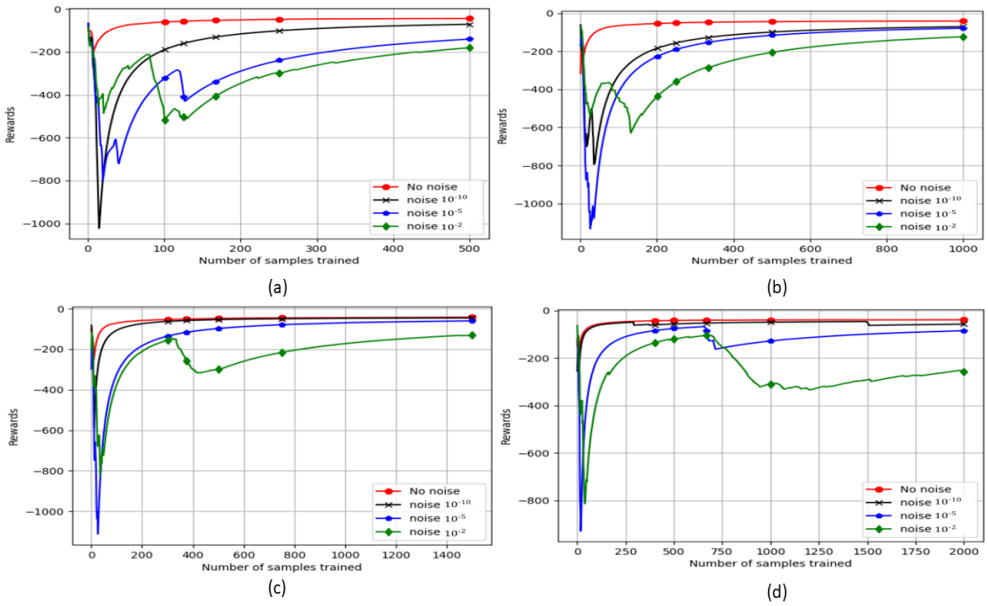


Figure 4.9: The result of Dueling DQN Health System Simulations experiment after the agent has been trained for 500(a), 1000(b), 2000(c) and 3000(d) runs

accuracy goes below the starting point.

4.4.2.2 Performance metric 2 (convergent rate/speed)

The converging is when the agent converts into an optimal policy and is able to maximise its total cumulative rewards over time where the speed of convergence is affected by different parameters such as discount factor γ [18]. We can obtain convergence from the stability of the final results obtained by the algorithm. [50]. In this part, we compare the convergent rate for the agent after applying DPSGD with different amounts of noise. Figure 4.9 shows the performance of the Health System Simulations agent, and the red line represents the agent with no noise as it is able to converge and find the optimal policy very quickly after 100 samples of training. The agents represented by the black and blue lines with a low amount of noise 10^{-10} and 10^{-5} need more samples to train to reach to close level of the red line and almost they are very close to converging after 1000 samples of training. On the other hand, the agent with a higher level of noise 10^{-2} is far away from being converted to optimal policy and still needs more samples to train. Figure 4.10 illustrates the Ambulance Location Problem performance, with the red line presenting the agent with no noise as it is able to converge to the optimal policy after 300 sample training. The agents represented by the black line with a low level of noise 5×10^{-3} are close to achieving convergence to the optimal policy after 1000 runs as it is

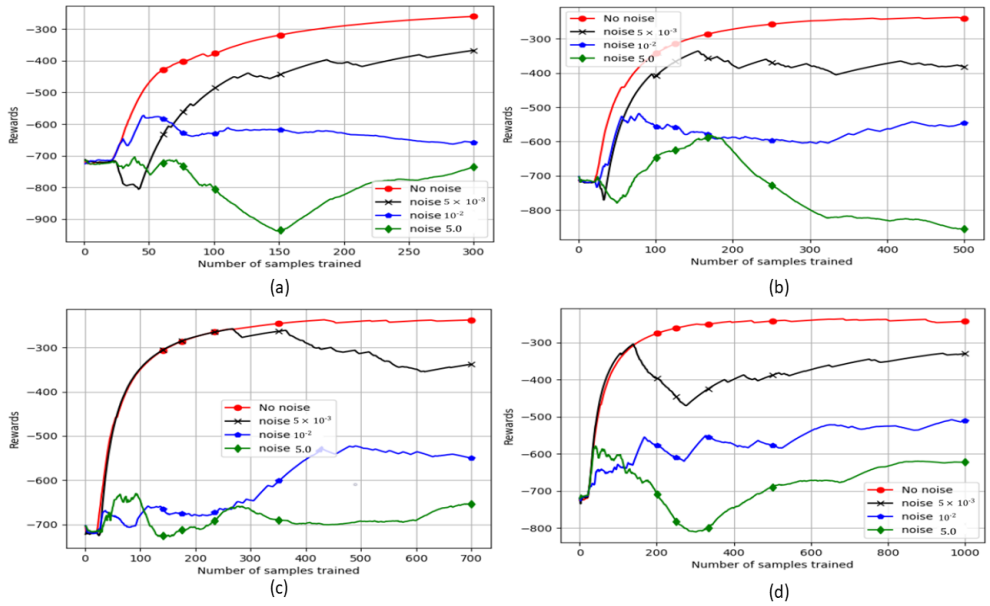


Figure 4.10: The result of Dueling DQN Ambulance location problem experiment after the agent has been trained for 300(a) ,500(b) ,700(c) and 1000(d) runs

able to increase the accuracy over time. On the other hand, the agent represented by the blue and green lines with a higher level of noise 10^{-2} and 5.0 are far away from being convergence to the optimal policy.

4.5 Conclusion

We considered the Double DQN and Dueling DQN algorithms and aim to preserve the privacy of these models by adopting the DPSGD algorithm. In this research, we conducted two experiments for Double DQN and two experiments for Dueling DQN. The two experiments for Double DQN were the Health System Simulations model, which is built to simulate real health systems, and the second experiment was the LunarLander-v2 OpenAI Gym. We also conducted two experiments for Dueling DQN. The first experiment was the Dueling DQN Health System Simulations model, and the second experiment was performed on the Ambulance Location Problem, which aims to minimize the time from the dispatch point of the ambulance to the accident site. For all experiments, different noise levels were injected to demonstrate the effectiveness of the injected noise on agent accuracy and learning rate. The effect of noise on the agent's performance is very clear. A noise increase leads to decreased accuracy. The agent with low noise levels needs more samples to train to be almost able to converge to the optimal

policy.

CHAPTER 5

PROTECTING USER LOCATION INFORMATION IN ONLINE FOOD DELIVERY SERVICES

Online food delivery services are growing in popularity nowadays. A significant privacy risk has been raised recently and private information such as customer location information could be disclosed. In this chapter, we consider this issue and we propose the Protects User Location Method (PULM) to protect customer location. The PULM injects differential privacy Laplace noise based on two parameters: city area size and customer frequency of online food delivery orders. We used two datasets, Shenzhen, China and Iowa, USA, to demonstrate the result of our experiments.

5.1 Introduction

With the emergence of the internet and smartphones, online food delivery applications have become prevalent and popular around the world. The online food delivery market is growing rapidly and is expected to reach about US$1.02tn in revenue in 2023. There is an expectation of a growth rate of 12.78% in revenue leading to an increase in the market volume of US$1.65tn by 2027. More specifically, the segment of grocery delivery is predicted to have a growth of revenue of 22.2% in 2024 and generate about US$0.63tn in 2023. wherein, by global comparison, the highest revenue generated will be in China. The expected number of users in the meal delivery segment is 2.5bn users by 2027 [93].

The changing nature of urban consumer behaviour could be explained by the popularity and heavy use of the emergence of online food delivery services. There are many reasons behind the increased use of food delivery services, such as having a convenient and quick meal after or during a busy working day, the need to take a particular food dish during a family gathering or relaxing and enjoying a delicious meal by using these services. Food delivery services aim to make the service more convenient for the consumer by removing the need to plan meals or go to a restaurant to buy food to bring back home or to the office. Thus, the behaviour of the consumer has changed greatly with the increase in food delivery services, most noticeably consumers in urban areas [22].

While online food delivery services are valuable and provide significant benefits, there are some privacy concerns have been raised relating to the probability of leaking sensitive information about the user, such as the customer's location. Thousands of food delivery orders are received daily by online food delivery applications, with vast amounts of information collected from the user. This data could be hosted by a third party and could be further processed for training and analysis purposes which could pose a serious threat to the privacy of customer information, such as the customer's location.

We propose a method called the Protect User Location Method (PULM) to tackle the privacy issues in online food delivery services. The objective of this method is to maintain the privacy of the customer's location. This method relies on the differential privacy Laplace mechanism by injecting noise into the customer's location and the courier's trajectory. The privacy parameter ϵ that affects the amount of injected noise depends on two factors: the city area size and the customer frequency of online food delivery orders. In small cities, the adversary has a higher opportunity to identify the customer's location due to the small area, fewer routes and fewer orders, which makes it easier for an adversary to infer some private information about the user. However, this could be more difficult in large cities. Also, when there are several food delivery requests from the same customer, the adversary may be able to link the records of the same customer and obtain their private information [5].

The main contributions of this work are as follows:

- proposes a privacy method called the Protect User Location Method (PULM). This method uses the city area size and the customer frequency of online food delivery orders to determine the privacy parameter ϵ.

- Employ differential privacy by injecting Laplace noise to protect the customer's location information along with the courier's trajectory.
- Use two datasets with different city area sizes and different geographic areas to demonstrate more results and comparisons.

The remainder of this chapter is organized as follows. Section 5.2 presents the problem statement. Section 5.3 describes the methodology. Sections 5.4, 5.5, and 5.6 detail the experiment design, the results and the conclusion.

5.2 Problem statement

5.2.1 Model overview and our method

Today, with the emergence of advanced technologies and the ubiquitous use of the Internet and smartphones, ordering food online has become more accessible and convenient with the prevalence of online food delivery services. This technology facilitates the process of obtaining food, as it acts as an intermediary between the seller and the customer by picking up the food from the restaurant or store and deliver it to the customer. While online food delivery services provide valuable services, some privacy concerns about the disclosure of private information have been raised. The customer's private information could be disclosed, and an adversary could infer private information such as the customer's location or behaviour pattern. As the customer's location is sensitive, we are required to protect the actual location and courier trajectory to maintain the privacy of the customer's location.

To tackle this issue, a defence method has been proposed to protect the customer's location. The proposed method called the Protect User Location Method (PULM), aims to maintain the privacy of the customer's location in online food delivery service. This method uses differential privacy and the Laplace mechanism by injecting Laplace noise into the customer's location and the courier's trajectory. The privacy parameter ϵ that affects the amount of injected noise depends on two parameters: the city area size and the frequency of customer's online food delivery orders. In small cities, the adversary has a higher opportunity to identify the customer's location due to the small area and fewer number of routes, which makes adversary easier to find the location of the user, whereas it could be more difficult in large cities due to the large geographical area and

increase number of roads. Also, when there are a number of food delivery requests from the same customer, the adversary may be able to link the records of the same customer and obtain the customer's private information. Therefore, we inject more noise into the records requested from small cities and with customers with a high number of records of online food delivery orders.

5.3 Methodology

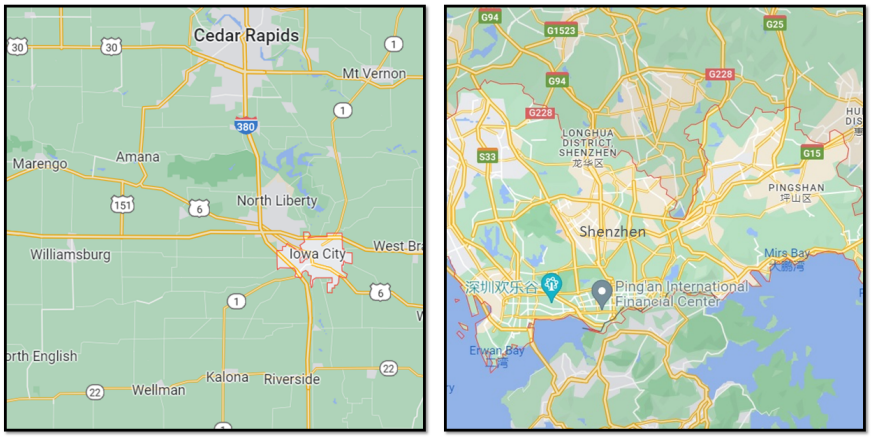


Figure 5.1: Comparison of the city area size between the Iowa(left) and Shenzhen(right)

5.3.1 Protect User Location Method (PULM)

Safeguarding the privacy of the customer's location in online food delivery applications is crucial. In this section, we focus on how to protect the user's location in an online food delivery service. Thus, we design the PULM method to protect the user's location by injecting noise into the user's location and the courier trajectory. We use differential privacy along with the Laplace mechanism to guarantee that the user's location is not disclosed. We determine the privacy parameter in this method based on two parameters. The first parameter is the city area size, and the second parameter is the customer frequency of online food delivery orders.

Algorithm 4 Protect user location method PULM

1: **Input**: Merchant's location and the customer's location
2: **Input**: Obtain courier trajectory datasets $\mathscr{D}$
3: **Input** Customer frequency of online food delivery orders.
4: **Input** City size.
5: Based on $Eq.$ (2.4), obtain the sensitivity Δf of the trajectory metric space.
6: Calculate the privacy parameter ϵ based on city size and customer order frequency.
7: Select random point from the input trajectory dataset $\mathscr{D}$ including the user location.
8: Add Laplace noise based on Δf and ϵ to a selected points.
9: **Output**: Perturbed user location and courier trajectory.

5.3.1.1 Overview of the PULM algorithm

Preserving the privacy of user location is essential. Differential privacy shows robustness and effectiveness in providing adequate privacy to datasets. Algorithm 4 present the execution of the PULM method and how it works. In the beginning, the merchant's location and the customer's location are obtained, and then the trajectory dataset of the courier is retrieved from Google Maps API and then entered in the form of location points containing longitude and latitude from the start point to the destination. In lines 3 and 4, the customer frequency of online food delivery orders and city area size is calculated and entered. The sensitivity Δf calculated in line 5 based on $Eq.$ (2.4). In line 6, the privacy parameter ϵ is calculated. To calculate the privacy parameter, we sum the results from the city area size, which has a weight of 2.5, with the results of the customer frequency of online food delivery orders, which has a weight of 2.5 where the lower privacy parameter is more strong privacy. For example, the Iowa City area < 500 km^2 will give 0.5, and let us suppose we have a customer with 5 times of orders and max(x) = 10, min = 1 in the dataset. This will be calculated like (5-10)\(1-10) × 2.5 = 1.388, in this case, the privacy parameter will be 0.5 +1.388 = 1.88. In line 7, random points are selected from the trajectory dataset, including the location of the customer. In line 8, Laplace noise is added based on the sensitivity Δf and privacy parameter ϵ. The output is a perturbed trajectory and customer location.

5.3.1.2 City area size

The city area size is an important factor, as it could affect the ability to infer user location and courier trajectory. Figure 5.1 shows the difference between the two examples of city area size, which we consider in this research. In a large city area size, there is a larger population, and there are more roads and cars. Therefore, inferring trajectory and

customer location could be more difficult compared with a smaller city area size. For example, figure 5.1 shows the size of Iowa and Shenzhen, where the area of Iowa is about 64.75 km^2 compared with about 1,748 km^2 in Shenzhen and the population in Iowa is about 75,233 compared with about 17.56 million in Shenzhen [91] [72] [29].

Globally, city area sizes are different and could start from less than 50 km^2 to more than 8000 km^2, such as Tokyo with 2,191 km^2 city area size [1] and Iowa with a city area size of 64.75 km^2 [92]. To allocate the proper amount of noise in our model, we need to classify the city based on the area size. Most of the city classifications are based on city population. For example, UK Parliament [13] classified the cities based on population into core cities such as London, Sheffield and Glasgow, other cities, large towns, medium towns, small towns and villages. Also, Liu and Chen [55] classify cities as a super city for a city that has a population of more than 10M population, a megacity for a city between 5 and 10M, a large city for a city with a population between 1 to 5M, a medium city for a city with a population between 0.5 to 1M and small city with a population less than 0.5M. In our model, we attempted to classify cities based on area size, and we approximately mimicked the population factor. Equation (5.1) shows the distribution of noise parameters based on the city area size that we used in our model, where the x is the city area size.

$$f(x) = \begin{cases} 0.5, & \text{if } x <= 500. \\ 1, & \text{if } 500 < x <= 1000. \\ 1.5, & \text{if } 1000 < x <= 1500. \\ 2, & \text{if } 1500 < x <= 2000. \\ 2.5, & \text{if } x > 2000. \end{cases} \tag{5.1}$$

5.3.1.3 Customer frequency of online food delivery orders

In our method, we consider the factor of how frequently the customer places an online food delivery order. When there are many orders from the same customer, there is a high chance of an adversary inferring the information of the customer. The attacker may be able to infer the customer's location and behavioural patterns or other information, especially if many repeat orders are from the same location. An example of the ability of the attacker to infer some private information from a particular user is that the attacker could infer if a particular individual may have health issues if the user frequently visits a

hospital location that is not his home or workplace[99]. Thus, we consider this factor and aim to protect user location by injecting more noise using the Laplace mechanism into the user with higher frequency orders by the same customer. We calculate the privacy parameter based on two parameters: the city area size and the customer frequency of online food delivery orders. The customer frequency of online food delivery orders can be determined using this equation.

$$\mathscr{Y} = \frac{(x - Max(x))}{(Min(x) - Max(x))} \times 2.5 \tag{5.2}$$

x represents each customer frequency of online food delivery orders, $Max(x)$ is the dataset the maximum frequent order number and $Min(x)$ is the dataset minimum frequent order number.

5.4 Experiment Design

In the following section, we detail the experiment setup for the Protect User Location Method (PULM) and provide an overview of the datasets we use in these experiments.

5.4.1 Trajectory Protection Method

Preserving the privacy of the user's location is vital. In this experiment, we use the Protect User Location Method (PULM), which employs a differential privacy Laplace mechanism to protect the user's location. We inject Laplace noise into the customer location along with the courier trajectory. The Method considers two important factors to determine the privacy parameter ϵ. The first parameter is city area size, and the second parameter is the customer frequency of online food delivery orders. These factors determine the amount of injected noise based on the differential privacy algorithm. To evaluate our approach, we use Shenzhen, China and Iowa, USA datasets. We use Google Maps to retrieve the courier's trajectory by sending the start and end points and then retrieve the trajectory of the courier. We then use the PULM method to obfuscate the trajectory by injecting noise into selected points of the trajectory along with obfuscating customer location.

5.4.1.1 Trajectory Data Utility

Adding Laplace noise to the trajectory inevitably affects data utility. In order to evaluate the data utility in the trajectory after adding noise, we use Hausdorff Distance (HD). The Hausdorff distance method is a widely used metric to measure the similarity of two datasets of points, and we can measure the utility of dataset D with respect to dataset $\tilde{D}$[51]:

$$utility(D) = max\{h(D,\tilde{D}), h(\tilde{D},D)\} \tag{5.3}$$

This experiment demonstrates the result of Hausdorff Distance (HD) for Shenzhen, China, and Iowa, USA datasets, along with the average Hausdorff Distance for both datasets.

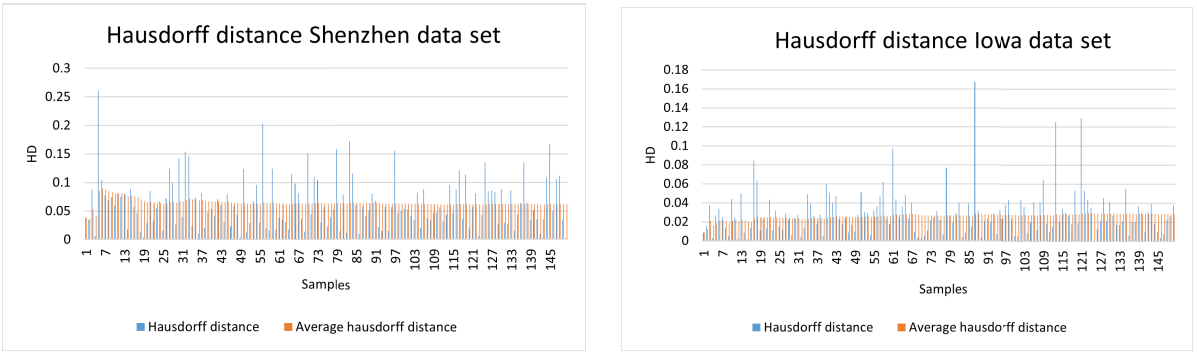
5.4.1.2 Analyze Privacy Parameter Intensity

In this part, the distribution of privacy parameters generated by PULM is analyzed. The privacy parameter is changed frequently based on the city area size and the customer frequency of online food orders for each customer. The privacy parameter affects the amount of noise generated by the privacy mechanism. We analyze the generated privacy parameter for the two datasets, Shenzhen, China and Iowa, USA. Moreover, we compared the generated privacy parameter ϵ with the frequency of customer orders of online food delivery in both datasets and we consider each unique destination coordinate as a separate customer.

5.4.2 Dataset Description

Two different datasets that have different city area sizes were used in our experiments for our approach. The first dataset is for Shenzhen city, China, provided by Alibaba Local Service Lab [32] contains on-demand food delivery order data. The datasets have 1048575 orders and 93 restaurants with the location of restaurants and delivery locations and time of pickup the meal from restaurants and delivery time. The second dataset used in our experiment is provided by Ulmer et al. [100]. This dataset contains data on meal delivery services in Iowa City, USA. This dataset has 111 restaurants and 1200391 delivery records. In this dataset, the actual location is used with random generation of order and equal request probability for each point of time and location. For both cities,

we consider a particular area to implement our experiment. The data may have been modified from the source for certain purposes, such as obfuscating the exact location or any other reasons.



((a)) ((b))

Figure 5.2: The result of using Hausdorff Distance to evaluate data utility before and after adding Laplace noise to the trajectory. We used Hausdorff distance to calculate the similarity of the two datasets. We counted the utility of dataset D towards $\tilde{D}$ dataset. The figure shows the result of Shenzhen, China, and Iowa, USA, where the X-axis demonstrates the samples used and the Y-axis demonstrates the corresponding results of the Hausdorff Distance in the blue line and the average Hausdorff Distance in the orange line

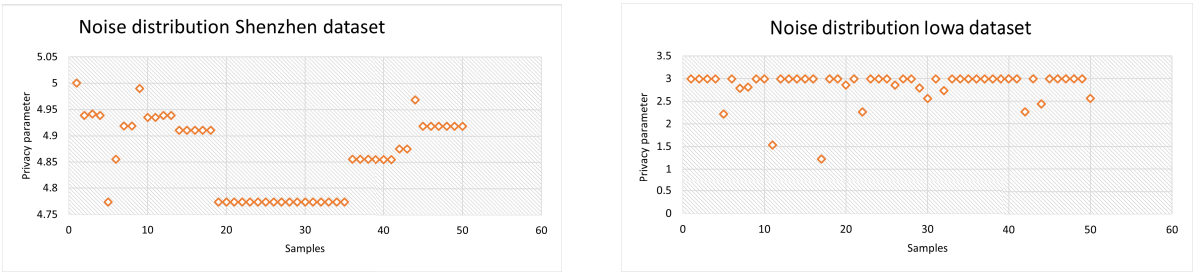
5.5 Experiment Results

Preserving the privacy of a customer's location is vital. This experiment shows the result of using the PULM method, where we aim to protect customer location along with courier trajectory. We use Google Maps to fetch the trajectory by sending the start and the end points to Google Maps API and then retrieving the trajectory. Using our approach, proper noise is injected into the customer location and to a selected point of the courier trajectory in order to safeguard the customer location.

5.5.1 Trajectory Data Utility

After Laplace noise is injected into the trajectory, the utility of the data is definitely impacted. To evaluate data utility in the trajectory after adding noise and to see how much the obfuscated trajectory has been impacted, we used the Hausdorff Distance (HD), which is commonly used to determine how much the similarity of two datasets points.

Figure 5.2 shows the result of Hausdorff Distance along with the average Hausdorff Distance in Shenzhen and Iowa. The Figure shows a slight similarity of HD between the two datasets with some fluctuation which normally occurs due to obfuscated function. The smaller value of the HD is more similar between the two datasets and zero means it is identical with no difference.



((a)) ((b))

Figure 5.3: The result of the privacy parameter ϵ distribution generated by the PULM algorithm for the two datasets, Shenzhen, China and Iowa, USA. The Y-axis shows the privacy parameter ϵ and the X-axis shows the samples.

5.5.2 Analyze Privacy Parameter Intensity

Each time, the PULM generates the proper noise based on the input. Thus, the privacy parameter is not fixed and it's changed each time based on the city area size and the customer frequency of online food orders. Figure 5.3 shows samples of the distribution of privacy parameter ϵ generated by the PULM algorithm over the two datasets, Shenzhen, China and Iowa, USA. The generated parameter can range between 0.5 to 5.0. Figure 5.3 shows that the distribution of privacy parameter ϵ in Shenzhen ranges between 4.75 and 5.0, and it ranges between 1.3 and 3.0 for the Iowa dataset in this selected sample of records.

Figure 5.4 shows the privacy parameter ϵ generated by the PULM in the Shenzhen dataset compared with the frequency of customer orders of online food delivery as it shows the range starts from 4.0 to 5.0 in this selected sample of records. Figure 5.5 shows the privacy parameter ϵ generated by the PULM in the Iowa dataset compared with the frequency of customer orders where the privacy parameter ϵ starts from 1.2 to 3.0 in this selected sample of records. In both Figures, the X-axis represents the privacy parameter ϵ and the Y-axis represents the customer frequency order of online food delivery services,

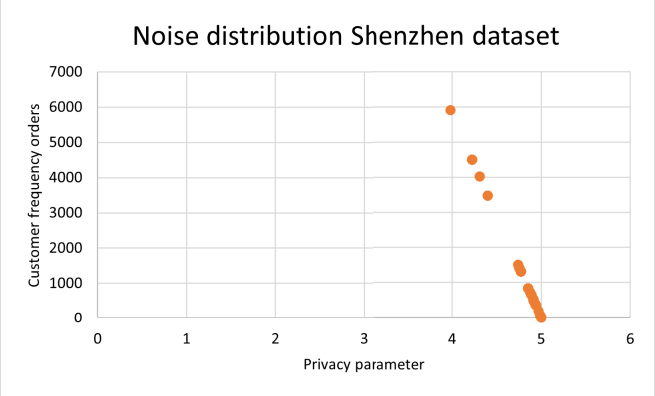


Figure 5.4: The generated privacy parameter compared with the customer frequency of online food delivery orders in the Shenzhen dataset

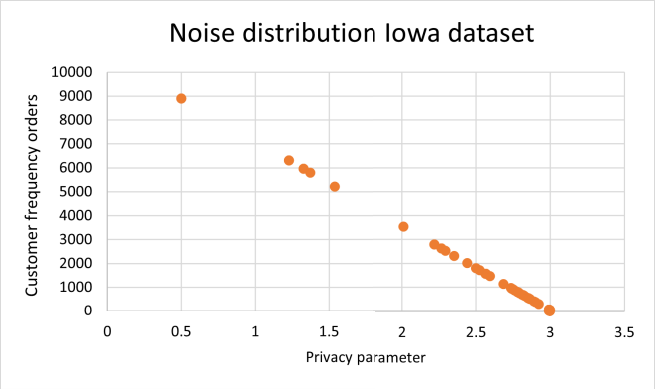


Figure 5.5: The generated privacy parameter compared with the customer frequency of online food delivery orders in the Iowa dataset

and it shows that when the number of customer frequency orders is high, the privacy parameter gets lower which represents stronger privacy.

5.6 Conclusion

This research considered how to protect the customer's location information in online food delivery services. We proposed the Protect User Location Method (PULM) to protect user location information in online food delivery services. PULM injects differential privacy Laplace noise and uses two factors to control the amount of used noise. The first factor is the city area size, and the second factor is the customer frequency of use of online food delivery services. We used two datasets, Shenzhen, China and Iowa, USA, to demonstrated our experiment results.

CHAPTER 6

PROTECTING THE CUSTOMER'S LOCATION IN FOOD DELIVERY SERVICE WITH DIFFERENTIAL PRIVACY AND MULTI-AGENT REINFORCEMENT LEARNING

Many food delivery companies operate globally in different regions today, with numerous employees having various access authorities to customer data. A third party could store the customer's data and could also be used in further analysis. The stored data must be stored properly to prevent a malicious from identifying the real data. This chapter considers this issue and proposes the Protect Trajectory and Location in Food Delivery (PTLFD), which aims to maintain the privacy of stored customer data by leveraging multi-agent reinforcement learning and differential privacy. We implemented our experiment on the meal delivery dataset in Iowa, USA.

6.1 Introduction

The food delivery business has grown rapidly, specifically during the COVID-19 pandemic [30]. The platforms of online food delivery connect multiple restaurants with customers and couriers on one platform. Many restaurants present their menus on these platforms, and customers select the restaurants and place their orders online. The platform of food delivery dispatches orders to the couriers, and based on specific techniques, they suggest a path from the restaurant to the customer. The food is picked up by the couriers from the restaurants and the couriers select a route to deliver the food to the customer. According

to Liu et al. [56], in 2018, the market volume of online food ordering and delivery in China reached 441.5 billion yuan and it was predicted that the market will reach 934 billion yuan in 2021. However, the online food delivery market today is getting more attention and popularity among people everywhere due to the unique service it provides to a large number of people around the world.

Many food delivery companies today operate globally in different countries, resulting in several headquarters with numerous employees to operate their business in each region and country. Different access authorities are granted access to various data, which gives the possibility of illegal access to customer data. Moreover, this data could be stored and analysed by third parties, or even the IT department could be operated by a third party as well. A malicious entity could obtain this access authority, and the data could be used illegally, which gives a chance to leak sensitive data about a group of people or particular individuals. Some of the data could be related to important individuals or linked with sensitive places such as some government offices, military or security. Disclosing this information could reveal private and sensitive information such as user location, behavioural pattern or job location, which could cause serious harm to the individual or group of people and could be used in a harmful manner.

In this work, we propose the Protect Trajectory and Location in Food Delivery Method (PTLFD) to protect the privacy of such data. This method leverages the differential privacy Laplace mechanism and multi-agent reinforcement learning to obfuscate the customer's location and courier's trajectory. Initially, the courier receives a food delivery order and then delivers the order to the customer. After that, by using different privacy parameters ϵ, the multi-agent reinforcement learning agent constructs N number of obfuscated trajectories by injecting Laplace noise to a selected point of the original trajectory and destination point. The multi-agent reinforcement learning is then used to select one of the constructed trajectories. The selected trajectory is then evaluated based on three factors: the similarity between the selected trajectory and the original trajectory, the sensitivity of destination location and the frequency of the number of orders by the customer [6].

This work aims to protect the customer's location information and proposes a method to safeguard the customer's location. The main contributions of this research are as follows:

1. This work is the first one that leverages multi-agent reinforcement learning in trajectory privacy protection and uses the QMIX method to train decentralised policies in a centralised end-to-end fashion.
2. To have a strong obfuscated trajectory, we used the differential privacy Laplace mechanism to construct N number of obfuscated trajectories.
3. To gain highly private results, we used the similarity of the selected trajectory with the original trajectory, the sensitivity of the destination location and the frequency of the customer orders as factors used in the reward function.

The remainder of this chapter is organized as follows. Section 6.2 details the main difference between PULM and PTLFD. Section 6.3 presents the problem statement. The methodology is presented in section 6.4. Sections 6.5, 6.6 and 6.7 detail the experiment design, the results and the conclusion.

6.2 The Main Difference Between PULM and PTLFD

The Protect User Location Method (PULM) was proposed to tackle privacy issues in online food delivery services and protect the location information of the customer from being disclosed. This method uses the differential privacy Laplace mechanism by injecting noise into the customer's location and the courier's trajectory. In this method, the privacy parameter ϵ that affects the amount of injected noise depends on two parameters: the city area size and customer frequency of online food delivery orders.

The PTLFD was proposed to deal with privacy issues in online food delivery services and to protect the customer's location information from being disclosed. This method uses the differential privacy Laplace mechanism and multi-agent reinforcement learning to obfuscate the customer's location and trajectory. After the courier delivers the order to the customer, the agent constructs N number of obfuscated trajectories with different amounts of privacy parameters by injecting Laplace noise to a selected point of the original trajectory and destination point. Multi-agent reinforcement learning is then used to select one of the obfuscated constructed trajectories. The selected trajectory is then evaluated based on three factors: the similarity between the selected trajectory and the original trajectory, the sensitivity of destination location and the frequency of the number of orders by the customer.

In the PULM, differential privacy is used to protect the customer location information and uses the city area size and customer frequency of online food delivery orders to determine the privacy parameter ϵ that affects the amount of injected noise. Whereas, in the PTLFD, the agent after the order is delivered by the courier to the customer, the agent constructs N number of obfuscated trajectories with different amounts of privacy parameters by injecting Laplace noise to a selected point of the original trajectory and destination point. In this method, multi-agent reinforcement learning is used to select one of the obfuscated constructed trajectories and multi-agent reinforcement learning then evaluates the selected trajectory based on three factors: the similarity between the selected trajectory and the original trajectory, the sensitivity of the destination location and the frequency of the number of orders by the customer.

6.3 Problem Statement

This section explains in detail the problem statement, provides an overview of the model, its components, and how it works. Also, it explains the formulation of the markov decision process.

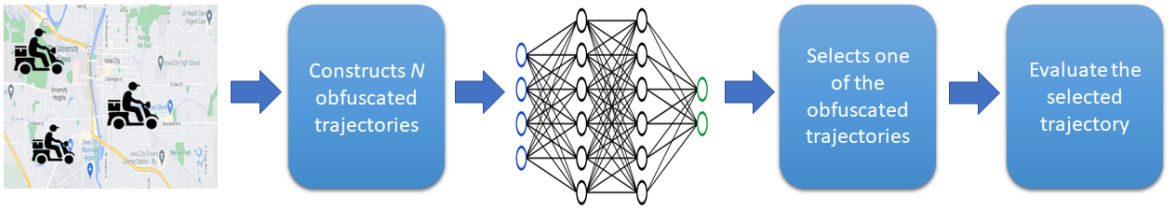


Figure 6.1: Model overview

6.3.1 Model Overview

Today, multiple food delivery companies work globally in different countries, with different headquarters and numerous employees to operate their business in each region, and different access authorities are granted to access various data. This creates the opportunity for illegal access to customer data in different regions. This data could be stored and analysed by third parties, or the IT department could be operated by a third party as well. Disclosing this information could reveal private or sensitive information

such as the user's location, behaviour patterns or job location. Furthermore, this data could be related to important individuals or linked with sensitive places such as some government offices, military or security sites. This work considers this issue and proposes a solution to maintain customer data privacy and not disclose their private information. We leverage differential privacy and multi-agent reinforcement learning in this work to maintain the customer's privacy.

Figure 6.1 shows the model overviews. Each courier is considered in this model as an independent agent. The courier receives the food order and then delivers the order to the customer. In order to obfuscate the customer's location, the PTLFD aim to obfuscate the customer location and the courier trajectory, and this is done by creating N number of obfuscated trajectories by adding Laplace noise into a selected point of the trajectory using a different amount of privacy budget in each trajectory. The multi agent reinforcement learning then uses their prediction and then picks up one of the created trajectories and uses it instead of the original trajectory. The multi agent reinforcement learning then evaluates the selected trajectory in the reward function based on three criteria: the similarity between the original trajectory and the selected trajectory, the sensitivity of destination location, and the frequency of the number of orders placed by the customer.

6.3.2 DRL Formulation

Our method is designed as the Markov decision process. The following section presents more details about the Markov decision process formulation and the major components of the method.

6.3.2.1 Agent

In this problem, we used multi-agent reinforcement learning and considered each courier as an independent agent. We formed more than two agents $\{A_1, A_2, \cdots A_n\}$, to assign the learning policy of the agent, respectively.

The agent aims to save the obfuscated trajectory instead of the original one, and this includes the obfuscated customer location. The agent constructs N different obfuscated trajectories by using differential privacy and injecting Laplace noise to selected points into the trajectory. The agent then picks up one of the constructed trajectories.

6.3.2.2 State

The state in our dynamic model is defined based on three components: the selected trajectory, the frequency of the customer order and the sensitivity of destination location. The following are more details for each of them.

- **The selected trajectory:** Each time the courier receives a new delivery request and then delivers it to the customer, the agent, at this point, constructs N number of obfuscated trajectories. Then the agent selects one of the obfuscated trajectories. In our state, we use the chosen trajectory as a component of the state.

- **The frequency of the customer order:** An important factor that has been considered is the frequency of orders by the same customer. This factor shows the number of repeat orders by a particular customer. A high number of repeated orders could increase the probability of revealing the customer's information as the attacker could link the records and infer the customer's information.

- **The sensitivity of destination location:** The food delivery order could be delivered to a sensitive location such as police or security offices. We use this factor as a component of the state.

6.3.2.3 Action

The agent, in each time step t has N actions. The agent action is to select one of the constructed obfuscated trajectories.

6.3.2.4 Reward Function

All agents receive an immediate reward r_t from the environment after taking actions to evaluate the collective actions. In each time step t an instant reward is received by the agent. Only if the agent is able to achieve the desired privacy preservation will the agent receive a positive reward, otherwise negative reward is received. The reward function is defined based on the following:

$$(SimTr \times 0.5) + (SenLoc \times 0.25) + (FreOr \times 0.25) \tag{6.1}$$

Where $SimTr$ represents the similarity between the selected trajectory and the original trajectory. $SenLoc$ represents the sensitivity of destination location, and $FreOr$ represents the frequency of customer orders of online food delivery.

6.3.2.5 The Similarity of the Selected Trajectory

We compared the similarity between the original trajectory and the chosen obfuscated trajectory using the Hausdorff Distance (HD). Hausdorff distance is broadly used to measure the similarity of two datasets of points of D with respect to dataset $\tilde{D}$ [51]:

$$similarity(D) = (max\{h(D,\tilde{D}), h(\tilde{D},D)\} \times 100) \tag{6.2}$$

6.3.2.6 The Sensitivity of the Destination Location

Some online food orders could be delivered to sensitive locations such as some government offices or security sites like police, military or national security centres. Disclosing such information could reveal some private information, like customer job location, customer behavioural pattern, or any other private information.

Thus, we consider this factor to provide more protection for customers with more sensitive information. Haversine formula was used to measure 100 meters from the entered coordinate of each sensitive location entered. Equation 6.3 shows how sensitivity is calculated in our model.

$$SenLoc = \begin{cases} 1, & \text{if } yes. \\ 0, & \text{if } no. \end{cases} \tag{6.3}$$

6.3.2.7 Frequency of the Number of Orders by the Customer

Another important factor that has been considered is the frequency of orders by the same customer. The high number of repeat orders by a particular customer could increase the chance of disclosing the customer's information as the attacker could be able to link the records and reveal private information. The frequency of orders by the customer can be calculated based on the following.

$$\mathscr{Y} = ((x - min(x))/(max(x) - min(x))) \tag{6.4}$$

Where x is the number of frequent orders, $min(x)$ is the minimum frequent order in the dataset and $max(x)$ is the maximum frequent order in the dataset.

6.4 Methodology

6.4.1 QMIX Algorithm

QMIX is a novel value-based technique for training decentralized policies in a centralized end-to-end way. Each agent in a multi-agent framework chooses an action, forming a collective action a_t, and shares a global immediate reward r_t that evaluates the collective action taken previously. There is collective agent-value function $Q_{tot}(s_t, a_t) = \mathbb{E}_{s_t+1:\infty, a_t+1:\infty}[R_t|s_t, a_t]$ for the collective action, as R_t is the discounted return at time t. Evaluating the contribution of each agent separately and accurately to obtain individual value function from the collective action-value function is the main challenge in the multi-agent reinforcement learning framework. To represent the individual value functions of agents A_i, it used $Q_i(o_t, a_t)$ [79].

6.4.2 Protect Trajectory Privacy in Food Delivery with Multi Agent Reinforcement Learning

This section explains how multi-ageing reinforcement learning and Protect Trajectory and Location in Food Delivery (PTLFD) work. Each time, the agent receives an order and delivers it to the customer. The agent performs the obfuscated method using algorithm 6. The following provides more details on how both multi-agent reinforcement learning and PTLFD algorithms work.

6.4.2.1 QMIX Multi Agent Reinforcement Learning

In the training process, we minimize the loss function using gradient descent with respect to parameter θ at iteration i. The neural network for this purpose is used as an approximator in reinforcement learning. Algorithm (5) presents an overview of the QMIX algorithm for multi-agent reinforcement learning. Initially, the reply buffer $\mathscr{D}$ with size $\mathscr{N}$ is initialized. In lines 2 to 4, the action value function is initialised with random weight θ and target action value function with $\bar{\theta} \leftarrow \theta$ and mixing network of the QMIX. The environment then is rested and while not in termination, the agents observe the current observation and the action is selected based on probability ϵ, or otherwise the highest

Algorithm 5 Overview of QMIX Multi-agent reinforcement learning

Initialize: reply buffer $\mathscr{D}$ with size $\mathscr{N}$
Initialize: action value function with random weight θ
Initialize: target action value function with $\bar{\theta} \leftarrow \theta$
Initialize: mixing network ϕ
while Episodes training not finished **do**
    Reset environment
    while The state not in terminal **do**
        for agent in Agents **do**
            Agent observe state o_i^t
            With probability ε select a random action a_t
            Otherwise select $argmax_{a_t}(o_t, a_t : \theta)$
        end for
        Execute joint action a_t and observe next state s_{t+1} , receive reward r_t and termination info
        Add (o_t, a_t, s_{t+1}, r_t) to reply buffer $\mathscr{D}$
        Sample random minibatch $\mathscr{K}$ from $\mathscr{D}$
        $Q_{tot} = mixing((Q_1(o_t^1, a_t^1) + Q_2(o_t^2, a_t^2) + ...Q_n(o_t^n, a_t^n))$
        Calculate loss function $\mathscr{L} = \left(r_t + \gamma max_{a_t} Q_{tot}(s_t, a_t|\bar{\theta}) - Q_{tot}(s_t, a_t|\theta)\right)^2$
        Perform a gradient descent step on $\mathscr{L}$ to the network parameter θ
        Update target network parameter $\bar{\theta}$
        Update exploration rate ϵ
    end while
end while

value is selected from $argmax_{a_t}(o_t, a_t : \theta)$. The action is executed, the corresponding rewards are received, and observe the next state. Add current observations, action, state and reward to reply buffer $\mathscr{D}$. Extract sample from reply buffer $\mathscr{D}$, calculate loss function $\mathscr{L}$. Perform a gradient descent step with network parameter θ and update the target network parameter and then update exploration rate ε.

6.4.2.2 Protect Trajectory and Location in Food Delivery (PTLFD)

Preserving the privacy of customer information in food delivery service is crucial. This method aims to maintain privacy in food delivery services. To obtain better results, some vital factors are considered, such as the similarity between the selected trajectory and the original trajectory, the sensitivity of destination location and the customer frequency of online food delivery orders. In algorithm (6), after the agent receives the food delivery request, the customer location and the courier starting point coordinates are sent to Google Maps API to fetch the trajectory of the courier as a dataset of

coordinates. The courier then delivers the order to the destination. The agent uses the Laplace mechanism to construct N number of obfuscated trajectories with different privacy budgets by injecting noise into a selected point of the original trajectory. Multi-agent reinforcement learning selects one of the constructed obfuscated trajectories. The multi-agent reinforcement learning reward function evaluates the selected obfuscated trajectory based on the similarity between the selected trajectory and the original trajectory, the sensitivity of the destination location, and the frequency of the number of orders by the customer.

Algorithm 6 Protect Trajectory and Location in Food Delivery (PTLFD)

1: **Input** A food delivery order information (Merchant location and customer location)
2: **Input** Trajectory dataset from Google Maps API
3: Courier delivers the order
4: Construct N number of obfuscated trajectories by injecting Laplace noise into a selected point of the original trajectory
5: Check if the order destination is a sensitive location
6: Use multi-agent reinforcement learning to select trajectory
7: Used the selected obfuscated trajectory
8: **Output** Obfuscated trajectory

6.5 Experiment Design

In the following section, we describe the experiment setup for PTLFD and provide details about the dataset that were used in these experiments.

6.5.1 Protect Trajectory and Location in Food Delivery (PTLFD)

Protecting customer data and customer location information is essential. In this experiment, we used PTLFD to protect the customer's location information in online food delivery services. The PTLFD employs the differential privacy Laplace mechanism along with the multi-agent reinforcement learning QMIX method. We run the experiment for more than 3000 runs times, and we manually enter a list of expected sensitive locations. Our experiment was implemented on the Iowa, USA dataset.

6.5.2 Trajectory Data Utility

After applying our obfuscated method to protect the customer's location information, the data utility is definitely affected. To measure the data utility after applying the obfuscated method, we used Hausdorff Distance and Dynamic time warping (DTW), which are two methods to evaluate the similarity between the original trajectory and the newly selected trajectory.

Dynamic time warping (DTW) is a method that aims to find an optimal alignment between two time series, where one of the time series could be warped not linearly by shrinking or stretching its time axis. To determine the similarity between the two time series or to find corresponding regions, this alignment can be used for this purpose. For one curve onto the other curve, the Dynamic time warping calculates the distance between points of these curves. Let us suppose curve C with c number of points and a curve D with d number of points, for every c point the distance between c and each d point is calculated [82] [43].

6.5.3 Analyze Privacy Parameter Intensity

Each time the courier picks up an order and deliver it to the customer, the agent constructs N number of obfuscated trajectories with different privacy parameter. The multi-agent reinforcement learning then picks up one of the constructed trajectories. In order to evaluate the behaviour of our model, we analyzed the privacy parameter resulting from using our model. This experiment shows the distribution of the privacy parameters comes after using the PTLFD method, which applies to Iowa, USA, dataset.

6.5.4 Analyze Driver Journey Time

In this section, we analyze the approximate time needed to deliver the food by the courier between the original trajectory delivery time and the newly constructed trajectory delivery time. In this experiment, we consider that the speed of the courier is fixed, which is 35 kilometres per hour, and we used the Haversine method to measure the distance.

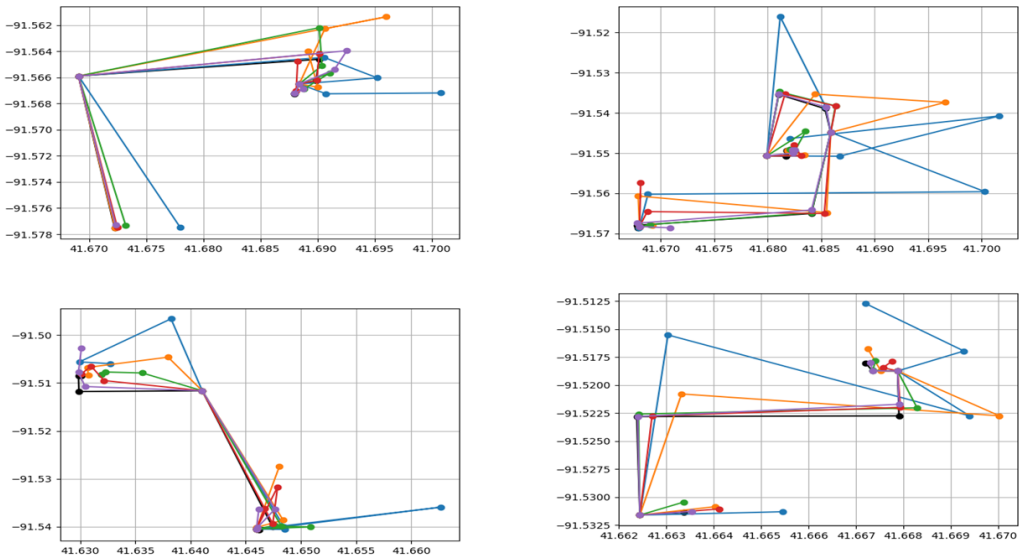


Figure 6.2: Samples of constructed trajectories and the original trajectory, where the original trajectory is shown in black.

6.5.5 Multi Agent Reinforcement Learning Rewards

Each time after taking action, an immediate reward is received by the agents from the environment in order to evaluate the collective actions. Only if the agent is able to achieve the desired privacy preservation goal the agent will receive a positive reward, otherwise negative reward is received. In this experiment, we analyze the average accumulated rewards the agent receives.

6.5.6 Dataset Description

In this work, we use the meal delivery services dataset provided by Ulmer et al. [100] from Iowa, USA. The dataset contains 1,200,391 records and 111 restaurants. The used locations in this dataset are real. The orders are randomly generated with equal request probability for every location and every point of time.

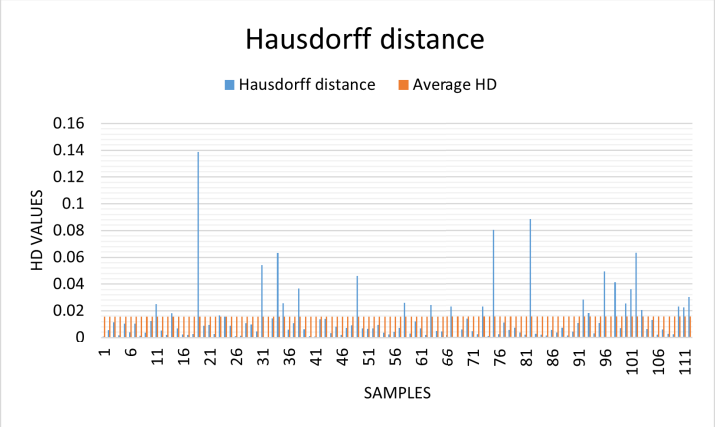


Figure 6.3: Trajectory similarity between the original trajectory and the selected constructed trajectory using Hausdorff distance for more than 100 samples with an average value.

6.6 Experiment Results

6.6.1 Trajectory Similarity and Data Utility

This section shows the similarity results between the selected and the original trajectory after using the Protect Trajectory and Location in Food Delivery method (PTLFD). Figure 6.2 shows samples of the constructed trajectories compared with the original trajectory. This figure shows the original trajectory in black along with the other constructed trajectories. This demonstrates how much similarity there is between the original trajectory and the constructed trajectories.

We also show the similarity using two measurement methods: Hausdorff distance and Dynamic time warping. Figures 6.3 and 6.4 show the results for more than 100 samples of Hausdorff distance and Dynamic time warping value, in this experiment, the model has been trained for more than 3000 run times. Figure 6.3 shows the values of Hausdorff distance for more than 100 samples in blue and the average value in orange. The values are approximate fluctuations between 0.01 and 0.14 with an average about 0.01. When the value of Hausdorff distance is small, this means the similarity between the original trajectory and the obfuscated trajectory is high, and if the value is zero, this means both trajectories are identical. Figure 6.4 shows the result of Dynamic time warping in blue and the average value in orange colour. The Figure shows the values of Dynamic time warping are approximate fluctuations between 0.01 to 0.5 the average is

about 0.05, the smaller value means more similarity.

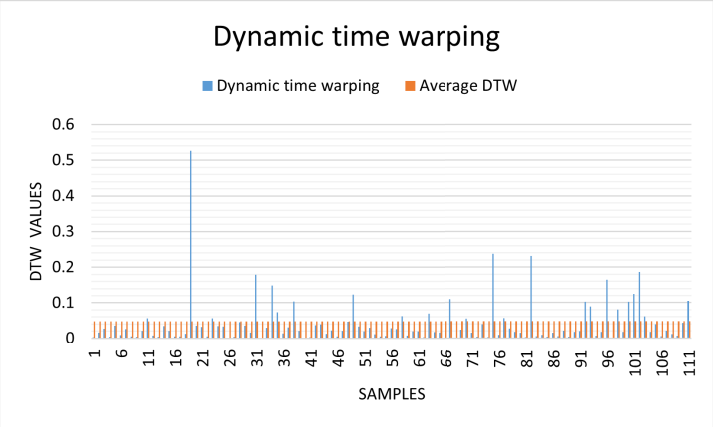


Figure 6.4: Trajectory similarity between the original trajectory and the selected constructed trajectory using the Dynamic time warping for more than 100 samples with an average value.

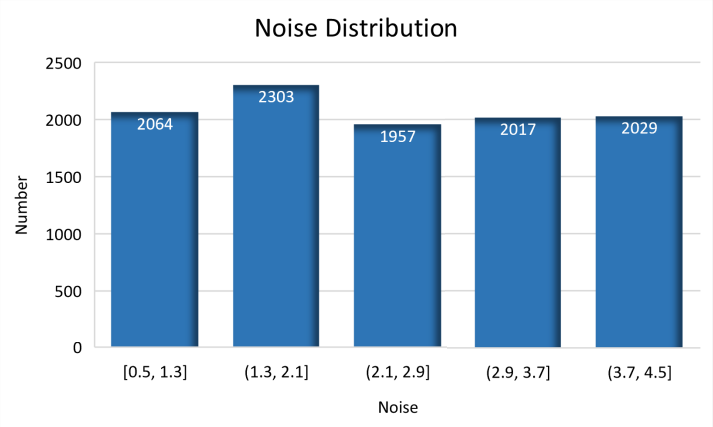


Figure 6.5: Distribution of the used privacy parameter in 100 samples of the selected trajectories by (PTLFD) method.

6.6.1.1 Analyze Privacy Parameter Intensity

Each time the PTLFD method selects one obfuscated trajectory, each trajectory has a different privacy parameter. Figure 6.5 demonstrates the frequency of the privacy parameter used during the training process, where the privacy parameter between 0.5 to 1.3 was used 2064 times, the privacy parameter between 1.3 to 2.1 was used 2303 times, and the privacy parameter between 2.1 to 2.9 was used 1957 times, privacy parameter between 2.9 to 3.7 was used 2017 times and privacy parameter between 3.7 to 4.5 was used 2029 times.

6.6.1.2 Analyze Driver Journey Time

In this section, we compare the journey time of the original trajectory and the journey time for the newly selected obfuscated trajectory that results by using PTLFD method. Figure 6.6 shows the original journey time in blue and the new obfuscated journey time in orange colour. This shows that the time taken for the journey with an obfuscated trajectory is usually longer in most cases.

6.6.1.3 Multi Agent Reinforcement Learning Rewards

This section shows the obtained multi-agent reinforcement learning average accumulative reward by the agents during the learning process. Figure 6.7 shows the average accumulative rewards for more than 3000 run times, showing that the average reward is more than -0.5.

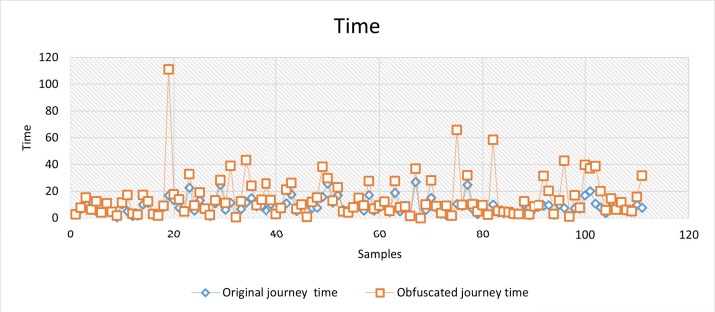


Figure 6.6: Comparison of the journey time of the courier between the original journey time and the newly obfuscated trajectories. The original journey time is shown in blue, and the newly obfuscated journey time is shown in orange.

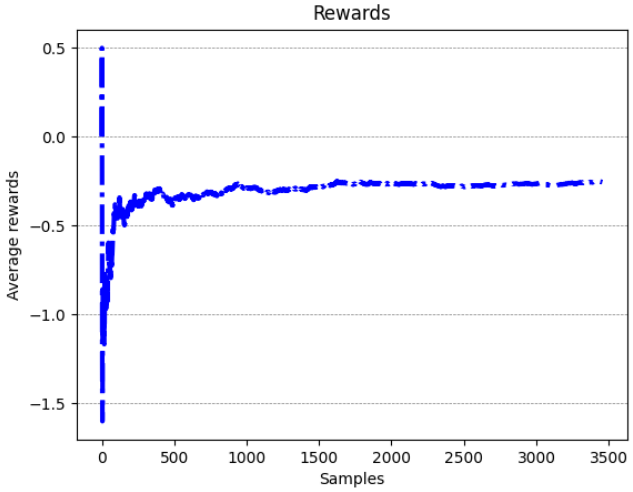


Figure 6.7: Multi agent reinforcement learning average accumulative rewards after more than 3000 run times

6.7 Conclusion

We considered the issue of maintaining the data privacy of saved customer information in food delivery services. We proposed the Protect Trajectory and Location in Food Delivery (PTLFD) method that uses differential privacy Laplace mechanism and multi-agent reinforcement learning to protect customer information privacy in food delivery services. First, the agent delivered the food to the customer, and then the agent constructed N number of obfuscated trajectories with different privacy budgets. Multi-agent reinforcement learning then chose one trajectory from the constructed trajectories. The selected trajectory is then evaluated in the reward function based on the similarity between the selected trajectory and the original trajectory, the sensitivity of the destination location, and the frequency of the number of orders by the customer.

CHAPTER 7

CONCLUSION

7.1 Conclusion

Deep reinforcement learning (DRL) is one of the promising fields in artificial intelligence that has been extensively studied by researchers in past years. Multi-agent reinforcement learning (MARL) and DRL are used in several fields, including recommendation systems, economics, health, gaming and robotics. Although these algorithms achieve promising results, there is significant concern about privacy and the potential for disclosing private information. This research considers deep reinforcement learning, multi-agent reinforcement learning and the related privacy issues. In this research, there are four main chapters, each of which tackles a different problem. Further details on these chapters are provided in the following.

Chapter 3, considers the issue of online food delivery services and how to increase the number of online food delivery orders received by the courier and in doing so, increase the long-term income of the courier. Multi-agent reinforcement learning is employed with two methods QMIX and IQL, to achieve good results. The map of the city is divided into a number of small grids, and each grid represents a particular area of the city. The agent has to run and learn which grid has the highest demand for food delivery orders. The agent has to learn to select the grid with the highest number of food delivery orders to increase the number of food delivery orders received and thereby increase the courier's long-term income. The experiment was conducted using two multi-agent reinforcement

learning methods QMIX and IQL, and one single agent to provide more results. We used two different datasets in this experiment, one for a small city and the second for a large city (Iowa, USA and Shenzhen, China) and another random synthetic dataset. The results demonstrate that there is an increase in the average accumulated reward by using QMIX and IQL. The experiment shows that the average number of food delivery orders increased with the Shenzhen and Iowa datasets and by using the QMIX method.

Chapter 4, considers the Double and Dueling Deep-Q-Network algorithms and how to preserve privacy when using these two algorithms. The Differentially Private SGD is adopted to inject controlled Gaussian noise into the gradients to protect privacy. Two Double DQN experiments were conducted. The first experiment was applied to the Health System Simulations model and the second was applied to the LunarLander OpenAI Gym. Also, two experiments were conducted for the Dueling DQN, the Health System Simulations model experiment and the Ambulance Location Problem. For all the experiments, different noise levels were injected to demonstrate the effectiveness of the injected noise and its ability to preserve privacy. The effect of noise is very clear on the agent's performance, namely increased noise leads to decreased accuracy, and the agent with low noise levels is able to reach convergence and optimal policy.

Chapter 5, considers the privacy issue in online food delivery services and how to protect the customer's location information. To tackle this issue, the Protect User Location Method (PULM) was proposed to protect the customer's location when using online food delivery services. PULM injects Differential Privacy Laplace noise into the customer's location and the courier's trajectory where the PULM uses the city area size and frequency of customer orders as two parameters to determine the amount of injected noise. The Hausdorff Distance method was used to evaluate the data's utility and the similarity before and after using PULM. This chapter also details the distribution of privacy parameter ϵ generated by PULM over two data sets, Shenzhen, China and Iowa, USA. Moreover, we show the generated noise compared with the frequency of customer orders in Shenzhen and Iowa datasets.

Chapter 6, considers the issue of maintaining customer data privacy when using online food delivery services as this is still a challenging issue. We propose the Protect Trajectory and Location in Food Delivery (PTLFD) method which leverages multi-agent reinforcement learning and differential privacy. In the PTLFD, the courier receives

the order and then delivers the order to the customer. The agent then constructs N of obfuscated trajectories with different privacy budgets. Multi-agent reinforcement learning then chooses one of the constructed trajectories. The selected trajectory is evaluated based on three factors: the similarity between the selected trajectory and the original trajectory, the sensitivity of the delivery destination location (such as police or security offices) and the frequency of the customer's online food delivery orders. To demonstrate the effectiveness of our results, we highlight the similarity between the selected and the original trajectory after the PTLFD using Hausdorff distance and the dynamic time warping methods. We detail the distribution of the privacy parameter ϵ, and compare the journey time of the original trajectory and the new obfuscated trajectories.

7.2 Future work

Although this research proposes solutions to different problems related to online food delivery and the related privacy issues, there is still space to improve this work and develop different methods that can achieve better results. The following are some suggestions to improve the current work.

The food delivery service is a well-used service nowadays. A large number of companies in many countries provide food delivery services to millions of people around the world. There is a great opportunity to improve this service as multi-agent reinforcement learning offers an effective solution to enable multiple agents to learn in the same environment and share their knowledge with each other, which accelerates the agents' learning rate. Therefore, there is an excellent opportunity to propose a new multi-agent reinforcement learning method or improve the current work to enhance the learning rate of the agent and thereby improve the agents' performance.

Even though there is increased demand for online food delivery services, there is huge concern about data privacy. Disclosing customer data could lead to harmful consequences as this could reveal critical information about the user, such as the customer's home location, job location or their behaviour patterns, information which could potentially be improperly used. Various mechanisms have been proposed to tackle this issue, and there is a need for an algorithm that can tradeoff between obfuscating the information and utility. Moreover, adopting advanced methods such as new methods of machine learning

can help to achieve remarkable outcomes.

BIBLIOGRAPHY

[1] *Population of tokyo - tokyo metropolitan government*.
[online]. Available at: https://www.metro.tokyo.lg.jp/english/about/history/history03.htm (Accessed: May. 09, 2023).

[2] *State of privacy brazil | privacy international*.
[online]. Available at: `https://privacyinternational.org/state-privacy/42/state-privacy-brazil#righttoprivacy`, (Accessed: May. 09, 2023).

[3] M. ABADI, A. CHU, I. GOODFELLOW, H. B. MCMAHAN, I. MIRONOV, K. TALWAR, AND L. ZHANG, *Deep learning with differential privacy*, in Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, 2016, pp. 308–318.

[4] S. ABAHUSSEIN, Z. CHENG, T. ZHU, D. YE, AND W. ZHOU, *Privacy-preserving in double deep-q-network with differential privacy in continuous spaces*, in Australasian Joint Conference on Artificial Intelligence, Springer, 2022, pp. 15–26.

[5] S. ABAHUSSEIN, D. YE, C. ZHU, Z. CHENG, U. SIDDIQUE, AND S. SHEN, *Multi-agent reinforcement learning for online food delivery with location privacy preservation*, Information, 14 (2023), p. 597.

[6] S. ABAHUSSEIN, T. ZHU, D. YE, Z. CHENG, AND W. ZHOU, *Protect trajectory privacy in food delivery with differential privacy and multi-agent reinforcement learning*, in International Conference on Advanced Information Networking and Applications, Springer, 2023, pp. 48–59.

[7] M. ALLEN AND T. MONKS, *Integrating deep reinforcement learning networks with health system simulations*, arXiv preprint arXiv:2008.07434, (2020).

[8] M. ALLEN, K. PEARN, AND T. MONKS, *Developing an openai gym-compatible framework and simulation environment for testing deep reinforcement*

learning agents solving the ambulance location problem, arXiv preprint arXiv:2101.04434, (2021).

[9] M. E. ANDRÉS, N. E. BORDENABE, K. CHATZIKOKOLAKIS, AND C. PALAMIDESSI, *Geo-indistinguishability: Differential privacy for location-based systems*, in Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, 2013, pp. 901–914.

[10] S. ARORA AND P. DOSHI, *A survey of inverse reinforcement learning: Challenges, methods and progress*, Artificial Intelligence, 297 (2021), p. 103500.

[11] K. ARULKUMARAN, M. P. DEISENROTH, M. BRUNDAGE, AND A. A. BHARATH, *Deep reinforcement learning: A brief survey*, IEEE Signal Processing Magazine, 34 (2017), pp. 26–38.

[12] D. BADZIAHIN AND E. ZORIN, *On generalized thue-morse functions and their values*, arXiv preprint arXiv:1509.00297, (2015).

[13] C. BAKER, *City and town classification of constituencies and local authorities*. [online]. Available at: https://commonslibrary.parliament.uk/research-briefings/cbp-8322 (Accessed: Aug. 15, 2023).

[14] B. BALLE, M. GOMROKCHI, AND D. PRECUP, *Differentially private policy evaluation*, in International Conference on Machine Learning, PMLR, 2016, pp. 2130–2138.

[15] V. BEHZADAN AND A. MUNIR, *Vulnerability of deep reinforcement learning to policy induction attacks*, in Machine Learning and Data Mining in Pattern Recognition: 13th International Conference, MLDM 2017, New York, NY, USA, July 15-20, 2017, Proceedings 13, Springer, 2017, pp. 262–275.

[16] S. BERRI, J. ZHANG, B. BENSAOU, AND H. LABIOD, *Privacy-preserving data-prefetching in vehicular networks via reinforcement learning*, in ICC 2020-2020 IEEE International Conference on Communications (ICC), IEEE, 2020, pp. 1–6.

[17] C. BETANCOURT AND W.-H. CHEN, *Deep reinforcement learning for portfolio management of markets with a dynamic number of assets*, Expert Systems with Applications, 164 (2021), p. 114002.

[18] O. BOUFOUS, *Deep reinforcement learning for complete coverage path planning in unknown environments*, 2020.

[19] A. BOZANTA, M. CEVIK, C. KAVAKLIOGLU, E. M. KAVUK, A. TOSUN, S. B. SONUC, A. DURANEL, AND A. BASAR, *Courier routing and assignment for food delivery service using reinforcement learning*, Computers & Industrial Engineering, 164 (2022), p. 107871.

[20] L. BUSONIU, R. BABUSKA, AND B. DE SCHUTTER, *A comprehensive survey of multiagent reinforcement learning*, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 38 (2008), pp. 156–172.

[21] L. CANESE, G. C. CARDARILLI, L. DI NUNZIO, R. FAZZOLARI, D. GIARDINO, M. RE, AND S. SPANÒ, *Multi-agent reinforcement learning: A review of challenges and applications*, Applied Sciences, 11 (2021), p. 4948.

[22] L. T. CHAI AND D. N. C. YAT, *Online food delivery services: Making food delivery the new normal*, Journal of Marketing advances and Practices, 1 (2019), pp. 62–77.

[23] H.-G. CHEN, C. C. CHEN, L. LO, AND S. C. YANG, *Online privacy control via anonymity and pseudonym: Cross-cultural implications*, Behaviour & Information Technology, 27 (2008), pp. 229–242.

[24] R. CHEN, B. FUNG, AND B. C. DESAI, *Differentially private trajectory data publication*, arXiv preprint arXiv:1112.2020, (2011).

[25] X. CHEN, M. W. ULMER, AND B. W. THOMAS, *Deep q-learning for same-day delivery with vehicles and drones*, European Journal of Operational Research, 298 (2022), pp. 939–952.

[26] X. CHEN, T. ZHANG, S. SHEN, T. ZHU, AND P. XIONG, *An optimized differential privacy scheme with reinforcement learning in vanet*, Computers & Security, 110 (2021), p. 102446.

[27] Z. CHENG, D. YE, T. ZHU, W. ZHOU, P. S. YU, AND C. ZHU, *Multi-agent reinforcement learning via knowledge transfer with differentially private noise*, International Journal of Intelligent Systems, 37 (2022), pp. 799–828.

[28] T. CHIBA, Y. SEI, Y. TAHARA, AND A. OHSUGA, *Trajectory anonymization: Balancing usefulness about position information and timestamp*, in 2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS), IEEE, 2019, pp. 1–6.

[29] W. CONTRIBUTORS, *Wikipedia contributors*.
[online]. Available at: https://en.wikipedia.org/wiki/Shenzhen (Accessed: Aug. 30, 2023).

[30] T. S. P. DE SOUZA, R. F. MIYAHIRA, J. R. V. MATHEUS, T. B. DE BRITO NOGUEIRA, C. MARAGONI-SANTOS, F. F. C. BARROS, A. E. C. ANTUNES, AND A. E. C. FAI, *Food services in times of uncertainty: Remodeling operations, changing trends, and looking into perspectives after the covid-19 pandemic*, Trends in Food Science & Technology, (2022).

[31] F. DELDAR AND M. ABADI, *Pldp-td: personalized-location differentially private data analysis on trajectory databases*, Pervasive and Mobile Computing, 49 (2018), pp. 1–22.

[32] Y. DING, B. GUO, L. ZHENG, M. LU, D. ZHANG, S. WANG, S. H. SON, AND T. HE, *A city-wide crowdsourcing delivery system with reinforcement learning*, Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 5 (2021), pp. 1–22.

[33] V. DOLK, *Survey reinforcement learning*, Eindhoven University of Technology, (2010).

[34] Y. DU, L. HAN, M. FANG, J. LIU, T. DAI, AND D. TAO, *Liir: Learning individual intrinsic reward in multi-agent reinforcement learning*, Advances in Neural Information Processing Systems, 32 (2019).

[35] C. DWORK, K. KENTHAPADI, F. MCSHERRY, I. MIRONOV, AND M. NAOR, *Our data, ourselves: Privacy via distributed noise generation*, in Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2006, pp. 486–503.

[36] C. DWORK, F. MCSHERRY, K. NISSIM, AND A. SMITH, *Calibrating noise to sensitivity in private data analysis*, in Theory of cryptography conference, Springer, 2006, pp. 265–284.

[37] E. ERDEMIR, P. L. DRAGOTTI, AND D. GÜNDÜZ, *Privacy-aware location sharing with deep reinforcement learning*, in 2019 IEEE International Workshop on Information Forensics and Security (WIFS), IEEE, 2019, pp. 1–6.

[38] V. FRANÇOIS-LAVET, P. HENDERSON, R. ISLAM, M. G. BELLEMARE, J. PINEAU, ET AL., *An introduction to deep reinforcement learning*, Foundations and Trends in Machine Learning, 11 (2018), pp. 219–354.

[39] H. HASSELT, *Double q-learning*, Advances in neural information processing systems, 23 (2010).

[40] S. HU, B. GUO, S. WANG, AND X. ZHOU, *Effective cross-region courier-displacement for instant delivery via reinforcement learning*, in International Conference on Wireless Algorithms, Systems, and Applications, Springer, 2021, pp. 288–300.

[41] R.-H. HWANG, Y.-L. HSUEH, AND H.-W. CHUNG, *A novel time-obfuscated algorithm for trajectory privacy protection*, IEEE Transactions on Services Computing, 7 (2013), pp. 126–139.

[42] H. JAHANSHAHI, A. BOZANTA, M. CEVIK, E. M. KAVUK, A. TOSUN, S. B. SONUC, B. KOSUCU, AND A. BAŞAR, *A deep reinforcement learning approach for the meal delivery problem*, Knowledge-Based Systems, 243 (2022), p. 108489.

[43] C. F. JEKEL, G. VENTER, M. P. VENTER, N. STANDER, AND R. T. HAFTKA, *Similarity measures for identifying material parameters from hysteresis loops using inverse analysis*, International Journal of Material Forming, 12 (2019), pp. 355–378.

[44] M. I. JORDAN AND T. M. MITCHELL, *Machine learning: Trends, perspectives, and prospects*, Science, 349 (2015), pp. 255–260.

[45] L. P. KAELBLING, M. L. LITTMAN, AND A. W. MOORE, *Reinforcement learning: A survey*, Journal of artificial intelligence research, 4 (1996), pp. 237–285.

[46] H. KARGUPTA, S. DATTA, Q. WANG, AND K. SIVAKUMAR, *On the privacy preserving properties of random data perturbation techniques*, in Third IEEE international conference on data mining, IEEE, 2003, pp. 99–106.

[47] I. KHACHEBA, M. B. YAGOUBI, N. LAGRAA, AND A. LAKAS, *Clps: context-based location privacy scheme for vanets*, International Journal of Ad Hoc and Ubiquitous Computing, 29 (2018), pp. 141–159.

[48] J. KOBER, J. A. BAGNELL, AND J. PETERS, *Reinforcement learning in robotics: A survey*, The International Journal of Robotics Research, 32 (2013), pp. 1238–1274.

[49] M. LAPAN, *Deep Reinforcement Learning Hands-On - Second Edition*, Packt Publishing Ltd, 2020.

[50] M. LI, X. GU, C. ZENG, AND Y. FENG, *Feasibility analysis and application of reinforcement learning algorithm based on dynamic parameter adjustment*, Algorithms, 13 (2020), p. 239.

[51] M. LI, L. ZHU, Z. ZHANG, AND R. XU, *Achieving differential privacy of trajectory data publishing in participatory sensing*, Information Sciences, 400 (2017), pp. 1–13.

[52] Y. LI, *Deep reinforcement learning: An overview*, arXiv preprint arXiv:1701.07274, (2017).

[53] E. LIEBMAN, M. SAAR-TSECHANSKY, AND P. STONE, *Dj-mc: A reinforcement-learning agent for music playlist recommendation*, arXiv preprint arXiv:1401.1880, (2014).

[54] M. LITTMAN AND A. MOORE, *Reinforcement learning: A survey, journal of artificial intelligence research 4*, 1996.

[55] D. LIU AND N. CHEN, *Satellite monitoring of urban land change in the middle yangtze river basin urban agglomeration, china between 2000 and 2016*, Remote Sensing, 9 (2017), p. 1086.

[56] S. LIU, H. JIANG, S. CHEN, J. YE, R. HE, AND Z. SUN, *Integrating dijkstra's algorithm into deep inverse reinforcement learning for food delivery route planning*, Transportation Research Part E: Logistics and Transportation Review, 142 (2020), p. 102070.

[57] X. LIU, R. H. DENG, K.-K. R. CHOO, AND Y. YANG, *Privacy-preserving reinforcement learning design for patient-centric dynamic treatment regimes*, IEEE Transactions on Emerging Topics in Computing, 9 (2019), pp. 456–470.

[58] Z. LIU, C. YAO, H. YU, AND T. WU, *Deep reinforcement learning with its application for lung cancer detection in medical internet of things*, Future Generation Computer Systems, 97 (2019), pp. 1–9.

[59] N. C. LUONG, D. T. HOANG, S. GONG, D. NIYATO, P. WANG, Y.-C. LIANG, AND D. I. KIM, *Applications of deep reinforcement learning in communications and*

networking: A survey, IEEE Communications Surveys & Tutorials, 21 (2019), pp. 3133–3174.

[60] P. MA, Z. WANG, L. ZHANG, R. WANG, X. ZOU, AND T. YANG, *Differentially private reinforcement learning*, in International Conference on Information and Communications Security, Springer, 2019, pp. 668–683.

[61] B. MAHESH, *Machine learning algorithms-a review*, International Journal of Science and Research (IJSR).[Internet], 9 (2020), pp. 381–386.

[62] P. MALIK, M. PATHANIA, V. K. RATHAUR, ET AL., *Overview of artificial intelligence in medicine*, Journal of family medicine and primary care, 8 (2019), p. 2328.

[63] M. MIN, W. WANG, L. XIAO, Y. XIAO, AND Z. HAN, *Reinforcement learning-based sensitive semantic location privacy protection for vanets*, China Communications, 18 (2021), pp. 244–260.

[64] M. MIN, S. YANG, S. LI, H. ZHANG, L. XIAO, M. PAN, AND Z. HAN, *Safe reinforcement learning-based indoor 3d location privacy protection*, in 2022 IEEE/CIC International Conference on Communications in China (ICCC), IEEE, 2022, pp. 53–58.

[65] V. MNIH, K. KAVUKCUOGLU, D. SILVER, A. GRAVES, I. ANTONOGLOU, D. WIERSTRA, AND M. RIEDMILLER, *Playing atari with deep reinforcement learning*, arXiv preprint arXiv:1312.5602, (2013).

[66] S. S. MOUSAVI, M. SCHUKAT, AND E. HOWLEY, *Deep reinforcement learning: an overview*, in Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016: Volume 2, Springer, 2018, pp. 426–440.

[67] U. NATIONS, *Universal declaration of human rights*.
[online]. Available at: `https://www.un.org/en/about-us/universal-declaration-of-human-rights` (Accessed: May. 09, 2023).

[68] K. NISSIM AND A. WOOD, *Is privacy privacy?*, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 376 (2018), p. 20170358.

[69] OAIC, *The privacy act*.
[online]. Available at: https://www.oaic.gov.au/privacy/the-privacy-act (Accessed: May. 09, 2023).

[70] OAIC, *What is privacy?*
[online]. Available at: https://www.oaic.gov.au/privacy/your-privacy-rights/what-is-privacy (Accessed: May. 09, 2023).

[71] O. OF THE AUSTRALIAN INFORMATION COMMISSIONER, *European court of human rights*.
[online]. Available: at https://www.echr.coe.int (Accessed: Apr. 19, 2022).

[72] N.-S. G. ONLINE, *City population stands at 17.56 million¡atestnews – shenzhengovernmentonline*.
[online]. Available at: http://www.sz.gov.cn/en_szgov/news/latest/content/post_8771774.html (Accessed: Aug. 30, 2023).

[73] A. OPPERMANN, *An introduction to double deep q-learning*.
[online]. Available at: https://builtin.com/artificial-intelligence/double-deep-q-learning (Accessed: Aug. 15, 2023).

[74] X. PAN, W. WANG, X. ZHANG, B. LI, J. YI, AND D. SONG, *How you act tells a lot: Privacy-leaking attack on deep reinforcement learning*, in Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, 2019, pp. 368–376.

[75] M. PANG, L. WANG, AND N. FANG, *A collaborative scheduling strategy for iov computing resources considering location privacy protection in mobile edge computing environment*, Journal of Cloud Computing, 9 (2020), pp. 1–17.

[76] R. C. POST, *Three concepts of privacy*, Geo. LJ, 89 (2000), p. 2087.

[77] A. RAGHU, M. KOMOROWSKI, I. AHMED, L. CELI, P. SZOLOVITS, AND M. GHASSEMI, *Deep reinforcement learning for sepsis treatment*, arXiv preprint arXiv:1711.09602, (2017).

[78] Y. RAO, J. LU, AND J. ZHOU, *Attention-aware deep reinforcement learning for video face recognition*, in Proceedings of the IEEE international conference on computer vision, 2017, pp. 3931–3940.

[79] T. RASHID, M. SAMVELYAN, C. SCHROEDER DE WITT, G. FARQUHAR, J. N. FOERSTER, AND S. WHITESON, *Monotonic value function factorisation for deep multi-agent reinforcement learning*, Journal of Machine Learning Research, 21 (2020).

[80] H. M. R. U. REHMAN, B.-W. ON, D. D. NINGOMBAM, S. YI, AND G. S. CHOI, *Qsod: Hybrid policy gradient for deep multi-agent reinforcement learning*, IEEE Access, 9 (2021), pp. 129728–129741.

[81] C. ROSSET, D. JOSE, G. GHOSH, B. MITRA, AND S. TIWARY, *Optimizing query evaluations using reinforcement learning for web search*, in The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 2018, pp. 1193–1196.

[82] S. SALVADOR AND P. CHAN, *Toward accurate dynamic time warping in linear time and space*, Intelligent Data Analysis, 11 (2007), pp. 561–580.

[83] T. SCHAUL, J. QUAN, I. ANTONOGLOU, AND D. SILVER, *Prioritized experience replay*, arXiv preprint arXiv:1511.05952, (2015).

[84] M. SEWAK, *Deep reinforcement learning*, Springer, Singapore, 2019.

[85] S. SHEN, *Differential Privacy in Reinforcement Learning*, PhD thesis, University of Technology, Sydney (Australia), 2022.

[86] S. SHEN, T. ZHU, D. YE, M. WANG, X. ZUO, AND A. ZHOU, *A novel differentially private advising framework in cloud server environment*, Concurrency and Computation: Practice and Experience, (2020), p. e5932.

[87] R. SHOKRI, M. STRONATI, C. SONG, AND V. SHMATIKOV, *Membership inference attacks against machine learning models*, in 2017 IEEE symposium on security and privacy (SP), IEEE, 2017, pp. 3–18.

[88] D. J. SOLOVE, *Understanding privacy*, (2008).

[89] S. SONG, K. CHAUDHURI, AND A. D. SARWATE, *Stochastic gradient descent with differentially private updates*, in 2013 IEEE Global Conference on Signal and Information Processing, IEEE, 2013, pp. 245–248.

[90] J. SORIA-COMAS AND J. DOMINGO-FERRER, *Differentially private data publishing via optimal univariate microaggregation and record perturbation*, Knowledge-Based Systems, 153 (2018), pp. 78–90.

[91] U. STATES CENSUS BUREAU, *U.s. census bureau quickfacts: Iowa city city, iowa*. [online]. Available at: `https://www.census.gov/quickfacts/fact/table/iowacitycityiowa/PST045222` (Accessed: Aug. 30, 2023).

[92] ——, *U.s. census bureau quickfacts: United states*. [online]. Available at: `https://www.census.gov/quickfacts/fact/table/US/PST045221` (Accessed: May. 09, 2023).

[93] STATISTA, *Online food delivery - worldwide*. [online]. Available at: `https://www.statista.com/outlook/dmo/online-food-delivery/worldwide?currency=usd` (Accessed: Aug. 30, 2023).

[94] R. SUTTON AND A. BARTO, *Reinforcement learning: An introduction*, 2018.

[95] R. S. SUTTON AND A. G. BARTO, *Reinforcement learning: An introduction*.

[96] A. TAMPUU, T. MATIISEN, D. KODELJA, I. KUZOVKIN, K. KORJUS, J. ARU, J. ARU, AND R. VICENTE, *Multiagent cooperation and competition with deep reinforcement learning*, PloS one, 12 (2017), p. e0172395.

[97] X. TANG, L. ZHU, M. SHEN, AND X. DU, *When homomorphic cryptosystem meets differential privacy: training machine learning classifier with privacy protection*, arXiv preprint arXiv:1812.02292, (2018).

[98] A. C. TOSSOU AND C. DIMITRAKAKIS, *Achieving privacy in the adversarial multi-armed bandit*, arXiv preprint arXiv:1701.04222, (2017).

[99] Z. TU, K. ZHAO, F. XU, Y. LI, L. SU, AND D. JIN, *Protecting trajectory from semantic attack considering k -anonymity, l -diversity, and t -closeness*, IEEE Transactions on Network and Service Management, 16 (2018), pp. 264–278.

[100] M. W. ULMER, B. W. THOMAS, A. M. CAMPBELL, AND N. WOYAK, *The restaurant meal delivery problem: Dynamic pickup and delivery with deadlines and random ready times*, Transportation Science, 55 (2021), pp. 75–100.

[101] K. G. VAMVOUDAKIS, Y. WAN, F. L. LEWIS, AND D. CANSEVER, *Handbook of Reinforcement Learning and Control*, Springer, International Publishing, 2021.

[102] H. VAN HASSELT, A. GUEZ, AND D. SILVER, *Deep reinforcement learning with double q-learning*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30, 2016.

[103] R. VARDHMAN, *machine learning statistics and facts - 2021*. [online]. Available at: `https://findly.in/machine-learning-statistics` (Accessed: May. 09, 2023).

[104] B. WANG AND N. HEGDE, *Privacy-preserving q-learning with functional noise in continuous spaces*, in Advances in Neural Information Processing Systems, 2019, pp. 11327–11337.

[105] J. WANG, S. LIU, AND Y. LI, *A review of differential privacy in individual data release*, International Journal of Distributed Sensor Networks, 11 (2015), p. 259682.

[106] S. WANG, D. JIA, AND X. WENG, *Deep reinforcement learning for autonomous driving*, arXiv preprint arXiv:1811.11329, (2018).

[107] W. WANG, M. MIN, L. XIAO, Y. CHEN, AND H. DAI, *Protecting semantic trajectory privacy for vanet with reinforcement learning*, in ICC 2019-2019 IEEE International Conference on Communications (ICC), IEEE, 2019, pp. 1–5.

[108] Z. WANG, T. SCHAUL, M. HESSEL, H. HASSELT, M. LANCTOT, AND N. FREITAS, *Dueling network architectures for deep reinforcement learning*, in International conference on machine learning, PMLR, 2016, pp. 1995–2003.

[109] P. WINDER, *Reinforcement learning*, O'Reilly Media, 2020.

[110] E. XING AND B. CAI, *Delivery route optimization based on deep reinforcement learning*, in 2020 2nd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI), IEEE, 2020, pp. 334–338.

[111] M. YANG, T. ZHU, Y. XIANG, AND W. ZHOU, *Density-based location preservation for mobile crowdsensing with differential privacy*, Ieee Access, 6 (2018), pp. 14779–14789.

[112] X. YANG, T. WANG, X. REN, AND W. YU, *Survey on improving data utility in differentially private sequential data publishing*, IEEE Transactions on Big Data, 7 (2017), pp. 729–749.

[113] Y. YANG, X. BAN, X. HUANG, AND C. SHAN, *A dueling-double-deep q-network controller for magnetic levitation ball system*, in 2020 39th Chinese Control Conference (CCC), IEEE, 2020, pp. 1885–1890.

[114] K.-L. A. YAU, J. QADIR, H. L. KHOO, M. H. LING, AND P. KOMISARCZUK, *A survey on reinforcement learning models and algorithms for traffic signal control*, ACM Computing Surveys (CSUR), 50 (2017), pp. 1–38.

[115] D. YE, S. SHEN, T. ZHU, B. LIU, AND W. ZHOU, *One parameter defense,Äîdefending against data inference attacks via differential privacy*, IEEE Transactions on Information Forensics and Security, 17 (2022), pp. 1466–1480.

[116] D. YE, T. ZHU, S. SHEN, W. ZHOU, AND P. YU, *Differentially private multi-agent planning for logistic-like problems*, IEEE Transactions on Dependable and Secure Computing, (2020).

[117] D. YE, T. ZHU, W. ZHOU, AND S. Y. PHILIP, *Differentially private malicious agent avoidance in multiagent advising learning*, IEEE transactions on cybernetics, 50 (2019), pp. 4214–4227.

[118] W. J. YUN, S. JUNG, J. KIM, AND J.-H. KIM, *Distributed deep reinforcement learning for autonomous aerial evtol mobility in drone taxi applications*, ICT Express, 7 (2021), pp. 1–4.

[119] J. ZHANG, Y.-R. HUANG, Q.-H. HUANG, Y.-Z. LI, AND X.-C. YE, *Hasse sensitivity level: A sensitivity-aware trajectory privacy-enhanced framework with reinforcement learning*, Future Generation Computer Systems, 142 (2023), pp. 301–313.

[120] K. ZHANG, Z. YANG, AND T. BAŞAR, *Multi-agent reinforcement learning: A selective overview of theories and algorithms*, Handbook of reinforcement learning and control, (2021), pp. 321–384.

[121] L. ZHANG, C. JIN, H.-P. HUANG, X. FU, AND R.-C. WANG, *A trajectory privacy preserving scheme in the cannq service for iot*, Sensors, 19 (2019), p. 2190.

[122] X. ZHANG, C. ZHAO, F. LIAO, X. LI, AND Y. DU, *Online parking assignment in an environment of partially connected vehicles: A multi-agent deep reinforcement learning approach*, Transportation Research Part C: Emerging Technologies, 138 (2022), p. 103624.

[123] X. ZHAO, D. PI, AND J. CHEN, *Novel trajectory privacy-preserving method based on clustering using differential privacy*, Expert Systems with Applications, 149 (2020), p. 113241.

[124] K. ZHOU AND J. WANG, *Trajectory protection scheme based on fog computing and k-anonymity in iot*, in 2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS), IEEE, 2019, pp. 1–6.

[125] T. ZHU, G. LI, W. ZHOU, AND S. Y. PHILIP, *Differentially private data publishing and analysis: A survey*, IEEE Transactions on Knowledge and Data Engineering, 29 (2017), pp. 1619–1638.

[126] G. ZOU, J. TANG, L. YILMAZ, AND X. KONG, *Online food ordering delivery strategies based on deep reinforcement learning*, Applied Intelligence, (2022), pp. 1–13.